DON

BOSCOCOLLEGEDEPARTMENTOFCOMPUTER SCIENCE

SUBJECTNAME:PROGRAMMINGINJAVA

PAPERCODE :17UCS09

SEMESTER VI

DBC

UNIT–I

Java Evolution – Simple Java Program – Java program structure – Java Tokens – JavaStatements – JVM – Command Line Arguments – Constants, Variables, and Data Types – OperatorsandExpressions.

UNIT–II

Decision Making and Branching: Introduction – Decision Making with if Statement – Simple if Statement – The if…else Statement – Nesting of if…else Statement – The else ifLadder–TheSwitchStatement–The?:Operator.DecisionMakingandLooping:Introduction – The While Statement – The Do Statement – The For Statement – Jumps in Loops – LabelledLoops. Classes, Objects and Methods: Introduction – Defining a Class – Fields Declaration –Methods Declaration – Creating Objects – Accessing Class Members – Constructors – MethodsOverloading – Static Members– Nesting of Methods– Inheritance – Overriding Methods–FinalVariablesandMethods–FinalClasses–FinalizerMethods–AbstractMethodsandClasses–MethodswithVarargs–VisibilityControl.

UNIT–III

Arrays,StringsandVectors:Introduction–One-dimensionalArrays–CreatinganArray – Two-dimensional Arrays– Strings– Vectors– WrapperClasses– Enumerated Types– Annotations.Interfaces:Introduction–DefiningInterfaces–ExtendingInterfaces–Implementing Interfaces – Accessing Interface Variables – Packages: Introduction – Java APIPackages – Using System Packages – Naming Conventions – Creating Packages – Accessing aPackage– UsingaPackage–AddingaClasstoaPackage –HidingClasses–Static Import.

UNIT–IV

Multithreaded Programming:Introduction – Creating Threads –Extending the Threadclass – Stopping and Blocking a Thread - Life cycle of a Thread – Using Thread methods –Thread Exceptions – Thread Priority – Synchronization – Implementing the Runnable interface –Inter-thread Communication. Managing Errors and Exceptions: Introduction – Types of Errors – Exceptions – Syntax of Exception Handling Code – Multiple Catch Statements – Using FinallyStatement – Throwing Our Own Exceptions. Applet Programming: Introduction – DifferenceBetween Applets and Applications – Write Applets - Building Applet code - Applet life cycle -Creating an Executable Applet – Designing a web page – Applet Tag – Adding Applet to HTMLFile– Running the applet – Applet Tags –Passing Parameters to Applets – Aligning the Display –DisplayingNumerical values– Gettinginput fromtheuser–Event handling.

UNIT– V

Graphics Programming: Introduction – The Graphics Class - Lines and Rectangles – Circles and Ellipses – Drawing Arcs – Drawing polygons – Line Graphs – Using Control LoopsinApplets–DrawingBarCharts.Managing I/OFiles inJava:Introduction– Concept ofstream – Stream classes – Byte stream classes – Character stream classes – Using stream – Using thefile class – Creation of Files – Reading/Writing characters – Reading/Writing Bytes – HandlingPrimitiveDatatypes– ConcatenatingandbufferingBytes-Randomaccess files.

TEXTBOOK

1. E. Balagurusamy, “Programming with Java,” 4th Edition, Tata McGraw Hill Pub. Ltd.,NewDelhi,2009.

REFERENCEBOOKS

1. HerbertSchild,"Java:TheCompleteReference,"NinthEdition,OraclePress,2014

2. RohitKhurana,"ProgrammingwithJAVA,"VIKASPub.,2014

DBC

UNIT

IJAVAEVOLUTIO

N

JavaHistory

- ϖ Javaisahigh-levelprogramminglanguageoriginallydevelopedbySunMicrosystemsandreleasedin1991.
- ϖJavarunsona varietyofplatforms,such asWindows,Mac OS,andthevariousversionsof UNIX. Sojavaisplatformindependent.
- ϖJavawasdesignedforthedevelopmentofsoftwareforconsumerelectronicdeviceslikeTVs,VCRs,toasters and such otherelectronicdevices.
- ϖ1990 ☾ AteamofSunMicrosystemsheadedbyJamesGoslingwasdecidedtodevelopaspecialsoftwarethat canbeused tomanipulateconsumer electronicdevices.
- ϖ1991 ☾ Theteamannouncedanewlanguagenamed“oak”.
- ϖ1992 ☾ Theteamdemonstratedtheapplicationoftheirnewlanguagetocontrolalistofhomeapplications.
- ϖ1993 ☾ The teamknownasGreenProjectteamcame upwiththe ideaof developingwebapplets.
- ϖ1994 ☾ Theteamdevelopedawebbrowsercalled“HotJava”tolocateandrunappletprogramsoninternet.
- ϖ1995 ☾ OakwasrenamedJava.
- ϖ1996 ☾ SunreleasesJavaDevelopmentKit1.0.
- ϖ1997 ☾ SunreleasesJavaDevelopmentKit1.1.
- ϖ1998 ☾ SunreleasestheJava2withversion1.2.
- ϖ1999 ☾ Sunreleasesstandardedition(J2SE)andenterpriseedition(J2EE).
- ϖ2000 ☾ J2SEwithSDK(softwaredevelopmentkit)1.3wasreleased.
- ϖ2002 ☾ J2SEwith SDK1.4wasreleased.
- ϖ2004 ☾ J2SEJDK5.0wasreleased.ThisisknownasJ2SE5.0.

FEATURESOFJAVA

1. Compiledandinterpreted.
2. Objectoriented
3. Distributed
4. Multithreadedand Interactive
6. Platform-independentandportable.
7. Robust andsecure
8. Familiar,simpleandsmall
9. Highperformance

5. DynamicandExtensible

Compiledandinterpreted

ϖJavaisatwo-stagesystem that isbothcompliedandinterpretedlanguage.

ϖFirstjavacompilertranslatessourcecodeintobytecodeinstructions.

ϖThebytecodesarenotmachineinstructions.

ϖSecondstagejavainterpretergeneratesmachinecodethatcanbedirectlyexecutedbythemachinethat runningthejavaprogram.

PlatformIndependentand Portable

ϖThemostsignificantcontributionof java overotherlanguage isitsportability.

ϖJavaprogramscanbeeasilymovedfromonecomputersystemtoanother,anywhereandanytime.

ϖChangesandUpgradesinoperatingsystems,processorsandsystemresourceswillnotforceany changesinJavaprograms.Thisisthereasonwhyjavahasbecomeapopularlanguageforprogrammingon Internet, whichinterconnectsdifferentkindsofsystems worldwide.

ϖWecandownloadajavaappletfromaremotecomputerontoourlocalsystemthroughInternetandexecuteit locally.

ϖJavaensuresportabilityintwoways.FirstjavacompilergeneratesBytecodeinstructionsthat canbeimplementedonanymachine.Secondly,thesizesoftheprimitivedatatypesaremachineindependent.

ObjectOriented:

ϖJavaistrulyobject-orientedlanguage.Almost everythinginJavaisanobject.

ϖAllprogramcodeanddataresidewithinobjectsandclasses.

ϖJavacomeswithanextensivesetoffclasses,arrangedinpackagesthatwecanuseinourprogramsbyinheritance.

ϖTheobjectmodelinjavaissimpleandeasytoextend.

RobustandSecure:

ϖJavaisarobustlanguage.Itprovidesmanysafeguardstoensurereliablecode.Ithasstrictcompiletimeand runtimechecking for datatypes.

ϖItisdesignedasgarbagecollectedlanguagerelievingtheprogrammersvirtuallyallmemory managementproblems.

ϖJavaalsoincorporatestheconceptofexceptionhandlingwhichcapturesserieserrorsand

eliminatesriskofcrashingthesystem.

- Theabsenceofpointersinjavaensuresthatprogramscannotgainaccesstomemorylocationswithoutproper authorization.

Distributed:

- Javaisdesignedasadistributedlanguageforcreatingapplicationsonnetworks.Ithastheabilitytoshareboth dataand programs.
- Javaapplicationscanopenandaccessremoteobjectsoninternetaseasilyastheycandoina localsystem.
- Thisenablesmultipleprogrammersatmultipleremotelocationstocollaborateandworktogetheronasingle project.

Simple,SmallandFamiliar:

- Javaisasmallandsimplelanguage.ManyfeaturesofCandC++thatareeitherredundantorsourcesof unreliablecodearenot partofjava.
- Forexamplejavadoesnotusepointers,preprocessorheaderfiles,gotostatementand overloadingandmultipleinheritance and manyothers.

MultithreadedandInteractive:

- Multithreadedmeanshandlingmultipletaskssimultaneously.Javasupportsmultithreadedprograms.This meansthatweneednotwaitfortheapplicationtofinishonetaskbeforebeginninganother.
- For example we can listen to an audio clip while scrolling a page and at the same time downloadanapplet from adistant computer.
- Thisfeaturegreatlyimprovestheinteractiveperformanceofgraphicalapplications.

Highperformance

- Javaperformanceisimpressiveforaninterpretedlanguage,mainlyduetotheuseofintermediatebyte code.Accordingtosun, javaspeed is comparableto thenativeC/C++.
- Javaarchitectureisalsodesignedtoreduceoverheadsduringruntime.Further,theincorporation ofmultithreadingenhances theoverallexecutionspeed ofjava programs.

DynamicandExtensible

- Javaisadynamiclanguage.
- It iscapableofdynamicallylinkinginnewclasslibraries, methods,andobjects.
- Javaprogramsupportfunctionswritteninotherlanguagessuchas C and C++. These functions are known as **nativemethods.**

Java differs from C and

C++Differencesbetweenjavaand

C:

- Javadoesnot includetheCstatementkeywords**sizeof**and**typedef.**
- Javadoesnotcontaindatatypes**struct** and**union.**
- Javadoesnotdefinethetypemodifierskeywords**auto,extern,register,signedandunsigned.**
- Javadoesnotsupport an explicitpointertype.
- Javadoesnothaveapreprocessor andwecannotuse**#define**,# **include**and**#ifdef** statements.
- Javarequiresthat thefunctionswithno argumentsmustbe declaredwithemptyparenthesisandnotwith the**void** keyword as done in C.
- Javaaddsnewoperatorssuchas**instanceof**and>>>.
- Javaaddslabeled**break** and**continue**statements.

DifferencesbetweenjavaandC++:

- Javadoesnotsupport operator overloading.
- Javadoes nothavetemplateclasses asinC++.
- Javadoesnotsupportmultipleinheritanceofclasses.Thiscanbeaccomplishedby anewfeaturecalledas **"interface"**.
- Javadoesnot supportglobal variables.
- Javadoes notusepointers.
- Javahasreplacedthedestructorfunctionwith afinalize()function.
- There are no headerfiles in java.

JAVAANDINTERNET

- JavaisstronglyassociatedwiththeinternetbecausethefirstapplicationprogramwritteninjavawasHot java.
- Itisawebbrowsertorunappletoninternet.
- Internetuserscanusejavatocreateappletprograms&runthemlocallyusinga"**javaenabledbrowser"**suchas Hotjava.
- Theabilityofjavaappletstohitcharideontheinformationsuperhighwayhasmadejavaa uniqueprogramminglanguagefortheinternet.
- Duetothis, javaispopularlyknownas**internetlanguage**.

JAVADEVELOPMENTENVIRONMENT

- Java environment includes a large number of development tools and hundreds of classes andmethods.
- The development tools are part of the system known as **Java Development Kit(JDK)** and theclassesandmethodsarepartofthe**JavaStandardLibrary(JSL),**alsoknownasthe**ApplicationProgramming Interface(API).**

JAVADEVELOPMENT KIT(JDK)

ϖJavaDevelopmentKitcomeswithacollectionoftoolsusedfordevelopingandrunningjavaprograms.Theyare

¬**Applet viewer**(forviewingjavaapplets) ☾ EnablesustorunJavaapplets.

¬**Javac**(javacompiler) ☾ IttranslatesJavasourcecodetobytecodefilesthattheinterpretercanunderstand.

¬**Java**(javainterpreter) ☾ Javainterpreter,whichrunsappletsandapplications byreadingand interpretingbytecodefiles.

¬**Javadoc**(forcreatingHTMLdocument) ☾ CreatesHTMLformatdocumentationfromsourcecodefiles.

¬**Javap**(javadissembler) ☾ Enablesusto convertbytecodefilesintoaprogramdescription.

¬**Javah**(forC header files) ☾ Producesheader files foruse withnativecodes.

¬**Jdb**(javadebugger) ☾ Ithelpsustofinderrorsinourprograms.

APPLICATIONPROGRAMMINGINTERFACE(API)

ϖThejavastandardlibrary(orAPI)includeshundredsofclassesandmethodsgroupedintoseveralfunctionalpackages. Mostcommonlyusedpackagesare

¬**Language support package**: -Acollection ofclasses and methods requiredfor implementingbasicfeaturesofjava.

¬**Utilities package**: - A collection of classes to provide utility functions such as date andtimefunctions.

¬**Input/outputpackages:**-AcollectionofclassesrequiredforInput/outputmanipulation.

¬**Networking packages:** - A collection of classes for communicating with other computersviainternet.

¬**AWTpackages**:-TheAbstractwindowToolKitpackagecontainsclassthatimplements platformindependent graphicalinterface.

¬**Appletpackage:**-Thisincludessetofclassesthatallowsustocreatejava applets.

JavaRuntimeEnvironment(JRE)

Itfacilitatestheexecution ofprogramsdevelopedinjava. Itcomprisesofthefollowing:

- **Java Virtual machine(JVM):**It is a program that interprets the intermediate java bytecode and generates the desired output. It is because if byte code and JVM concepts thatprogramswritten in Javaarehighlyportable.

- **Runtimeclasslibraries:** Thereareasetofcore classlibrariesthatarerequiredfortheexecutionofjavaprograms.
- **Userinterfacetoolkits:** AWTandswingareexamplesoftoolkitsthatsupportvariedinput methodsfortheuserstointeractwithapplicationprogram.
- **Deploymenttechnologies:** JREcomprisesthefollowingkeydeploymenttechnologies:
  - ¬**Javaplug-in:** Enablestheexecutionofajava appletonthebrowser.
  - ¬**JavaWebstart:** Enablesremote-deployment ofanapplication.

SIMPLE JAVAPROGRAM

Simplejavaprogram

```
classSampleOne
{
        publicstaticvoidmain (Stringargs[])
            {
                System.out.println(“JavaisbetterthanC++”);
            }
}
```

Classdeclaration

Thefirstline

ϖClassSampleOne declaresaclass,javaisatrueobject-orientedlanguageandtherefore,

everythingmustbeplacedinsideaclass.

ϖ**class**isa**keyword**anddeclaresthata newclassdefinitionfollows.

ϖSampleOne isajavaidentifierthatspecifiesthenameoftheclasstobedefined.

OpeningBrace

ϖEvery class definition in java begins with an opening brace **"{"** and ends with amatchingclosing brace**"}".**

Themainline

ϖThethirdline

publicstaticvoidmain(Stringargs[])

¬Theabove linedefinesamethodnamedmain.

¬This issimilar tothe**main()**functioninC/C++.

¬Every java application program must include the **main() method**. This the starting point fortheinterpreterto begintheexecutionoftheprogram.

¬Ajavaapplicationcanhaveanynumber ofclasses but**only one**ofthem mustincludea **main** methodtoinitiatetheexecution.

¬Thelinecontains anumberofkeywords**public,staticandvoid.**

¬***Public :***The keyword public is an access specifier that declares the main method asunprotectedandthereforemaking itaccessibletoallother classes.

¬***Static :*** *D*eclares this method as one that belongs to the entire class and not a part of any objectof the class. The main methods must always be declared as static since the interpreter uses thismethodbeforeanyobjectarecreated.

¬***Void:***Thevoidstatesthatthemainmethod doesnotreturn anyvalue.

Theoutputline

¬Theonlyexecutablestatementin theprogramis

System.out.println("JavaisbetterthanC++");

¬This issimilartoprintf() statementofCorcout<<constructofC++.

¬Sincejavaisatrueobject orientedlanguage,everymethod mustbepartofanobject.

¬The**println**methodisamemberofthe **out**object,whichisastaticdatamember of **System** class.

¬This lineprintsthestring"javaisbetterthanC++."

Anapplicationwithtwoclasses

```
classRoom
{
    floatlength;
    floatbreadth;
  void getdata(floata,float b)
    {
      length=a;
       breadth=b;
    }
}
```

```
ClassRoomArea
    {
publicstaticvoidmain(Stringargs[])
{
 float area;
 Roomroom1=newRoom();
 room1.getdata(14,10);
area=room1.length*room1.breadth;
 System.out.println("area="+area);
    }
}
```

Documentation

sectionPackagestateme

nt

Import

statementsInterfacestat

ements

Class

DefinitionsMainmethod

Class

{

MainmethodDefinition

JAVAPROGRAMSTRUCTURE

GENERALSTRUCTUREOFAJAVAPROGRAM

☾ Suggested

☾

O

p

t

i

o

n

a

}

lOptionalO

ptional

ESSENTIAL

DBC

Documentationsection:

- Thedocumentationsection comprisesaset ofcommentlines givingthenameoftheprogram,theauthorandother details.
- Commentsmustexplainwhyandwhatofclassesand howofalgorithms.
- Javaalsousesathird styleofcomment/**….*/knownasdocumentationcomment.Thisformofcommentisusedforg eneratingdocumentation automatically.

PackageStatement:

- Thisstatementdeclaresapackagename andinformsthecompilerthattheclassesdefineherebelong to thispackage. Example Packagestudent;

Thispackagestatementisoptional.

ImportStatements:

- Thisissimilartothe#includestatementinC.ExampleImport student. test;

InterfaceStatements:

- Aninterfaceis likeaclassbutincludesagroupofmethoddeclaration.
- Thisisalsoanoptionalsectionandis usedonlywhenwewishto implementthemultipleinheritancefeatures intheprogram.

ClassDefinitions:

- Ajavaprogram maycontain multipleclassdefinitions.
- Classesaretheprimary andessentialelementsof ajavaprogram.
- Theseclassesareusedtomap theobjectsof realworldproblem.

Mainmethodclass:

- Everyjavaprogramrequiresamainmethodasitsstartingpoint.
- Themainmethodcreates objectsofvariousclassesandestablishescommunication between them.
- On reaching the end of main, the program terminates and the control back to theoperatingsystem.

JAVATOKENS

ϖAjavaprogramisbasicallyacollectionofclasses.

ϖA class is defined by a set of declaration statements and methods containing executablestatements.

ϖMoststatementscontainexpression,whichdescribe the actionscarriedoutondata.

ϖ**Smallestindividualunitsinaprogramare knownastokens.**

ϖThe compilerrecognizesthemforbuilding upexpressionsandstatements.

ϖInsimpleterms,ajava programisacollectionof tokens,commentsandwhite spaces.

ϖJavalanguageincludes**fivetypesoftokens.**

Theyare

- ¬ Reservedkeywords
- ¬ Identifiers
- ¬ Literals
- ¬ Operators
- ¬ Separators

¬**JavaCharacterset**

¬Thesmallestunitsofjavalanguagearethecharactersusedtowritejavatokens.

¬These characters are defined by the Unicode character set, an emerging standardthattriestocreatecharactersforalargenumberof scriptsworldwide.

ϖ**Reservedkeywords**

¬Javalanguagehas reserved60 wordsas keywords.

¬These keywords combined with operators and separators according to syntax,form definitionofthejavalanguage.

¬Sincekeywordshave specificmeaningin java,wecannotusethem asnamefor variables,classes,methods andsoon.Allkeywordsaretobewritten inlowercaseletters.Someexamples arebyte,class,

¬ do,extends,for,importetc.

DBC

ϖIdentifiers

¬They are used for naming classes, methods, variables, objects, labels, packagesandinterfaces inaprogram.

¬Rulesfornaming anidentifier

⎪Theycanbeofanylength.

⎪Uppercaseandlowercaseletteraredistinct.

¬Literals

ϖOperators

⎪Theycanhavealphabets,digits,underscoreanddollarsigncharacters.

⎪Theymustnotbeginwithdigit.

Egaverage, sum

¬Literals in java are a sequence of characters that represent constant values to bestoredin variables.

¬Javalanguagespecifiesfivemajortypesofliterals.Theyare

⎪Integerliterals

⎪Floating-pointliterals

⎪Characterliterals

⎪Stringliterals

⎪Booleanliterals

¬An operator is a symbol that takes one or more arguments and operators on themtoproducearesult.

ϖ**Separators**

¬Separators are symbols used to indicate where groups of code aredividedand arranged.

() ☾ **parantheses,{}** ☾ **braces, []** ☾ **brackets,;** ☾ **semicolon,,** ☾ **comma,.** ☾ **period**

.

JAVASTATE MENTS

ϖJavastatementsarelikenatural**language**s.

ϖAstatementisanexecutablecombinationoftokensendingwitha**semicolon(;)** mark.

ϖStatementsareexecutedinsequenceinthe orderinwhichtheyappear.

Javaimplementsseveraltypesofstatements,theyare

ϖ***Empty statement*** ☾ These do nothing and are used during program development as a placeholder.

ϖ***Labeled statement*** ☾ Any statement begin with a label, such labels must not bekeywords, already declared local variables, or previously used labels in this module.Labelsin javaareused astheirarguments ofjumpstatement.

ϖ***Expressionstatements*** ☾ JavahasseventypesofExpressionstatements.Assignment,pre-increment, pre-decrement, post-increment, post-decrement, method call andAllocationExpression.

ϖ***Selection statement*** ☾ These select one of the several control flows. They are**if, if-else andswitch**

ϖ***Iterationstatement*** ☾ Thesespecifyhowandwhenloopingwilltake place.Theyare**while, doandfor.**

ϖ***Jump statement*** ☾ Jump statement pass control to the beginning or end of the current block or toa labeledstatement.Theyare**break,continue,returnandthrow.**

ϖ**Synchronizationstatement** ☾ Theseare usedforhandlingissueswithmulti-threading.

ϖ**Guardingstatement** ☾ Usedfor safehandlingofcodethatmay causeexception.

Thesestatementsusethekeywords **try,catch,andfinally**.

JAVAVIRTUALMACHINE

ϖAlllanguagecompilerstranslatesourcecodeintomachinecode.
ϖJava compilerproducesan intermediate codeknown as**bytecode**for amachine that doesnot exist.
ϖThismachineiscalledthe**Java VirtualMachine**.
ϖTheprocessofcompilingajava programintobytecode isreferredtoas**virtualmachinecode**.

SOURCECODE **BYTE**

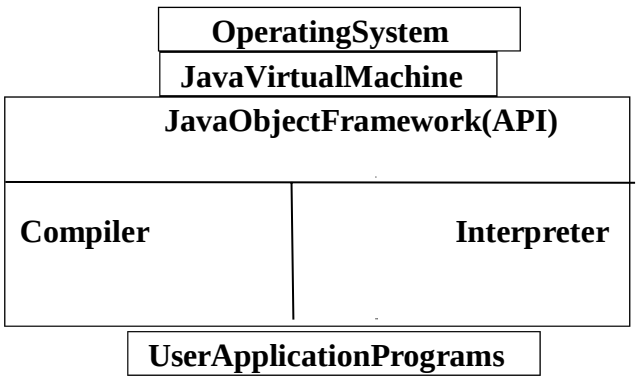
CODEProcessofCompilation

ϖThevirtualmachinecodeisnotmachinespecific.
ϖThe machine specific code (machine code) is generated by the java interpreter by actingasaintermediarybetween thevirtual machineandtherealmachine.

VIRTUALMACHINE **REALMACHINE**

Processofconvertingbytecodeintomachinecode

Thejavaobject framework(JavaAPI)actsastheintermediarybetweentheuserprogramsandthevirtual machine which in turn acts as the intermediary between the operating system and the javaobjectframework



LayersofinteractionsforJavaprograms

COMMANDLINEARGUMENTS

Commandlineargumentsareparameters thataresuppliedtotheapplicationprogram atthetimeofinvoking it forexecution.

```
ClassComLineTest
{
    Publicstaticvoidmain(Stringargs[ ])
   {
          int count,i=
          0;Stringstring;
          count=args.length;
          System.out.println("Numberofarguments="+count);whi
          le(i<count)
          {
                string=args[i];i
                =i+1;
                System.out.println(i+":"+        "Javais"+string+"!");
           }
     }
}
```

Compileandruntheprogramwiththecommandlineasfollows:

Java**ComLineTest**SimpleObject_OrientedDistributedRobustSecure PortableMultithreadedDynamic

Duringtheexecution,the commandlineargumentsSimple,Object_Oriented,etc. arepassedto theprogram through the array **args.** That is the element **args[0]** contains Simple,**args[1]** containsObject_Oriented ,andsoon.Theseelementsareaccessed usingtheloopvariable Iasan indexlike

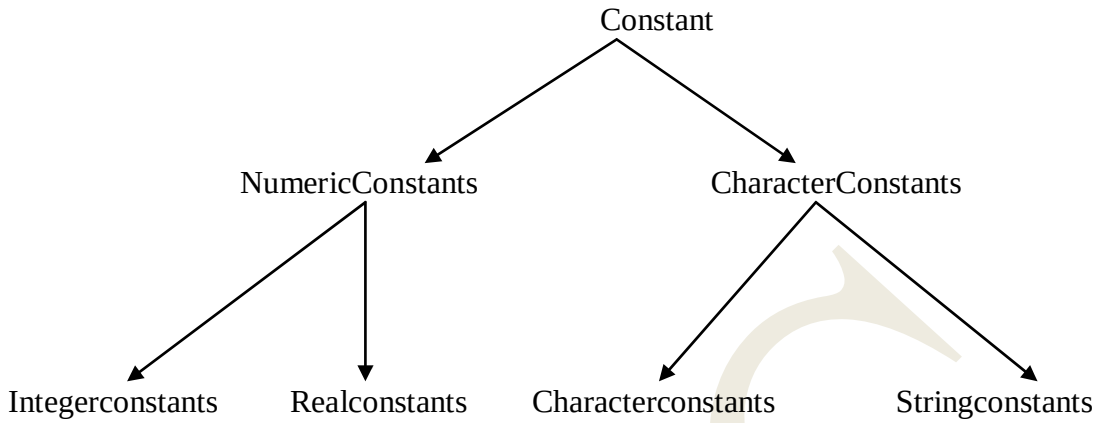
```
name=args[i]
```

Theindexi isincrementedusingawhileloopuntilalltheargumentsare accessed.Thenumberifargumentsis obtained bystatement

```
count=args.length;
```

CONSTANTS

Constantsreferstofixedvalues thatdonotchange duringtheexecutionof a program.



¬**IntegerConstants**

Itrefersto asequenceofdigits.Thereare3typesofintegers,namely

- Decimalinteger
- Octalinteger
- Hexadecimalinteger

⎫ Decimalintegersconsistofasetofdigits,0through9,precededbyanoptional minussign.

⎫ Embeddedspaces,commas,andnon-digitcharactersarenotpermittedbetweendigits.

Valid	Invalid
1234	70.00
-789	$234

⎫ Anoctalintegerconstantconsistsofanycombinationofdigitsfromset0through7,witha leading0.

Examples

0345

0556

✓ Asequenceofdigitsprecededby0x or0Xisconsideredashexadecimalinteger.
✓ TheyincludealphabetsAthrough For athroughf.

- ✓ AletterAthroughFrepresentsthenumbers10through15.Followingaretheexamplesofvalidhexadecimali ntegers.

  0X2 0X9F 0xbcd 0x

¬**RealConstants**

⎪Numberscontaining fractionalpartsarecalledrealorfloating pointconstants.Examplesare0.087

-0.98 456.78

⎪Thesenumbersareshownindecimalnotation,havingawholenumberfollowedby decimalpointandthefractionalpart,which isaninteger.Thatis

213. .98 -.71areall validreal numbers.

⎪A real number may also be expressed in exponential or scientific notation.For example, thevalue215.45 maybewrittenas 2.1545e2in exponentialnotation. Thegeneral form is

⎪Themantissaiseitherarealnumberexpressedindecimalnotationoraninteger.

⎪ Theexponentisan Integerwithan optionalplusorminussign.

⎪Thelettereseparatingthemantissaandtheexponentcanbewrittenineitherlowercaseoruppercase.Examples oflegal floatingpointconstants are

0.65e4 12e-2 1.5e+5

Afloating pointconstantmaycomprise fourparts:

- awholenumber
- adecimalpoint
- afractionalpart
- anexponent

¬**Singlecharacterconstants**

⎪Asinglecharacterconstantcontainsasinglecharacterenclosedwithinapairofsinglequotemarks.

Examplesofcharacterconstants are:

‘5’ ‘X’ ‘:’ ‘‘

¬**Stringconstants**

⎪Astringconstantissequenceofcharactersenclosedbetweendoublequotes.

⎪Thecharactersmaybealphabets,digits,specialcharactersandblankspaces.Examplesare:

“Hello” “1997”

¬**Backslashcharacterconstants**

- Javasupportssomespecial backslashcharacter constantsthatareusedinoutputmethods.
- For example,the symbol ‘\n’stands a newlinecharacter.
- Theyconsistoftwocharacters.Thesecharacterscombinationsareknownas**escapesequences.**

Constant	Meaning
‘\b’	Backspace
‘\f’	Formfeed
‘\n’	Newline
‘\r’	Carriagereturn
‘\t’	Horizontaltab
‘\”	Singlequote
‘\”’	Doublequote
‘\\’	Backslash

DATATYPES

Datatypesspecifythesizeand typeof valuesthatcan be stored.

¬**IntegerTypes**

⎪Integertypescanholdwholenumberssuchas123,-96,5678.Javasupportsfourtypesofintegers.

Theyare**byte,short, int, andlong.**

Type	Size	Minimumvalue	Maximumvalue
Byte	Onebyte	-128	127
Short	Twobytes	-32,768	32,767
Int	Fourbytes	-2,147,483,648	2,147,483,647
Long	Eightbytes	-9,223,372,036,854,775,808	9,223,372,036,854,775,807

¬**FloatingPointTypes**

⎪Floatingpointtypecontainsfractionalpartssuchas26.78and-7.890.

⎪The**float**typevaluesaresingle-precisionnumberswhilethe**double**typesrepresentdoubleprecisionnumbers.

⎪Floatingpointnumbersaretreatedasdouble-precisionquantities.WemustappendforFtothenumbers.

Example:

```
1.
2
3
f7.6756
7
e5
```

⎪Double-precisiontypesareusedwhenweneedgreaterprecisioninstorageoffloatingpointnumbers.

Floatingpointdatatypessupportaspecialvalueknownas Not-a-Number(NaN).

⎫ Itisusedtorepresenttheresultofoperationssuchasdividingbyzero,whereanactualnumberisnotproduced.

Type	Size	Minimumvalue	Maximum value
Float	4bytes	3.4e-038	1.7e+0.38
double	8bytes	3.4e-038	1.7e+308

¬**CharacterType**

⎫ Javaprovidesacharacterdatatypecalled**char**.

⎫ Thechartypeassumesasizeof2bytesbut,basically,itcanholdonlyasinglecharacter.

¬**BooleanType**

⎫ Itisusedtotestaparticularconditionduringtheexecutionoftheprogram.

⎫ Thereareonlytwovaluesthatabooleantypecantake:trueorfalse.

⎫ Booleantypeisdenotedby thekeywordbooleanandusesonlyonebitofstorage.

VARIABLES

ϖAvariableisanidentifierthatdenotesastoragelocationusedtostore a datavalue.

ϖ Variable namesmay consist of alphabets, digits,the underscore(_)anddollarcharacters,

subjecttothefollowingconditions:

1. Theymustnotbeginwithadigit.
2. Uppercaseandlowercase aredistinct.
3. Itshould notbeakeyword.
4. Whitespaceisnot allowed.
5. Variablenamescanbeofanylength.

¬**Declarationof variables**

The declarationstatementdefinesthe typeof variable.Thegeneralformof declarationofavariableis:

Variablesareseparatedbycommas.Adeclarationstatementmustendwithsemicolon.Somevaliddeclarations are:

```
intc
ount;
float   x,y;
```

¬**Givingvaluestovariables**

Avariablemustbegivenavalueafterithasbeendeclareditisusedinanexpression.Thiscanbe achieved intwoways:

1. Byusingan assignmentstatement
2. Byusingareadst atement

¬**AssignmentStatement**

Asimple methodof givingvalue to avariableis through theassignment statement asfollows:

Forexample:

initialvalue=0;

finalvalue =123;

Wecanalso

stringassignmentexpressionasshownbelow:x=y=z=0;

Itisalsopossibletoassignavaluetoavariableatthetimeofitsdeclaration.Thegeneralformisas follows:

Examples:

int finalvalue =

123;

char yes =

'x';

¬**ReadStatement**

⎪Wemayalsogivevaluestovariablesinteractivelythroughthekeywordusingthe**readline()** method.

⎪The**readline()**methodreadstheinputfromthekeyboardasastringwhichisthenconvertedtothe correspondingdatatype usingthedatatypewrapperclasses.

¬**Scopeof variables**

Javavariablesareactuallyclassifiedintothreetypes:

- Instancevariables
- Classvariables
- Localvariables
- Instanceandclassvariableredeclaredinsideaclass.Instancevariablesarecreatedwhen theobjectsareinstantiatedandtheyare associatedwiththeobjects.
- Classvariablesareglobaltoaclassandbelongtotheentiresetofobjectsthatclasscreates. Onlyonememorylocationis createdforeach classvariable.
- Variablesdeclaredandusedinsidemethodsarecalledlocalvariables.Theyarenotavailable foruseoutsidemethoddefinition.

¬**Typecasting**

⎪Weoftenencountersituationswherethereisaneedtostoreavalueononetypeintoavariableof anothertype.

⎪Insuchsituation,wemustcastthevaluetobestoredbyproceedingitwiththetypenamein parentheses.Thesyntaxis

Theprocess ofconverting onedatatypeto another is called**casting.** Examples:intm=50;

byten=(byte)m;

Fourintegertypescanbecasttoanyothertypeexcept Boolean.Castingintoasmallertypemayresultinlossofdata.Similarly,thefloatanddoublecanbe casttoanyothertype exceptBoolean.

From	To
bytes	short,char,int,
hortc	long,float,doubleint,long,
harin	float,double
tlong	int, long, float,
float	doublelong,float,doub
	lefloat,double
	Double

Castsresultsinnolossofinformation

¬**Gettingvaluesof variables**

Javasupportstwooutputmethodsthatcanbeused to sendtheresults tothescreen.

- print()method//print andwait
- println()method//print alineandmovethenext line

⎫ Theprint()methodprintsoutputononelineuntilanewlinecharacterisencountered.

For example, the statementsSystem.out.print ("Hello");System.out.print("Java!");

willdisplaythewordsHelloJava!Ononelineandwaitsfordisplayingfurtherinformationon thesameline.

⎫ Theprintln()method,bycontrast,takestheinformationprovidedanddisplaysitona line followed by a line feed.For exampleSystem.out.println("Hello");System.out.println ("Java!");

will produce the following output:Hello Java!

OPERATORSANDEXPRESSIONS

Introduction:

ϖJavasupportsarichsetofoperators.

ϖAnoperatorisasymbolthat isusedformanipulatedataandvariables.

ϖOperatorsareusedinprogramstomanipulatedataandvariables.

Javaoperatorsareclassifiedintonumberofcategories.

¬Arithmeticoperators

¬Relationaloperators

¬Logicaloperators

¬Assignmentoperators

¬Incrementanddecrementoperators

¬Conditionaloperators

¬ Bitwiseoperators

¬Special operators

¬ARITHMETICOPERATORS

ϖArithmetic operatorsare used toconstructmathematicalexpressionsasinalgebra

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulodivision

Ex: a-b a+b a*b a/b a%b

ϖHereaandbmaybevariableorconstants. Theyarealsoknownasoperands

ϖ-,+,*,/,%arecalledoperators.

i) Integerarithmetic

ϖWhen both the operands in a single arithmetic expression such as a + b are integer, theexpression is called an "integer expression" and the operation is called as "integerarithmetic".

ϖIntegerarithmeticalwaysgivesanintegervalue.

E.g.:a=14,b=4

a+b=18

a-

b=10a*

b=56

a/b=3(decimal part

truncated)a%b=2(reminderofinteger

division)

¬Formodulodivision,thesignoftheresultisalwaysthesignofthefirstoperand.

-14%3=-2 14%-3 =2

ii) Realarithmetic

ϖAnarithmeticoperationinvolvingonlyrealoperandiscalled“**realarithmetic**”.

ϖArealoperandmayassumevalueseitherindecimalorexponentialnotation.

Sampleprogram:

```
import java.io.
*;class floatpoint
{
    publicstaticvoidmain(Stringargs[])
        {
        floata=20.5,b=6.4;
        System
        .out.println("a="+a);System.out.pr
        intln("b="+b
        );System.out.println("a+b="+(a-
        b));
        System.out.println("a-b="+( a-b
        ));System.out.println("a*b="+(
        a*b));System.out.println("a/b="+(
        a/b);System.out.println("a%b="+(a
        %b));
        }
 }
```

Mixedmodearithmetic

ϖWhen one of the operands is real and the other is an integer, the expression is called amixed-modeexpression.

ϖIfeitheroperandisof the realtype,thenthe otheroperandisconvertedtorealandthe realarithmeticisperformed.

E.g.: 15/10.0=1.5

15/10=1

DBC

¬RELATIONALOPERATORS

ϖ

Integer | Typevaria | **Highest** | ae-lrelation**i**aloperatorae-2

point | ( ) | [] | .

++ | — | *

* | / | %

+ | –

>> | >>> | <<

> | >= | <

== | !=

&**l**

^

|&

&

X=(a>b)

TheoutputisX=15

Operato r	**Meaningis e**
!	isless than
<=	is less than or equalto
>	isgreaterthan
>=<=	isgreaterthanore qual to
==	isequal to
!=	isnotequalto

Eg:Expression **value**

4.5<=10 True

4.5>=10 False

¬LOGICALOPERATORS

ϖJavahasthreelogicaloperators.

Operator	**Meaning**
&&	LogicalAND
||	LogicalOR
!	LogicalNOT

ϖAlogicaloperatorreturnseither**TRUE**or**FALSE**values.

TruthTable

op-1	**op-2**	**op-1&&op-2**	**op-1||op-2**
True	True	True	True

True	False	False	True
False	True	False	True
False	False	False	false

ϖLogical operator **&&**and || are used to check compound condition (**ie** for combining two or morerelations)

ϖWhen an expression combines two or more relational expressions then it is called logical expressionoracompound relational expression

Eg: if (age>55 &&
salary<1000)if(mark1>40&&
mark2>40)

¬**ASSIGNMENTOPERATORS**

ϖUsedtoassignthevalueofan**expression**toa**variable**.

ϖAssignmentoperatorsareusuallyintheform**"=".**

ϖShorthandform:

vop=exp;

v ☾ variable

exp ☾ expression

op ☾ javabinaryoperator

ϖTheassignmentstatement**vop=exp;**isequivalentto**v=vop(exp);**

Statementwithsimpleassignmentoperator	**Shorthandoperator**
a=a+1	a+=1
a=a-1	a-=1
a=a*(n+1)	a*=n+1
a=a/(n+1)	a/=n+1

A=a%b	a%=b

Eg: Z+=Y+1whichisequaltoZ=Z+(Y+1)

Advantages:

It has3advantages.

- Easyto write.
- Easyto read
- Efficientcode.

INCREMENTANDDECREMENTOPERATORS

Theyarealsocalledunaryoperator.

++ Incrementoperator,add1to theoperand

-- Decrementoperator,subtract1to theoperand

Theymayalsousedtoincrementsubscriptedvariables

Eg. a[i++]

SampleProgram:

```
classIncrementoperator
{
        publicstaticvoidmain(Stringargs[ ])
                {
                        intm=10,n=20;
                        System.out.println("m="+m);
                        System.out.println("n="+n);
                        System.out.println("++m="+++m);
                        System.out.println("n++="+n++);Sy
                        stern.out.
                        println("m="+m);System.out.println
                        ("n="+n);
                }
}
```

Output

m=10

n=20

++m=11

n++=21

m=11n=

21

¬**CONDITIONALOPERATORS**

ϖThe characterpair**?:**isusedforconditionaloperator.
ϖItisalsocalledas**ternary**operator.

ϖexp1,exp2,exp3areexpressions
ϖTheoperator?:worksasfollows

¬**Expression1**isevaluatedfirst, ifitsis **true** thenthe**expression2** isevaluated.

¬Ifexpression1is**false**,**expression3** isevaluated.

¬**BITWISEOPERATORS:**

ϖBitwiseoperatorsareusedtomanipulatedataatvaluesofbitlevel.
ϖThese operatorsareusedfor testingthebits,orshiftingthemtotherightor left.
ϖBitwiseoperatorsmaynottofloatordouble.

Operator	Meaning
&	BitwiseAND
!	BitwiseOR
^	BitwiseexclusiveOR
~	One'scomplement

<<	Shiftleft
>>	Shiftright
>>>	Shiftrightwith zerofill

Eg:5 = 0000

0101A=5

>>A // 1000 0010

<<A // 0000 1010

¬**SPECIALOPERATORS**

ϖJavasupportsspecialoperators

¬Instanceofoperator

¬Dotoperator(or)member selectionoperator (.)

⎪***Instanceofoperator:***

ϖInstanceofoperatorisanobjectreferenceoperator.

ϖAllowustodeterminewhetherthe objectbelongstoaparticularclassornot.

ϖReturn true, if the object on the left-hand side is an instance of the class given on the right-handside.

E.g.person**instanceof**student

¬Is **true**iftheobjectpersonbelongstoclass**student**;otherwiseitis**false**.

⎪***Dotoperator***

ϖThedotoperator**(.)** isusedtoaccesstheinstance variablesandmethodsofclassobjects.

Person1.age //**referencetothevariableage**

Person1.salary() //**referencetothemethodsalary()**

It isalsousedtoaccessclassesandsubpackagesfromapackage.

¬**ArithmeticExpressions**

- Anarithmeticexpressionisacombinationofvariablesconstantsandoperatorsarrangedasperthe syntaxofthelanguage.
- Javacanhandledanycomplex mathematicalexpressions.
- Java does not have an operator for exponentiation. Some of the expressions areAlgebraicexp: Javaexp:

ab-c a*b-c

(m+n)(x+y)	(m+n)*(x+y)
ab/c	a*b/c
3x2+2x+1	3*x*x+2*x+1

¬E**valuationofExpression:**

- Expressionsare evaluatedusingan assignmentstatementoftheform

Variable=ex pression;

- VariableisanyvalidJavavariablename.Whenthestatementisencountered,the expression is evaluated first and the result then replacesthe previousvalueofthevariableontheleft handside.

Eg:x=a*b-c;

y= a-b/c+d;

¬**PrecedenceofArithmeticoperators:**

- Anarithmeticexpressionwithoutanyparentheseswillbeevaluatedfromlefttoright usingtherules ofprecedenceofoperators.
- Thetwodistinctprioritylevelsofarithmeticoperatorsin Javaare,

1. Highpriority
   */%
2. Lowpriority
   + -

¬**Type conversions**

inExpressions:AutomaticTyp

e

conversion:

⎪ Iftheoperandsareofdifferenttypes,thelowertypeisautomaticallyconvertedtothehigher typebeforetheoperationproceeds.
⎪ Theresultisofthehighertype.
⎪ If**byte,shortandint**variablesareusedinexpression,theresultisalwayspromoted to**int,**toavoid overflow.
⎪ Ifasingle**long**isusedinexpression,thewholeexpressionispromotedto**long.**

Casting avalue:

⎪ Toconvertanobjectorvariableofonetypetoanotheriscalledcastingavalue.
⎪ Thegeneralformofacastis

(type_name)expression

Example

X=(int) 7.5 ☾ 7.5isconverted tointegerbytruncation.

OperatorPrecedenceandAssociativity:

The Precedenceof the JavaOperators

QUESTIONS

2Marks

1. Whatisthedifferencebetween CandC++?
2. ExpandJDKandSDK.
3. DefineJVM.
4. DefineTokens.
5. Whatdoyoumeanbyescape sequences?
6. Definecastingavalue.
7. Whatisthedifferencebetween print()and println( )method?

5Marks

1. Explainindetailaboutfeaturesofjava.
2. DescribeaboutJavaprogramstructure.
3. ExplainindetailaboutJavastatements.
4. ExplainCommandlineargumentswithsuitableexample.
5. Whatarethetypesof constants?Explainindetail.
6. WriteshortnotesonDatatypesinjava.
7. Explainindetailaboutvariables.

10Marks

1. ExplainindetailaboutOperatorsandExpressionsinjava.

UnitIcompleted

DBC

UNITII

DECISIONMAKINGANDBRANCHINGI

NTRODUCTION

ϖ When a program breaks the sequential flow and jumps to another part of the code,itis called **branching.**

ϖ Whenthebranchingisbasedonaparticularcondition,itisknownas**conditional branching**.

ϖ If branching takes place without any decision, it is known as **unconditionalbranching**.

Thefollowingstatementsareknownascontrolordecisionmakingstatements.

–**if**statement

–**switch**statement

–Conditionaloperatorstatement

DECISIONMAKINGWITHIFSTATEMENT

ϖThe if statement is a powerful decision making statement and is used to **control the flow ofexecutionofstatements**.

Generalform

if(testexpression)

ϖTheexpressionisfirstevaluated.

ϖDepending on the value of the expression is true or false, control is transfer to a particularstatement.

ϖTheifstatementare

1. simple**if**statement
2. **if…else**statement
3. Nested**if…else**statement
4. **elseif**ladder

1. SimpleIfStatement

–If the test expression is **true** the **statement block will be executed**; otherwise theexecutionwill **jump to the statement-x**

¬Statementblockmaybesinglestatementora groupofstatement.

Generalform

```
if(testexpression)
{
   statement-block;
}
statement-x;
```

Example

```
if(category==SPORTS)
{
   marks =marks +bonus_marks;
}
System.out.println(marks);
```

2. TheIf…ElseStatement

¬Ifthetestexpressionis **true**,then the**true-blockstatementsareexecuted**.

¬Otherwise,**thefalseblockstatementsareexecuted**.

Generalform

```
if(testexpression)
{
Trueblockstatements;
}
else
{
Falseblockstatements;
}
Statement-X;
```

Example

```
if(degree==“BCA”)
{
points=
points+500;System.out.println(“It
sBCA”);
}
else
{
points=
points+200;System.out.println(“I
tsBSC
CS”);
….
```

3. Nestingofif …elsestatement

¬Hereifthe**condition-1**is**false**,the**statement-3**willbeexecuted;**otherwise**it**evaluates**

the**condition-2**.

¬Ifthe**condition-2**is**true**,then**statement-1**willbeexecuted;**otherwise**the**statement-2**

Genera$_i$l$_{ll}$Form i **tr**

```
if(testcondition1)
        if(testcondition2)
                {
                Trueblockstatements-1;
                }
        else
                {
                Falseblockstatement-2;
                }
        else
                {
                Falseblockstatements-3;
                }
Statement-x;
```

Examp$_t$le$_t$.

```
if(gender==“female”)
        if(balance>5000)
          {
                Bonus=0.03*balance;}
        else
          {
                Bonus=0.02*balance;
          }
 else

          {
                Bonus=0.01*balance;
          }
balance=balance+bonus;
```

4. Theelseifladder

¬Elseifladderisachain ofifsinwhichthestatementassociated witheachelseisanif.

¬Thecondition isevaluated fromthetoptodownwards.

¬As soon as the **condition is true**, then **the statements associated with it are executed** andthe**control is transferredto thestatement-x.**

¬When allthe**nconditionisfalse**,thenthe**finalelsecontaining**the**default-statement** willbeexecuted.

Generalform

```
If(condition-1)
        statement-1;
      elseif(condition-
          2)statement-2;
      elseif(condition-
          3)statement-3;
          …..
          …..
       else if (condition
          n)statement-n;
  else
       default-
  statement;statement-x;
```

Example

```
If(marks>79)
        grade=”honors”;
      elseif(marks>79)
        grade=”first”;els
      eif(marks>79)
        grade=”second”;
      elseif(marks>79)
        grade=”third”;
  else
      grade=”fail”;  // Default-
    stmtSystem.out.println(“grade=”+gr
    ade);
```

TheSwitchStatement

¬It an **multiway**decision statement.
¬Theswitchstatementteststhevalueofa givenvariable againstalistof**case**values.
¬Whena **match isfound**,ablockofstatementassociatedwiththat**case**isexecuted.
¬The expressionisanintegerexpressionorcharacterknownas**caselabels**.
¬Block1,block2…arestatementslistsmaycontainzero ormorestatements.
¬Noneed toput**braces**around**each block**
¬Caselabelsendwitha**colon(:)**
¬The **break statement** at the end of each block signal the **end of a particular case** andcauses an **exit** from the **switch statement**, transferring the control to the statement -xfollowingtheswitch.

¬The**default**isan**optioncase**;itwillbe**executed** ifthevalueofthe**expressiondoes notmatch**withanyofthe**casevalues.**

¬Ifnotpresent,noactiontakes place **whenallmatchesfail**andthecontrolgoestothe **statement–x.**

Generalform

```
switch(expression)
  {
     casevalue-1:
              block-1
              break;
     casevalue-2:
              block-2
              break;
     casevalue-3:
              block-3
              break;
      ……..
      …….
   default:
              default-
              blockbreak;
    }
statement-x;
```

Example1

```
switch(expression)
{
   case‘1’:
            System.out.println(“Monday”);
            break;
   case‘2’:
             System.out.println(“Tuesday”);
            break;
  case‘3’:
            System.out.println(“Wednesday”);
            break;
   case‘4’:
            System.out.println(“Thursday”);
            break;
          ……
          ……..

 default:
             System.out.println(“WRONGINPUT”);
            break;
}
System.out.println(“WELCOMETOTHISWEEK”);
…..
```

?:

Itisa**two-waydecision**makingoperator.

This operator is a combination of **? and :** and takes three operands.This operatorispopularlyknown astheconditionaloperator.

```
Conditionalexpression?Expression1:expression2;
```

```
If(x<0)
    Flag=0;
else
    Flag=1;
```

Ex1:Canbewrittenas

```
Flag=(x<0)?0 :1;
```

Ex2:

```
Y = (x>2)
?(2*x+1):(3*x+2);
```

DECISIONMAKINGANDLOOPINGI

NTRODUCTION

ϖTheprocessof**repeatedlyexecutingablockofstatement**isknownas**looping**.

ϖThestatementsintheblockmaybeexecuted**anynumberoftimes**,from**zero**to**infinite** number.

ϖIfaloopcontinuesforever,itiscalledan**infiniteloop.**

ϖAprogramloopconsist**oftwosegments**,oneknownasthe **bodyofthe loop**andotherknownasthe**control statements.**

ϖThe controlstatementstestcertainconditionsandthendirecttherepeatedexecutionofthe statements containedin thebodyof theloop.

ϖAloopingprocessingeneralconsistsoffollowingsteps:

- ¬Settingandinitializationofacounter
- ¬Executionofthestatementsin theloop
- ¬Testforaspecified conditionforexecutionoftheloop
- ¬Incrementingthecounter

ϖJavalanguageprovidesthreeloopingstatementsforloopoperations.

- ¬The **while**statement
- ¬The **do**statement
- ¬The **for**statement

THEWHILESTATEMENT

ϖThe**simplest**ofalltheloopingstructuresinjava isthe **whilestatement.**

ϖThe whileisan**entry-controlledloop**statement.

ϖThe**testconditionisevaluated**andif theconditionis**true**,thenthe **bodyoftheloop**is **executed**.

ϖAfterexecutionof the body,the **testconditionisonceagainevaluated**andif it is**true,**the **bodyisexecuted onceagain**.

ϖThe **execution of the body continues**, **until the test condition** becomes **false** and the control istransfer **out oftheloop**.

ϖOnexit,theprogramcontinueswiththestatementimmediatelyafterthebodyoftheloop.

Generalform

Initialization;

while(testcondition)
{
Bodyoftheloop
}

Example

```
Sum=0;
N=1;
while(N<=10)
        {
        sum=sum +
        N;N=N+1;
        }
System.out.println("sum""+sum);
```

Herethebodyoftheloopisexecuted10timesforn=1,2..10.

THEDOSTATEMENT

ϖIndostatement,the programproceedsto**evaluatethebodyoftheloopfirst**.

ϖAtthe**endofthe loop**,the **testcondition**inthe **whilestatementisevaluated**.

ϖIf theconditionis**true,**theprogramcontinuesto**evaluate thebodyoftheloop**once again.

ϖThe**programcontinues**toevaluatethe bodyofthe loopaslongas**theconditionistrue**.

ϖWhen the condition becomes **false**, **the loop will be terminated** and the control goes to thestatementthat appears immediatelyafterthewhilestatement.

General form

```
Initializa
tion;do
{
Bodyoftheloop
}
while(testcondition);
```

Example

```
Sum=0;
N=1;
        do
        {
        sum=sum +
        N;N=N+1;
        }
        while(N<=10);System.out.println("sum""+sum);
```

Herethebodyoftheloopisexecuted10timesforn=1,2..10.

THEFORSTATEMENT

ϖForloopisanentry-controlledloop.
ϖTheexecutionoftheforstatementisasfollows:

(1) **Initialization**ofthecontrolvariablesisdone first,usingassignmentstatementssuch asi=1andcount=0.Thevariablei and count areknownas loop-controlvariables.

(2) Thevalueofthe**controlvariablesistested**usingthetestcondition.Thetestconditionisarelational expression, suchas**i<10** thatdetermineswhentheloop willexit.

¬It the condition is **true**, the **body** of the loop is **executed**; otherwise the loop is**terminated** and the execution continues with the statement that immediately followstheloop.

(3) When the body of the **loop is executed**, the control is transferred back to thefor statementafterevaluating thelast statement in theloop.

ϖNow thecontrolvariable isincrementedusinganassignmentstatementsuch asi=i+1and thenewvalueofthecontrolvariableisagaintestedtoseewhetheritsatisfiestheloopcondition.

ϖ Ifthe conditionissatisfied,thebodyoftheloopis againexecuted.

ϖThis process continues till the value of the control variable fails to satisfy the testcondition.

Generalform

```
for(initialization;testcondition;increment)
{
  Bodyoftheloop
}
```

Simpleexample-2

```
for(n=1;n<=10;n++)
{
   sum=sum+n;
}
```

Simpleexample-1

```
for(i=0;i<=10;i++)
{
   System.out.println(i);
}
```

Additionfeaturesofforloop

```
p=1;
for(n=0;n<17;++n)
```

valid

```
m=5;
for(;m!=100;)
{
   System.out.println(m);
   m=m+5;
}
```

Canbewrittenas

¬for(p=1,n=0; n<17;++n) ☾ valid

¬for(n=1,m=50;n<=m;n=n+1,m=m-1) ☾ valid

¬for(i=1;i<20 &&sum<100;++i) ☾ valid

¬for(m=5;m!=100;m=m+5) ☾ valid

Nestingofforloops

¬A**for**loopwhichispresentinsideofanother**for**loopiscalled nestingof**forloop.**

```
for(i=1;i<=10;i++)
   {
    …….
    …….
        for(j=1;j!=5;++j)
            {…….
             ……...
             }
            ……
  }
    ……...
```

```
for(i=1;i<=10;i++)
{
      for(j=1;j<=10;j++)
          {
              System.out.println(i,j);
          }
}
```

JUMPSINLOOPS

ϖLoops perform a set of operations repeatedly until the control variable fails to satisfy the testcondition.

ϖJava permitsa jumpfromone statementtotheendor beginningofa loopaswellasa jumpoutof aloop.

Jumpingoutofa loop

ϖExitfromaloopcanbeaccomplishedbyusingthe**breakstatement**.

ϖ**Break**canalsobeusedwithin**while,doand for**loops.

ϖWhen break statement is encountered inside a loop, the loop is immediately exited and theprogramcontinues withthestatement immediatelyfollowing theloop.

ϖIn nestedloop,the**break wouldonlyexit fromthe loopcontainingit.**

ϖThebreakwillexitonlyasingleloop.

```
while(…….)

{
    ……
    …….
   if
   (condition)br
   eak;
   ……
   …..
  }.....
```

```
do(……)
{
    …..
    …..
     if
    (condition)br
    eak;
    …..
    …..
}while (….)
   …….
```

```
for(……)
{
    …..
    ……
for(…….)
   {
    ….
    …..
  if
   (condition)b
   reak;
   ……..
   …….
    }// forsecond for
   …..
   …..
}//forfirstforit
```

ϖContinuestatementskips**apar** **.**

ti

ϖThe continue statement; cause the loop to be continued with the nestatementin between.

ϖThe continue statement tells the compiler **"skip the following statements and continue withthenextiteration".**

EXAMPLE

```
while(testcondition)
{
    ……
    if(condition)
    continue;
     …….
     …….
}
```

EXAMPLE

```
do(…….)
{
          ……
           if(condition)
           continue;
           …….
  }while(testcondition);
```

EXAMPLE

```
for(…….)
{
          ……
           if(condition)
           continue;
           …….
          …….
}
```

LABELLEDLOOPS

ϖAlabelisany**validjavavariablename**.

ϖTogive a labeltoa loop,place the labelbeforethe loopwitha colonatthe end.

Format1:

```
loop1:for(…..)
{
……
……
}
```

Format2:Ablockofstatementscanbel abeled as shownbelow:

```
block1:
{
……..
……
block2:
{
……
}
…….
…….
}
```

CLASSES,OBJECTSANDMETHODSINT

RODUCTION

ϖCLASS:"**Aclassisawayof bindingthe dataandassociatedmethodsinasingle unit**"
ϖAnyJAVAprogram ifwewanttodevelop thenthat shouldbe developedwith respectiveclass onlyi.e.,withoutclassthere isnoJAVAprogram.
ϖClassescreateobjectsandobjectusesmethodstocommunicatebetweenthem.
ϖClassesprovideconvenientmethodforpackingtogetheragroupoflogicallyrelateddataitems and functionsthatworkonthem.
ϖThe dataitemsare calledfieldsandthefunctionsare calledmethods.

DEFININGACLASS

ϖAclassisa user-defineddata type witha template thatservestodefineitsproperties.
ϖAnythinginsquarebracketisoptional.

ϖClassnameandsuperclassnamearevalidjavaidentifiers.

ϖThe keyword extends indicates that the properties of the superclassname class are extended theclassnameclass.

Thebasicformofaclassdefinitionis
classclassname[**extends**superclassname]
{
[fieldsdeclaration;]
[methoddeclaration;]
}

Example1
classempty
{
}

Example2
classsample
{
inti,j;
voidgetdata();
}

FIELDSDECLARATION

ϖDataisencapsulatedinaclassbyplacingdatafieldsinsidethebodyoftheclassdefinition.
ϖThese variablesarecalled*instancevariables*becausetheyarecreatedwheneveranobjectofthe classisinstantiated.
ϖ***Instancevariablesarealsoknowasmembervariables.***

Example
classTriangle
{
intlength;i
ntheight;
}

Theclasstrianglecontainstwointegertypeinstancevariable,lengthandheight.

METHODSDECLARATION

ϖMethoddeclarationshavefourbasicparts:

- ¬Thenameofthemethod(method name)
- ¬Thetypeofthevaluethemethodreturns(type)
- ¬Alistofparameters (parameter-list)
- ¬Thebodyofthe method

Thegeneral formof the method declarationis

typemethodname(parameter-list)

{

method-body;

}

example

```
classTriangle
{
 intlength;i
ntheight;
voidgetdata(intx,inty)
{
length=x;
height=y;
 }
}
```

CREATINGOBJECTS

ϖ**OBJECT:**Inordertostore the data for the data membersofthe class,wemustcreate anobject.

¬Instance(instanceisamechanismof allocatingsufficientamountofmemoryspacefor datamembersofaclass)ofa classisknown asan object.

¬Classvariableis knownasan object.

¬Groupeditem(groupeditemisavariablewhichallowsustostoremorethanonevalue) isknownasanobject.

¬Valueformofaclassisknownasanobject.

¬ Blue printofaclassisknownasanobject.

¬Realworldentitiesarecalledasobjects.

ϖCreatinganobjectisalsoreferredtoasinstantiatinganobject.

ϖObjectsinjavaarecreatedusingthe**new**operator.

ϖThenewoperatorcreatesanobjectof thespecifiedclassandreturnsa referencetothatobject.

ϖEverytimetheclassisinstantiated,anewcopyofeachofthemiscreated.

Eg:

```
TriangleTri1;                    //
declareTri1=newTriangle()
                                 //instantiate
```

ϖThe firststatementdeclaresavariabletoholdtheobjectreference

ϖThe secondoneactuallyassignstheobjectreferencetothevariable.

ϖThe variabletri1isnowanobjectoftherectangleclass

Eg1:Triangletri1=newTriangle(); ☾ valid

Eg2: Triangletri1=newTriangle(); ☾ tri1andtri2aretheobjectsofTriangleclassTriangletri2=

newTriangle();

Eg3: Triangletri1=newTriangle();Tria

ngletri2=tri1;

ACCESSINGCLASSMEMBERS

ϖwecanaccessclassmembersusing **DOT(.)**operator

Syntax:

```
objectname. variablename =
value;objectname.methodname(paramete
r-list);
```

d

20);

```
classSquare
        {
                intside;
                voidgetdata(ints)
                {
                        side=s;
                }
                intrectarea()
                {
                intarea=side*side;r
                eturn(area);
                }
        }
classSquareaea
        {
                publicstaticvoidmain(Stringargs[])
                {
                        Square s1= new
                          Square();s1.getdata(5);
                          sqarea
                            =s1.rectarea();System.out.println(sqarea);
                    }
        }
```

CONSTRUCTORS

ϖJava supports a special type of method called a constructor that enables an object to initializeitselfwhen it is created.

¬Constructors havethe **samename asthe classname**.

¬They**doesnotreturnanyvalue** and**donotspecifyevenvoid**.

¬Constructors are**automaticallycalled**duringthe**creationof theobjects**.

EXAMPLE:

```
classVolume
{
      intx,y,z;V
      olume()          //constructor
        {
         x=10;
         Y=10;
         Z=30;
        }
       intcalvolume()
       {
           intvol=x*y*z;
           returnvol;
       }
}
classDemovolume()
{
     publicstaticvoidmain(Stringargs[])
      {
      Volumevolobj=newVolume;          //
      creatingobjectintresult=volobj.calvolume();
                                              // calling
      methodSystem.out.println("thevolumeis="+result);
       }
```

ADVANTAGESofconstructors:

1. Aconstructoreliminatesplacingthe defaultvalues.
2. Aconstructoreliminatescallingthenormal methodimplicitly.

RULES/PROPERTIES/CHARACTERISTICSofaconstructor:

1. Constructornamemustbesimilar tonameoftheclass.
2. Constructorshouldnotreturnanyvalueevenvoidalso(ifwewritethereturntype fortheconstructorthenthat constructorwill betreatedasordinarymethod).
3. Constructors should not be static since constructors will be called each and every timewhenever an object is creating.
4. Constructor should not be private provided an object of one class is created in another class(constructorcanbeprivateprovidedan objectofoneclass createdin thesameclass).
5. Constructors willnotbeinheritedatall.
6. Constructors arecalled automaticallywheneveranobjectiscreating.

TYPESOFCONSTRUCTORS:

ϖBasedoncreating objects in JAVAwehavetwotypes of constructors.

ϖTheyare

1. Default/parameterless/noargumentconstructorand
2. Parameterizedconstructor.

1. DEFAULTCONSTRUCTOR

¬Adefault constructorisonewhichwillnottakeanyparameters.

Syntax:

```
class<clsname>
{
clsname()//defaultconstructor
        {
        Blockofstatements;

……;

……;
        }
……;
……;
        };
```

```
classTest
{
        int a,
        b;Test(
        )
        {
        System.out.println ("I AM FROM
DEFAULTCONSTRUCTOR...");
        a=10;b=20;
        System.out.println("VALUEOFa="+a);Sy
        stem.out.println("VALUEOFb="+b);
        }
};
classTestDemo
{
        publicstaticvoidmain(String[]args)
        {
        Testt1=newTest();
        }
};
```

2. Parameterizedconstructors

DBC

METHODOVERLOADING

- Methods that have the same name, but different parameter lists and different definitions arecalledmethodoverloading.
- Methodoverloadingis usedwhenobjects arerequiredto performsimilartasks butusingdifferent inputparameters.

ϖJava matches up the method name first and then the number and type of parameters to decidewhichoneofthedefinitiontoexecute.Thisprocess isknownas**polymorphism.**

```
classCalculatearea
{
          intarea()
          {
            int
            val=10*20*30;ret
            urnval;
           }
          intarea(intm,intn)
          {
            intval=m*n;
            returnval;
           }
         intarea(intl, intm,int n)
            {
            int
            val=l*m*n;ret
            urnval;
            }
          floatarea(floatm,floatn)
           {
            float
            val=m*n;retur
            nval;
            }
         floatarea(intl,floatm,intn)
            {
           float
            val=l*m*n;retur
            nval;
            }
 }
classDemovolume()
{
  publicstaticvoidmain(Stringargs[])
      {
        Calculatearea calarea=new
       calculatearea();intresult1,result2,result3;flo
       at
      result4,
      result5;result1=calarea.area();resul
      }t ti t llt t it t i i ular
}
```

bject.

- Itdefinesao r
- Thatisthememberbelongstotheclassasawholerather thantheobjectscreatedformtheclass.
- Staticmemberscan bedefined asfollows

staticintcount;

staticintmax(intx,inty);

- Staticmembersareassociated withtheclassitselfratherthanindividualobject.
- Staticvariablesandstaticmethods areoften referredto asclassvariablesandclassmethods.
- Staticvariablesareusedwhenwewant tohaveavariable commontoallinstances ofaclass.
- Eg:variablethatkeepacount ofhowmanyobjectsof aclasshave beencreated.
- Staticvariables andstaticmethodsarecalled withoutusingtheobjects.

Restrictionofstaticmethods

¬ Theycancall onlyother staticmethods.
¬ Theycan only access staticdata.
¬ Theycannot referto **this**or**super**in

```
anyway.classmathoperation
{
   staticfloatmul(floatx,float y)       // staticmethod
    {returnx*y;
     }
   staticfloatdivide(floatx,floaty)//staticmethod
     {returnx/y;
      }
}
classMathapplication
 {publicstaticvoidmain(Stringargs[])
      {
         float                          a=
         mathoperation.mul(4.0,5.0);float
         b=mathoperation.divide(4,2.0);Syst
         em.out.println(a,b);
       }
 }
```

A t ll i it t t l . i n asnesti t .

Example

classNesting

```
{
      int a, b,
    result;Nesting(intx,
    inty)
      {
       a=x;b
       =y;
       }
   voidProcess()
      {result=a+b;d
       isplay();
      }
  voiddisplay()
    {System.out.println(result);}
}
classnestingtest
{
 publicstaticvoidmain(Stringargs[])
 {Nestingnest=newNesting(10,20);n
   est.process();
   }}
```

INHERITANCE

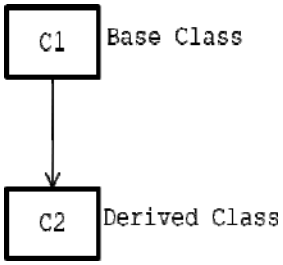
ϖWhen one class acquires the properties of another class it is known as ***inheritance.(***Themechanismofderivinganewclassfrom an oldoneiscalled inheritance.)
ϖAclassthatisinheritediscalledasuperclassorbaseclass.
ϖThe classthatdoestheinheritingiscalledasubclassorderivedclass.
ϖAdvantageofinheritanceisthatitallowsreusabilityofcoding.
ϖInheritancemay takedifferentforms. They are

- Singleinheritance(onesuperclass,onesubclass)
- Multipleinheritance(severalsuperclasses)
- Hierarchicalinheritance(onesuper class andmanysubclasses)

ϖMultilevelinheritance(derivedfromaderivedclass)

1. **Singleinheritance**

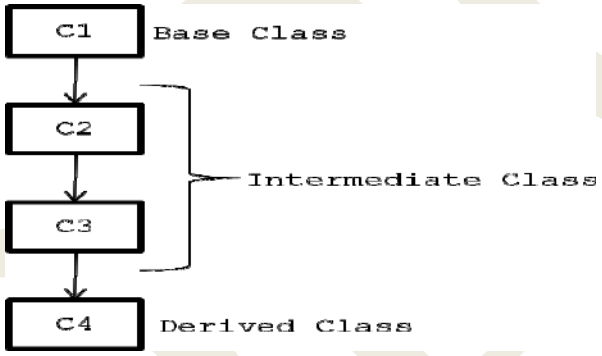
¬Singleclassisoneinwhichthereexistssinglebaseclassandsingle derivedclass.



2. **Multilevelinheritance**

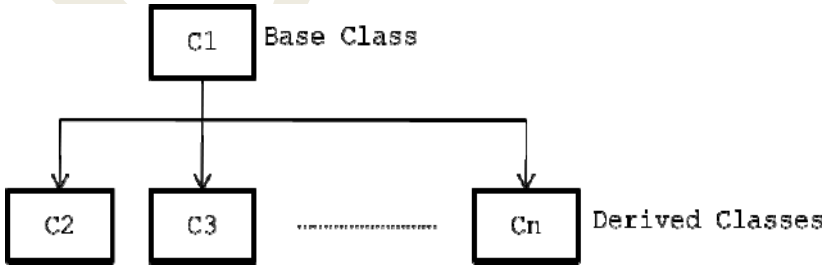
¬Multilevelinheritanceisonewhichthereexistsinglebaseclass,single derivedclassandn numberofintermediatebaseclasses.

¬An intermediate base class is one, in one contest it acts as bass class and in another context it actsasderived class



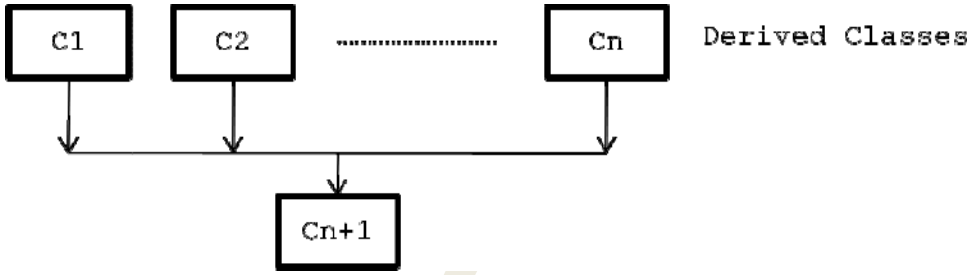
3. **Hierarchicalinheritance**

¬Hierarchical inheritance is one in which there exits single base class and n number of derivedclasses.



4. **Multipleinheritance**

¬Multipleinheritanceisoneinwhichthereexistsnnumberofbaseclassesandsinglederived



classes.

¬Multiple inheritances are one supported by JAVA through classes but it is supported by JAVAthroughtheconcept ofinterfaces.

Definingasubclass

ϖAsubclassisdefinedasfollows:
classsubclassnameextends superclassname

{

variables

declaration;methodsd

eclaration;

}

Thekeywordextendssignifiesthatthepropertiesofthesuperclassnameareextendedtothe subclassname.

SINGLEINHERITANCE

ϖWhenasinglesubclassextendsthepropertiesofasi *levelinheritance*.

ϖExample

```
classRoom
{
        int Length,
        Breadth;Room(intx,
        inty)
        {
                Length=x;
                Breadth=y;
        }
        intvolume()
        {
                return(Length*Breadth*Height);
        }
}
classinhert
{
publicstaticvoidmain (Stringargs[])
{
Room1 obj=new
Room1(14,12,10);intarea1=obj.area
();
int
volume1=obj.volume();System.out.println
(“Area1
=”+area1);System.out.println(“Volume=”+
volume1);

}
```

}

DBC

```
        intarea ()
        {
                return(Length*Breadth);
        }
}
        classRoom1 extendsRoom
        {
                intheight;
                Room1 (intx,int y,intz)
                {
                        super(x,y);
                        Height=z;
                }
        }
```

SUBCLASSCONSTRUCTOR

ϖAsubclassconstructorisusedtoconstructtheinstancevariablesofboththesubclassandthesuperclass.

ϖThesubclassconstructorusesthekeywordsupertoinvoketheconstructormethodofthesuper class.

ϖThe keywordsuperisusedinfollowingconditions.

- Supermayonlybeusedwithin asubclassconstructor method.
- Thecalltosuperclassconstructormustappearasthefirststatementwithinthesubclassconstructor.
- Theparameterinthesupercallmustmatchtheorderandtypeoftheinstance variable declared intheprogram.

MULTILEVELINHERITANCE

ϖSince javasupportstheconceptofconceptofinheritanceitisextensivelyusedinbuildingthe classlibrary.

ϖMultilevelinheritanceisusedtobuildchainofclasses.

Example

```
classRectangle
{
intL,W;
Rectangle(inti,intj)
 {
        L=i;
        W=j;
        System.out.println("Area="+(L
   }
}
classBox extendsRectangle
{
Box(int i,int j,intk)
 {
        super(i,j);
```

```
classCircleextendsBox
{
circle(inti,intj,intk)
{
super(i, j, k);System.out.println("Area="+(3.14*i*i));
}
}
classmainclass
{
publicstaticvoidmain(Stringargs[])
{
Circleobj=newCircle(10,
}
}
```

Output

```
Area=200

Volume=6000
```

```
        System.out.println("Volume="+(i*j*k));
    }
}
```

HIERARCHICALINHERITANCE

ϖWhen a finally of classes is created in hierarchical model then it is called as *hierarchicaalinheritance***.**

Example

```
classA
{
        int a,
        b;voidinput
        ()
        {
                a=10;b=20;
                System.out.println("a="+a+"b="+b);
        }
}
classBextendsA
{
        void addition ()
        {
                System.out.println("a+b="+(a+b));
        }
}
class C extendsA
{       void product()
        {
```

```
classmainclass
{
publicstaticvoidmain(Stringargs[])
        {
        Aobja=newA();
        Bobjb=newB();
        C objc=new
        C();obja.input();
        objb.a=10;objb.
        b=20;objb.addti
        on();objc.a=10;

        objc.b=20;objc
        .product();
        }
```

```
                System.out.println("a*b="+(a*b));
        }
    }
```

OVERRIDINGMETHODS

ϖAmethoddefinedinasuperclassisinheritedbyitssubclassandisusedbytheobjectscreatedby thesubclass.

ϖTheremaybeoccasionswhenwe wantanobjecttorespondtothesamemethodiscalled.

ϖThisispossible bydefiningamethodinthe subclassthathasthe samename,sameargumentsand samereturntypeasamethod inthesuperclass.

ϖWhenthemethods arecalled,themethoddefinedinthesubclassisinvokedand executedinsteadof theoneinthesuperclass.Thisisknownas***overriding.***

ϖExample

```
classSuper
{       int x;
        Super (int x)
        {
                this.x=x;
         }
 voiddisplay()
        {
           System.out.println("x="+x);
        }
  }
classSubextendsSuper
{       inty;
        Sub(intx,inty)
        {
                super
                (x);this.y
                =y;
        }
}
voiddisplay()
{
        System.out.println("Superx="+x);System.out.print
        ln("Suby="+y);
}       }
classoverridetest
{
publicstaticvoidmain(Stringargs[])
{
        SubS1=newSub(100,200);S1.display();
}
}
```

FINALVARIABLESANDMETHODS

ϖItpreventsthesubclassesformoverridingthememberof thesuperclass.

ϖFinalvariablesandmethodsaredeclaredasfinalusingthe keyword**final**asa modifier.

Example: size=100

DBC

finalintSIZE=100;
finalvoidshowstatus(){ .....}

ϖMakingamethodfinalensuresthatthe functionalitydefinedinthatmethodwillneverbealtered inanyway.

ϖThe valueofafinalvariablecanneverbechanged.

FINALCLASSES

- Aclassthatcannotbesub-classed iscalledafinalclass.
- Itpreventsaclassbeingfurther sub-classedforsecurityreasons.
- Anyattempttoinheritheseclasseswillcauseanerror.

```
finalclassAclass
{
}
FinalclassBclassextendsomeclass
{
}
```

DS

ϖIn t anobjectwhenitisdeclared.This

ϖFinalizationafinalizermethodisjustoppositetoinitialization,itautomaticallyfreesupthe memoryresourcesusedbytheobjects.Thisprocessisknownasfinalization.

ϖItactslikeadestructor.

ϖ The method can be added to any class.The**finalize()**methodhasthisgeneralform:

```
protected voidfinalize()
{
//finalizationcodehere
}
```

Here,thekeyword**protected**isaspecifierthatpreventsaccessto**finalize()**bycodedefinedoutsideits class.

ABSTRACTMETHODSANDCLASSES

ϖInJAVAwe have twotypes ofclasses.

Theyare

1. **concreteclassesand**
2. **abstractclasses.**

ϖ**A concrete class** is one which contains fully defined methods. Defined methods are alsoknown as implemented or concrete methods. With respect to concrete class, we can createanobject ofthat class directly.

ϖ**Anabstractclass**isone whichcontainssome definedmethodsandsomeundefinedmethods.Undefined methods are also known as unimplemented or abstract methods. Abstract method is onewhich doesnotcontainanydefinition.Tomakethemethod asabstractwehavetouseakeyword called

abstractbeforethefunctiondeclaration.

SyntaxforABSTRACTCLASS:

```
abstractclass<clsname>
{
Abstractreturn_typemethod_name(methodparametersif any);
};
```

ABSTRACTMETHODS

ϖWhenamethodisdefinedasfinalthanthatmethodisnotre-definedina subclass.

ϖJavaallowsamethodtobere-definedinasubclassandthosemethods arecalled***abstractmethods.***

ϖWhenaclasscontainsone ormoreabstractmethods,thenitshouldbedeclaredasabstractclass.

ϖWhenaclassisdefinedasabstractclass,itmustsatisfyfollowingconditions

¬We can't use abstract classes to instantiate objects directly. For

exampleOps =new Op()

isillegalbecause Opisanabstractclass.

¬Theabstractmethodsofanabstractclassmustbedefinedinitssubclass.

¬Wecan'tdeclareabstractconstructorsorabstractstaticmethods.

ϖFinalallowsthemethodsnotredefineinthesubclass.
ϖAbstractmethodmustalwaysbe redefinedina subclass,thusmakingoverridingcompulsory.
ϖThisisdoneusingthemodifierkeyword**abstract**inthemethoddefinition.
ϖWhenaclasscontainsoneormoreabstractmethods,itshouldalsobe declaredabstract.

Example

abstractclassshape

{

……..

……

abstractvoid draw();

……

……..

}RULE

S:

ϖWecannotuseabstractclasses toinstantiateobjectsdirectly.
ϖTheabstractmethodsofanabstractclassmustbedefinedinitssubclass.
ϖWecannotdeclareabstractconstructorsorabstractstaticmethod.

METHODSWITHVARARGS

ϖVarargs represents variablelength arguments in methods.
ϖItmakesthatjavacodesimpleandflexible.

Generalform

<accessspecifier><static>voidmethod-name(object…arguments)
{
}

ϖ**Intheabovesyntax** The methodcontainsan argumentcalledvarargsinwhich
¬ Objectisthetypeof anargument
¬Ellipsis(…)isthekeytovarargs
¬Argumentit shonameofhextriable

Example program

Classexampleprg

Foreg: Publicvoidsample(Stringusername,Stringpassword,Stringmail)
CanbewrittenasPublicvoidsample(String…var_name)
Here var_name is thevariablenamethatspecifiesthatwe can passany number ofStringargumentstothesamplemethod.

```
    System.out.println("hello"+name);
   }
 }
publicstaticvoidmain(Stringars[])
{
 exampleprg("ram","siva","suriya");
}
}
```

VISIBILITYCONTROL

- Themodifiersarealsoknownas**accessmodifiers.**
- Javaprovidesthreetypesofvisibilitymodifiers:**public,privateandprotected.**

PublicAccess:

- Todeclarethevariableormethodaspublic,itisvisibletotheentireclassinwhichit isdefined.

Example:

```
publicintnumber;
publicvoidsum( ){.....}
```

FriendlyAccess:

- Whennoaccessmodifierisspecified,thenumberdefaultstoalimitedversionofpublic accessibilityknownas“friendly”level ofaccess.
- Thedifferencebetweenthe“public”and“friendly”accessisthatthepublicmodifiermakes fieldsvisibleinallclasses.
- Whilefriendlyaccessmakesfieldsvisibleonlyinthesamepackage,butnotinotherpackage.

ProtectedAccess:

- The**protected**modifiermakesthefieldsvisiblenotonlytoallclassesandsubclassesinthe samepackagebutalsoto subclassesin otherpackages.
- Non-subclassesinotherpackagescannotaccessthe“protected”members.

PrivateAccess:

- Privatefieldsareaccessibleonlywiththeirownclass.
- Theycannotbeinheritedby subclassesandthereforenotaccessibleinsubclasses.
- Amethoddeclaredas**private**behaveslikeamethoddeclaredas**final.**

PrivateprotectedAccess:

- Afieldcanbedeclaredwithtwokeywords**private**and**protected**togetherlike:
  privateprotectedintcodeNumber;
- Thisgivesavisibilitylevelinbetweenthe“protected”accessand“private”access.

Rules:

1. Use**public**ifthefieldistobevisibleeverywhere.
2. Use**protected** ifthe fieldistobevisibleeverywhereinthecurrent packageandalso subclassesinother packages.
3. Use“default”ifthefieldistobevisibleeverywhereinthecurrentpackageonly.
4. Use**privateprotected**ifthefieldistobevisibleonlyinsubclasses,regardlessof packages.
5. Use**private**if thefieldis**not**tobevisibleanywhereexceptin itsownclass.

QUESTIONS

2Marks

1. Whatarethetypesof branching?

DBC

2. Whatdoyoumeanbyconditionaloperator?
3. Definelooping.
4. Whatisthedifferencebetween entrycontrolledloopandexitcontrolledloop?
5. Whatistheuseofbreak statement?
6. Definecontinuestatement.
7. DefineConstructor.
8. Whatdoyoumeanbyinheritance?
9. Definemethodoverriding.

5Marks

1. ExplainindetailaboutIfstatements.
2. Describeaboutswitchstatement.
3. ExplainindetailaboutLoopingstatements withsuitableexamples.
4. WriteaJavaprogram usingparameterizedanddefaultconstructor.
5. WriteshortnotesonMethodoverloadinginjava.
6. ExplainindetailaboutAbstractmethodsandclasses.
7. Describeabout Finalvariables andmethods.

10Marks

1. Explainindetailabout Classesand Objects.
2. Explainindetailaboutinheritanceand itstypes.

UnitII completed

DBC

UNITIII

ARRAYS,STRINGSANDVECTORSINTR

ODUCTION

ϖAnarrayisagroupofcontiguousorrelateddataitemsthat shareacommonname.

ϖAparticularvalueinanarray isindicatedbywritinganumbercalledindexnumberor subscriptinbrackets afterthearrayname

E.g.name[10]

ϖTheindividualvaluesarecalledelements.

ONE-DIMENSIONALARRAYS

ϖAlistofitemscanbegivenonevariablenameusingonlyonesubscriptcalledsinglesubscriptedvariableor onedimensional array.

```
Eg:intnumber[]=newint[5];
```

Forexample,Thevaluestothearrayelements35,40,20,57,19 canbeassignedasfollows:number[0]=35;

number[1] =40;

number[2] =20;

number[3] =57;

number[4]=19;

CREATINGANARRAY

ϖArraysmustbedeclaredandcreatedinthecomputermemorybefore they areused.

Creationofanarrayinvolvesthreesteps:

- ¬ Declaringthearray
- ¬ Creatingmemorylocations
- ¬ Puttingvaluesinto thememorylocations.

Declarationof arrays

Creatingmemorylocations

ϖAfterdeclaringanarray,weneedtocreate memory.

ϖJavaallowscreatingarraysusing **new**operator.

arrayname = new type

[size];Eg:number=new int[5];

Combiningdeclarationandcreation

ϖItisalsopossibletocombinethetwosteps-declarationandcreation

ϖ**Int number[]=new**

int[5];InitializationofArrays

ϖPutting valuesintothearrayisknownasinitialization.

ϖThisprocessisdoneusingthearraysubscript.

arrayname[subscript]=value;

Eg:

number[0]=90;n

umber[1]=100;

…….

ϖArraysstartswithasubscript0andendswithavalueone lessthanthe sizespecified.

ϖTryingtoaccessanarraybeyonditsboundarieswillgenerate anerrormessage.

Initializinglistofvaluestoanarray

ϖValuesareseparatedbycommasandsurroundedbycurlybraces.

typearrayname[]={list of values};

ϖIf size is not specified means compiler allocates enough space for all the elements specified inthelist

Eg: **intnumber[]={23,45,56,67}**

Arraylength

ϖAccessthelengthofthearrayusing**length**keyword.

Eg: intassize=a.length();

Eg: intsize[][]=newint[3][10];

¬Herelength ofsize.length() ☾ 3

¬Lengthofsize[i].length() ☾ 10;

TWODIMENSIONALARRAYS

Byusingtwo dimensionalarrays wecan storetableofvalues.

Given: int twoD [ ] [ ] = new int [4] [5] ;

Eg : **int**

myarray[][];myarray=

newint[3][4];

Or**intmyarray[][]=newint[3][4]**

Exampleprogram:

```
classMulTable
 {
   final static int ROWS
   =0;finalstaticintcolumns=20;
    publicstaticvoidmain(Stringargs[])
     {
          intproduct [ ] [ ] = new int [ ROWS ]
          [COLUMNS];introw,column;
          System.out.println("MULTIPLICATIONTABLE");
           System.out.println("");
         inti,j;
        for(i=10; i<ROWS;i++)
              {
                   for(j=10; j<COLUMNS;j++)
                    {
                         product [i][j]=i *
                         j;System.out.println(""+ product[ i][ j];
                     }
                        System.out.println("");
                }
          }
```

DBC

}

STRINGS

ϖStringrepresentsasequenceofcharacters.

ϖStringcanberepresentedintwowaysinjava

¬Usingcharacterarray

¬Usingstringobjectorstringbuffer class

Usingcharacterarray

```
charcarray[]=newchar[4];c
array[0]='j';
carray[1]='a';
carray[2]='v';
carray[3]='a';
```

Usingstring class

ϖIn java stringsareclassobjectsand implementedusingtwo classes;Stringand StringBuffer.
ϖA java string is an instantiated object of the **String c**lass. Strings may be declared and created asfollows:

Generalform

```
String stringname;
Stringname=new  String("string");
```

Eg

```
Stringfirstname;firstname=ne
wString("anil");
        Or
Stringfirstname=newString("anil");
```

Togetlengthofthestring ☾ intm=firstname.length();

Toconcatenatetwostring ☾ Stringfullname=name1+name2;Stringcity=”new”+”delhi”;

Stringarrays

Eg : Stringitem[]=newString[3];

ϖThestringarrayitemstoresthreestrings.

item[0]=”soap”;ite

m[1]=”biscuits”;ite

m[2]=”powder”;

Stringmethods

Mostcommonly usedstringmethods

Method	taskperformed
s2=s1.toLowerCase;	Converts thestrings1toalllowercase
s2=s1.toUpperCase’;	Convertsthestrings1toalluppercase
s2=s1.replace(‘x’,’y’);	Replace allappearances ofx withy
s1.equal(s2);	Returnstrueifs1 isequal tos2
s2=s1.trim();	RemovewhitespaceatthebeginningandendoftheString s1
s1.equalsIgnoreCase(s2);	Returnstrueifs1isequalto s2,ignoringthecaseofcharacters
s1.length();	Givesthelengthofs1.
s1.charAt(n)	Givesnthcharacterofs1
s1.concat(s2);	Concatenatess1ands2

s1.substring(n);	Givessubstringstartingfromnthcharacter.
s1.substirng(n,m);	Givessubstringstartingfromnthcharacterupto mthcharacter
String.valueOf(p);	Createsastringobjectoftheparameterp(simpletypeorobject)
p.toString();	Createsastringrepresentationofobjectp
s1.indexOf('x')	Givesthe position ofthefirstoccurrenceof'x'in the strings1
s1.indexOf('x','n');	Givesthe position 'x'that occursafter nthpositionin thestring s1
String.valueOf(variable);	Convertstheparametervaluetostringrepresentation.
s1.compareTo(s2)	Returns negativeifs1<s2,positiveif s1>s2,zeroifs1ands2equal.

ExampleprogramAlphabeticorderingofstrings

```
classStringordering
{
 static String name[]={"Madras", "Delhi, "Ahmedabad", "Calcutta","
                Bombay"}publicstaticvoid main(Stringargs[]);
                        {
                          int
                         size=name.lenght();Stri
                         ngtemp=null;
                                for(inti=0;i<size;i++)
                                {
                                  for(intj=i+1;j<size;j++)
                                 {
                                 ifname[j].comapreTo(name[i])<0)
                                     {
                                        temp=name[i];na
                                        me[i]=name[j];na
                                        me[j]=temp;
                                      }
```

```
                    }
                }
                for(inti=0;i<size;i++)
                {
                    System.out.println(name[i]);
                }
            }
        }
```

StringBufferclass

- ϖStringBufferclassisapeerclassofString.
- ϖStringcreatesstringsoffixed_length
- ϖStringBuffer class creates string of flexible length that can be modified in terms of both lengthandcontent.
- ϖIn stringbuffer class we can insert characters and substrings in the middle of a string, or appendanotherstringtotheend.

Eg:StringBufferstr=newStringBuffer("annu");

Methods	**Task**
s1.setChartAt(n,'x');	Modifies thenthcharactertox s1.append(s2);
Appendsthestrings2tos1atthe	ends1.insert(n,s2); Insertsthe
strings2 atthepositionnofthe	strings1
s1.setLength(n);	Setsthelengthofthestrings1 ton.ifn<s1.length()s1istruncated.If n>s1.length()zerosare addedtos1.

Exampleprogram

```
classstringmanipulation
 {
```

```
publicstaticvoidmain(Stringargs[])
        {
        StringBufferstr=newStringBuffer("objectlanguage");S
        ystem.out.println("originalstring:"+str);
                //obtainstringlength
        Systme.out.println("lengthofstring:"+str.lenght());
                //accessingcharactersinastring
                for(inti=0;i<str.length();i++)
                {
                intp=i+1;
                System.out.println("characteratposition:"+p+"is"+str.charAt(i));
                }
                //inserting astringinthemiddle
                String astring=new String(str.toString());int
                pos=astring.indexOf(" language");
                str.insert(pos,
                "oriented");System.out.println("modifiedst
                ring:"+str);
                // modifying
                charactersstr.setcharAt
                (6,'-');
                System.out.println("stringnow:"+str);
                // append a string at the end
                str.append("improve
                security.");System.out.println("appendingstr
                ing:"+str);
        }
}
```

VECTORS

ϖVectorclasscontaininjava.utilpackage.

ϖVectorclass is usedto create agenericdynamicarrayknownas vectorthatcan holdobjects ofanytypeandanynumber.

ϖItisheterogeneousdatanothomogenous.

Createdlike

¬Vectorintvect=newvector();//declaredwithoutsize

¬Vectorlist=newvector(3);//declaredwithsize

Advantages

ϖItisconvenienttousevectorstostoreobjects.

ϖAvectorcanbeusedtostorealist ofobjectsthatmayvaryinsize.

ϖWecanaddanddeleteobjectsfromthelistasandwhenrequired.

Methods	Taskperformed
list.addElement(item)	Addstheitem specifiedtothelistat theend
list.elementAt(10)	Givesthenameofthe10th object
list.size()	Givesthenumberofobjectspresent.
list.removeElement(item)	Removesthespecified itemfromthelist.
list.removeElementAt(n)	Removes the item stored in the nth position of thelist
list.removeAllElements()	removesall theelementsinthelist
list.copyInto(array)	copiesallitemsfromlisttoarray
list.insertElementAt(item,n)	Insertstheitematnthposition

Example program

```
importjava.util.*;classLanguageVector
{
        publicstaticvoidmain(Stringargs[])
        {
                Vectorlist=newVector();intlength=args.length;
            for(inti=0;i<length;i++)
                {
                list.addElement(args[i]);
                   }
                 list.insertElementAt("COBOL",2);intsize=list.size();
                  String listarray[]=new String(size);List.copyInto(listArray);System.out.println("Listoflanguages");
            for(inti=0;i<length;i++)
                 {
                 System.out.println("listArray[i]");
                }
        }
}
```

Commandlineinputandoutputare:

C:\>javaLanguagevectorAdaBasicC++FORTRANJava

Output:

Listoflanguages

Ada

BasicCOBOL

C++FORT

RAN

Java

WRAPPERCLASS

ϖVectorscannothandleprimitivedatatypes likeint, float,doubleand char.

ϖPrimitivedatatypesconvertedintoobjecttypes.

ϖThisconversionisdonebyusingthewrapperclasscontainsinthejava.langpackage

Convertingprimitivenumberstoobjectnumbersusingconstructormethods

Constructorcalling	**Conversionaction**
IntegerIntval=newInteger(i);	PrimitiveintegertoIntegerobject
FloatFloatval=newFloat(f);	Primitivefloatto Floatobject
DoubleDoubleval=newDouble(d);	PrimitivedoubletoDoubleobject
LongLongval=new Long(l)	PrimitivelongtoLongobject

ConvertingobjectnumberstoprimitivenumberusingtypeValue()method

Methodcalling	**ConversionAction**
inti=Intval.intValue();	Objecttoprimitiveinteger
floatf=Floatval.floatValue();	Objecttoprimitivefloat
longl=Longval.longValue();	Objecttoprimitivelong
doubled=Doubleval.doubleValue();	Objecttoprimitivedouble

ConvertingnumberstostringusingtoString()method

Methodcalling	**ConversionAction**
str=Integer.toString(i);	Primitiveintegertostring
str=Float.toString(f);	Primitivefloattostring
str=Double.toString(d);	Primitivedoubletostring
str=Long.toString(l);	Primitivelongtostring

ConvertingstringobjectstonumericobjectsusingthestaticmethodValueOf()

Methodcalling	**ConversionAction**

DoubleVal=Double.ValueOf(str); ConvertsstringtoDoubleobject

DBC

E.g.forPrimitivetoobject

floatnum1=0.3F; ☾ primitivedatatype

Floatf1=newfloat(num1) ☾ changingtoobject

E.g.forconverting stringtonumericobject

float

num1=Float.ValueOf(in.readLine());intva

l=Integer.ParseInt(in.readLine());

E.g.:objecttoprimitive

floatnum1=0.3F; ☾ primitivedatatype

Floatf1=newfloat(num1) ☾ changingprimitive toobject

num1=f1.floatValue(); ☾ changingobjecttoprimitive

ENUMERATEDTYPES

- Javaallowsustousetheenumeratedtypeusingthe **enum**keyword.
- This keywordcanbeusedsimilartothestaticfinalconstantsintheearlierversionofJava.

```
publicclassDays
  {
      public static final int
      DAY_SUNDAY=0;public static final int
      DAY_MONDAY=1;public static final int
      DAY_TUESDAY=2;publicstaticfinalintDAY
      _WEDNESDAY=3;public static final int
      DAY_THURSDAY=4;public static final int
      DAY_FRIDAY=5;publicstaticfinalintDAY_S
      ATURDAY=6;
```

}

Usingtheenumeratedtypefeaturethe above codecanbewrittenas

publicenumDay{SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY}

Advantages:

- Compile-timetypesafety.
- Wecanusetheenum keywordinswitchstatements.

ANNOTATIONS

⎪ Itisalsoknownasmetadata.

⎪ We can use this feature to merge additional java elements with the programmingelements,suchasclasses,methods,parameters, localvariables,packages,andfields.

⎪ Metadataisstoredinjavaclassfilebythecompiler

⎪ TheseclassfilesareusedbytheJVM.

⎪ Javacontainsthefollowingstandardannotations.

Annotation	Purpose
@Deprecated	Compiler warns when deprecated java elements are used in non-deprecatedprogram.
@Overrides	Compiler generated error when the method uses this annotation typedoesnot overridethemethodspresent inthesuper-class.

Java also contains some meta-annotations available in the java.lang.annotation package. The followingtableprovides themeta-annotations:

Meta-annotation	Purpose
@Documented	Indicatesannotationofthistypetobedocumentedby Javadoc.Indic
@Inherited@	atesthatthis typeisautomaticallyinherited.
Retention@Ta	Indicatestheextendedperiodusingannotationtype.
rget	Indicatestowhichpramelementtheannotation isapplicable.

Forexample,considerthefollowingcode thatcontainsthedeclaration ofan annotation:

```
package
njunit.annotation;Importjava.l
ang.annotation.*;
@Retention(Retentionpolicy.RUNTIME)
@Target
((ElementType.METHOD))public@interf
aceUnitTest
{
   Stringvalue( );
}
```

- Where@Retentionis ameta-annotaion,whichdeclaresthatthe@UnitTestannotationmustbestoredinaclassfile.
- @Targetmeta-annotaionisusedtodeclare the @UnitTestannoatation.,whichannotatesthe methodsinthejavaclassfiles.
- The @interface meta-annotaion is used to declare the @ UnitTest annoatation with the membercalledvalue,whichreturns Stringas anobject.

Whileusingannotations,weneed tofollowtherules:

- Donotuseextendclause.Itautomaticallyextendsthemarkerinterface java.lang.annoataion.Annotation
- Donot use anyparameter foramethod
- Donotuse genericmethods.
- Donotusethrowsclause.

INTERFACESIN

TRODUCTION

- Interfacesarebasically usedtodevelopuserdefineddatatypes.
- Withrespecttointerfaceswecanachieve theconceptofmultiple inheritances.
- Withinterfaceswecanachievetheconceptofpolymorphism,dynamicbindingandhence wecanimprovethe performance ofa JAVAprograminturnsof memoryspaceandexecutiontime.
- An interface is a construct which contains the collection of purely undefined methods oraninterfaceis a collectionofpurelyabstract methods.

DEFININGINTERFACES

```
interface<InterfaceName>
{
   variables
```

- i i t

Example

```
interfaceItem t
{
      staticfinalintcode=1001;
```

```
    declaration;methodsd
    eclaration;
}
```

DBC

ϖInterfacenamerepresent aJAVAvalid variablename.

Variablesaredeclaredasfollows:

staticfinaltypeVariableName=Value;

Allv [redacted] declarationwillcontainonlyalistof methodswithoutanybodystatements.

[redacted]

EXTENDINGINTERFACES

ϖInterfacecanalsobeextended.

ϖAninterfacecanbesub-interfacedfromotherinterfaces.

ϖThenewinterfacewillinherit allthememberof thesuper-interface.

ϖInterfacecanbeextendedusingthekeywordextends.

```
interfacename2extendsname1
        {
                Bodyofname2

        }
```

Forexample,wecan putall theconstantsinoneinterfaceand themethodsintheother.

```
interfaceItemConstants
  {
        intcode=
        1001;stringname="Fa
        n";
  }
interfaceItemextends ItemConstants
        {
           voiddisplay();
        }
```

Theinterface**Item**willinheritboththeconstants**code**and**name**intoit.

IMPLEMENTINGINTERFACES

ϖImplementtheinterfaceusing implementskeyword.

Generalform1
classclassname**implements**
interfacename
{
bodyofclassname
}

Generalform2
class classname**extends**supperclass**implements**
interface1,interface2,……..
{
bodyofclassname
}

ACCESSINGINTERFACEVARIABLES

ϖInterfacedeclaresasetofconstantvalues.

ϖThe constantvalueswillbeavailabletoanyclassthatimplementstheinterface.

Example

interfaceinterfaceA
{
finalintm=10;
}

classmainpgm
{
publicstaticvoidmain(Stringargs[])
{
ClassAobjA=newclassA();
objA.show();

Example1

interfaceArea
{
finalstaticfloatpi=3.14F;
floatcompute(flaotx,floaty);
}
classRectangleimplements Area
{
publicfloatcompute(float x,floaty)
{
return(x*y);
}
}

classCircleimplementsArea
{
publicfloatcompute(floatx,floaty)
{
return(pi*x*x);}
}

classInterfaceTest
{
publicstaticvoidmain(Stringargs[])
{
Rectanglerect=newRectangle();
Circlecir=newCircle();
Areaarea;ar
ea=rect;
System.out.println(area.compute(10,20));
area =cir;
System.out.println(area.compute(10,0));
}
}

```
    }
}
```

DBC

PACKAGESINT

RODUCTION

- ϖ**Apackageisacollectionofclasses,interfacesandsub-packages.**
- ϖAsub-packageinturnsdividesintoclasses,interfaces,sub-sub-packages,etc.
- ϖLearningabout JAVAis nothing but learningabout various packages.
- ϖ Bydefaultone predefinedpackageisimportedfor eachandeveryJAVAprogramand whose nameis**java.lang.*.**
- ϖWhenever we develop any java program, it may contain many number of user definedclassesanduserdefined interfaces.
- ϖIfwearenotusing anypackagenametoplaceuserdefined classesandinterfaces,JVMwillassumeitsownpackage called NONAMEpackage.
- ϖInjavawe have twotypesofpackagestheyare**I)predefined or built-in or core packages andII)userorsecondaryorcustomdefinedpackages**.

BENEFITS:

- ϖTheclassescontainedinthepackagesofotherprogramscanbeeasilyreused.
- ϖInpackages,classescanbeuniquecomparedwithclassesinotherpackages.
- ϖThatistwoclassesin twodifferentpackagescanhavethesamename.
- ϖTheymaybereferredbytheirfullyqualifiedname,comprising thepackagenameandtheclass name.
- ϖPackagesprovideawayto"hide" classesthuspreventingotherprogramsorpackagesfrom accessingclassesthatare meantforinternaluseonly.
- ϖPackagesalsoprovideaway forseparating"design"form"coding".

JAVAAPIPACKAGES

Packages	Contents
java.lang	Languagesupportclasses.
java.util	Languageutilityclassessuch asvectors,hashtables,randomnumber,date etc.
java.io	Inputandoutputclasses.
java.awt	Setofclassesforimplementing graphicaluserinterfaces.
java.net	Classesfornetworking.
java.applet	Classesforcreatingandimplementingapplets.

USINGSYSTEMPACKAGES

ϖPackagesareorganizedinahierarchicalstructure.

ϖThe package namedjavacontainsthepackageawt,whichinturnscontainvariousclasses requiredforimplementinggraphicaluserinterface.

Therearetwowaysofaccessingtheclassesstoredinapackage

1. Usingthefullyqualified classname.(Usingthepackage name containingthe classand then appendingtheclass namebyusingthedotoperator.)

E.g.java.awt.Color

- Bestandeasiestoneto accesstheclass
- Usedonlyonce,notpossibletoaccessother classesofthepackage.

2. Using the import statement, appear at the top of the file. Imported package class can beaccessed anywhereintheprogram

Syntax:importpackagename.classname;

Or

Importpackagename.*

Theseareknown as importstatementsand mustappear at thetopofthefile,before anyclassdeclarations,**import** is akeyword.

⎪Thefirststatementallowsthespecifiedclassinthespecifiedpackagetobeimported.Forexample,thestatement

importjava.awt.Color;

importstheclass**Color**andtherefore theclassnamecannowbedirectlyusedintheprogram.

⎪Thesecondstatementimportseveryclasscontainedinthespecifiedpackage.Forexample,thestatement

importjava.awt.*;

willbringallclassesofjava.awtpackage.

NAMINGCONVENTIONS

ϖPackagecanbenamedusingthestandardjavanaming rules.

ϖNameshouldbeunique,duplicatenamecauserun-timeerror.

ϖPackagebeginswithlowercaseletter.

(Todistinguishwithclassname,classnamebeingswithuppercase.)Exa

mple:

double y = java . lang. Math .

sqrt(x);packageclassmetho

dnamenamename

This statement usesafullyqualifiedclassnameMathtoinvokethemethod**sqrt().**

CREATINGPACKAGES

ϖFirstdeclarethenameofthepackageusingthe**package**keywordfollowedbyapackagename.

ϖThismustbethefirststatementinajavasourcefile.

Example

```
packagefirstpackage;      // package
declarationpublicclassFirstClass          //
classdefinition
        {
        …….
        …….       // bodyofclass
        }
```

ϖPackagenameisfirstpackage.

ϖThis file is saved as a file called FirstClass.java and located in a directory namefirstpackage.

ϖWhensourcefile iscompiled,java willcreatea .classfile andstore itinthe same directory.

ϖThe.classfileshouldinthedirectorythathasthesamenameasthepackage.

Steps:

ϖDeclarethepackageatthebeginningofafileusing theform

;

ϖDefinetheclassthatistobeputinthepackageanddeclareit**public.**

ϖCreateasubdirectoryunderthedirectorywherethemainsourcefilesarestored.

ϖStore thelistingastheclassname.javafileinthesubdirectorycreated.

ϖCompile thefile.Thiscreates**.class**fileinthesubdirectory.

Java supportPackagehierarchy

¬Specifymultiplepackagenamesinapackagestatement,separatedbydots.

E.g.:Packagefirstpackage.secondpackage;

¬Storethepackagesecondpackageinasubdirectoryoffirstpacakge.

ACCESSINGAPACKAGE

Thegeneralformofimportstatementisasfollows

importpackage1[.package2][.package3].classname;

ϖPackage1isthenameofthetoplevelpackage

ϖPackage2isthenameofthepackagethatisinsidethepackage2andsoon.

ϖWecanhaveanynumberof packagesinapackage hierarchy.Finallyexplicitclassnameisspecified.

USINGAPACKAGE

ϖCreatea**classA**underthepackage1;

ϖImportthe**classA**frompackage1inaprogram

ϖCompiletheprogram

ϖDuring compilation, the compiler checks for the file **ClassA.class** in the package1 director forinformationit needs.

ϖDuringrunningtheprogramtheinterpreterloadstheprogramalongwiththecodeofthe **ClassA.class**file.

Example1

```
packagepackage1;p
ublicclassClassA
{
    publicvoiddisplayA()
  {
   System.out.println(“classa”);
  }
 }
```

```
importpackage1.ClassA;class
PackageTest1
 {
   publicstaticvoidmain(Stringargs[])
    {
      ClassAobj=newClassA();o
      bj.displayA();
    }
 }
```

Example2

```
packagepackage2;p
ublicclassClassB
{
   publicvoiddisplayB()
     {
       protected intm=
     10;System.out.println(“classb”);Sy
     stem.out.println(“m=”+m);
     }
 }
```

```
importpackage1.ClassA;import
package2.*;
classPackageTest1
{
publicstaticvoidmain(Stringargs[])
    {
      ClassA
     obj=newClassA();ClassBobjb
     =newClassB();obj.displayA()
     ;obj.displayB();
    }
 }
```

ADDINGCLASSTOAPACKAGE

Itissimpletoaddaclasstoanexistingpackage.

Considerthefollowingpackage:packagep1;

publicClassA

{

// bodyofA

}

The package **p1** contains one public class by name **A**.Suppose we want to add another class **B** to thispackage.This canbedoneas follows:

1. Definethepackageandmakeitpublic.
2. Place the package

   statementpackagep1;

   before the class definition as

   follows:packagep1

   publicclassB

   {

   // bodyofB

   }

3. Storethis as**B.java**fileunderthedirectory**p1**.

DBC

4. Compile **B.javafile.** Thiswillcreatea**B.class**fileandplace itin thedirectory**p1**.

Now,the packagewillcontain bothclassesAand B.Astatement

likeimportp1 . *;

willimportbothofthem.

HIDINGCLASSES

⎫ Tohidecertainclassesfromaccessing fromoutsideofthepackage.Suchclassesshouldbedeclared "notpublic".Example:

```
packagep1;
publicclass X                    //publicclass,availableoutside
{
    // bodyofX
}
classY                           //notpublic,hidden
{
   // bodyofY
}
```

⎫ HeretheclassYwhich isnotdeclaredpublic is hiddenfrom outsideofthepackagep1.

⎫ Thisclasscanbeseenandusedonlybyotherclassedinthesamepackage.

Considerthefollowingcode,which importsthepackagep1thatcontains

classesXandY:importp1. *;

XobjectX; //ok;classXisavailablehere.

YobjectY; //not ok:Yis not available.

Javacompiler generate anerrormessagebecausetheclassYisnotdeclared aspublic.

STATICIMPORT

ϖThe staticimportdeclarationissimilartothatofimport.
ϖTheimportstatementtoimportclassesfrompackagesandusethemwithoutqualifyingthe package.
ϖThe static import statement to import static members from classes and use them withoutqualifyingtheclass name

Syntax:

importstaticpackage-name.subpackage-name.class-name.staticmember-name;(or)
importstaticpackage-name.subpackage-name.class-name.*;

Example:

```
publicinterfaceSalary_increment
    {
        public static final double
        Manager=0.5;publicstaticfinaldoubleCl
        erk=0.25;
     }
```

Toaccesstheinterface,wecanimporttheinterfaceusingthestaticimportstatementasfollows:importstati cemployee.employee_details.Salary_increment;

```
 classSalary_hike
            {
               publicstaticvoidmain(Stringargs[ ])
                 {
                    doublemanager_salary=Manager*Manager_current_salary;
                     doubleclerk_salary=Clerk*Clerk_current_salary;
                      ……..
                      ……………
                  }
            }
```

We can use the static member in the code without qualifying the class name or interface name.Also, the static feature eliminates the redundancy of using the qualified class name with the staticmembernameandincreasesthereadabilityoftheprogram.

QUESTIONS

2Marks

1. Whatdo youmean bysingle-subscriptedvariable?
2. Whatarethestepsinvolved tocreateanarray?
3. Whataretheadvantagesofenumeratedtype?
4. WriteanytwoJavaAPIpackages.
5. Whatarethestandardannotationsusedinjava?

6. Howwill youdefineaninterface?

5Marks

1. ExplainindetailaboutArrays.
2. Describeaboutstringmethods.
3. Explainindetailaboutwrapperclasses.
4. Howwill youimplementinterfacesinjava?
5. Howwillyou addaclasstoapackage?
6. WriteshortnotesonHidingclasses.

10Marks

1. ExplainindetailaboutPackages.
2. WriteshortnotesonInterfaces.

UnitIIIcompleted

DBC

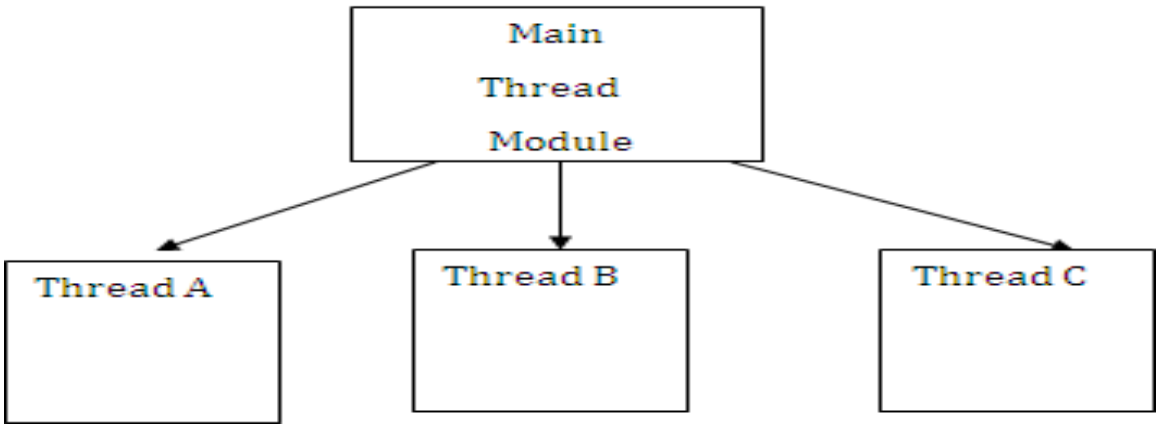
UNIT

IVMULTITHREADEDPROGRAMMI

NG

INTRODUCTION

ϖ**Aflowofcontrolis knownasthread**.
ϖIfa programcontains**multiple flowofcontrols**forachievingconcurrentexecutionthenthat programis**knownasmultithreadedprogram**.
ϖAprogramissaidtobea multi threadedprogramifandonlyifinwhichthereexist‘n’number ofsub-programsthereexistaseparate flowofcontrol.
ϖAllsuchflowofcontrolsareexecutingconcurrentlysuchflowof controlsareknownasthreads and suchtypeofapplicationsorprogramsiscalledmultithreaded programs.
ϖAthreadissimilartoa programthat hasasingle flowofcontrol.Ithasabeginningabody,andan end, andexecutescommands sequentially.



Switching Switching

CREATINGTHREADS

ϖCreatingthreadsinJavaissimple.
ϖThreadsareimplementedintheformofobjectsthatcontainamethodcalledrun().

ϖTherun()methodistheheartandsoulof anythread.Itmakesuptheentire bodyofthe thread and is theonlymethodinwhichthethread'sbehaviorcanbeimplemented.

ϖAtypicalrun()wouldappearasfollows:

```
publicvoidrun()
        {
          ----(Statementsforimplementingthread)
          ---
        }
```

ϖThe run( ) method should be invoked by an object of the concerned thread. This can be achievedbycreatingthethreadand initiatingitwiththehelpofanother threadmethodcalledstart().

ANEWTHREADCANBECREATEDINTWOWAYS.

1. **Bycreatingathreadclass:** Defineaclassthatextends**thread**classandoverrideits **run()** methodwith thecoderequiredbythethread.
2. **By converting a class to a thread**: Define a class that implements **Runnable** interface. The**Runnable** interface has only one method**, run( )** , that is to be defined in the method with thecodeto beexecutedbythethread.

ϖThe approachtobeuseddependsonwhattheclasswearecreatingrequires.

ϖIf it requires to extend another class, then we have no choice but to implement the Runnableinterface,sinceJavaclasses cannot havetwo superclasses.

EXTENDINGTHETHREADCLASS

ϖWecanmake ourclassRunnableasa threadbyextendingthe class**java.lang.Thread**.Thisgivesusaccesstoall thethreadmethodsdirectly.Itincludesthefollowing steps:

1. Declaretheclassasextendingthe**Thread**class.

2. Implementthe **run()**methodthatisresponsiblefor executingthesequenceofcodethat thethreadwill execute.

3. Createathreadobject andcall the **start()**methodto initiatethethreadexecution.

Declaringtheclass

Thethreadclasscanbeextendedasfollows:

classMyThread extends**Thread**
{
……..
……..
}

Nowwehaveanewtypeofthread**MyThread.Imp**

lementingtherun( )method

ϖThe run() methodhasbeeninheritedbytheclass**MyThread.**
ϖWehave tooverride thismethodinordertoimplementthe codetobe executedbyourthread.
ϖThebasicimplementationofrun()willlooklikethis:

publicvoidrun()
{
……..
……..//Threadcodehere
}

ϖwhenwe start thenewthread,Javacallsthe thread'srun()method,soitistherun()wherealltheaction takesplace.

STARTINGNEWTHREAD

ϖToactuallycreateandrun oninstance ofthe threadclass,wemustwritethe following

MyThread aThread = new MyThread(

);aThread.start();//invokesrun()method

ϖThesecondline callsthe start() methodcausingthethreadtomove intothe runnablestate.

ϖThenthe Javaruntime willschedulethe threadtorun byinvokingitsrun( )method.Now,thethreadis said tobeintherunningstate.

AnexampleofusingtheThreadclass

```
classAextendsThread
{
    publicvoidrun()
    {
        for(int i=1;i<=3;i++)
        {
        System.out.println("\tFromthreadA:i="+i);
        }
        System.out.println("ExitfromA");
    }
}
classBextendsThread
{
    publicvoidrun()
    {
    for(inti=1;i<=3;i++)
    {
    System.out.println("\tFromthreadB:i="+i);
    }
    System.out.println("ExitfromB");
    }
}
classThreadTest
{
    publicstaticvoidmain(String[]args)
    {
    newA().start();
    newB().start( );
    }
}
```

Output

FromthreadA:i=1

FromthreadA:i=2

From thread

B:i=1From thread

B:i=2Fromthread

A:i=3ExitfromA

From thread

B:i=3Exitfrom B

ϖThemainthreaddiesatthe endof itsmainmethod.However,before itdies,itcreatesandstarts allthetwothreadsAandB.Notethattheoutput fromthethreads are not speciallysequential.

STOPPINGANDBLOCKINGTHETHREAD

StoppingaThread

ϖWhenever we want to stop a thread from running further, we may do so by calling its stop( )method,like:

aThread.stop();

ϖThis statement causes the thread to move to the **dead** state. The **stop( )** method may be usedwhenthepremature death of athread is desired.

BlockingaThread

ϖA thread can also be temporarily suspended or blocked from entering into the runnable andsubsequentlyrunningstatebyusingeither ofthefollowingthread methods:

sleep() // blocked for a specified

timesuspend( )//blocked until further

orderswait()//blockeduntilcertaincondition

ϖThe thread will return to the runnable state when the specified time is elapsed in the case of**sleep( ),**the**resume()**methodisinvokedinthecaseof**suspend()**andthe**notify()**methodiscalledin thecaseof**wait( ).**

LIFE CYCLEOFATHREAD

ϖDuringthelifetimeofa thread,thereare manystatesitcanenters.Theyinclude:

1. Newbornstate
2. Runnablestate
3. Runningstate
4. Blockedstate
5. Deadstate(Terminated)

start

i l stop

ActiveThread

il

stop

Killedthread

sleep,wait,suspend resume,notify stop

l

Idlethread

STATE TRANSITIONDIAGRAMOFATHREAD

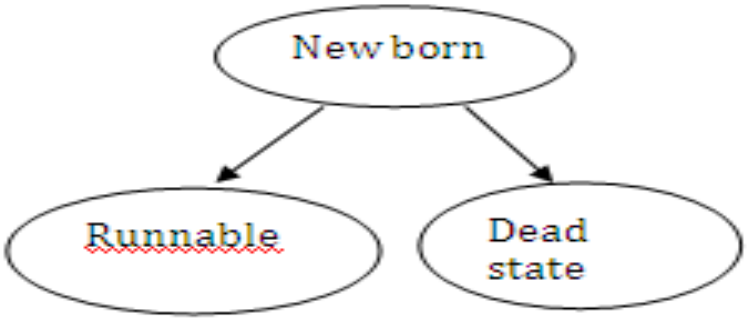
NEWBORNSTATE

ϖWhen we create a thread object, the thread is born and is said to be in newborn state. The threadisnotyetscheduledforrunning.Atthisstate,wecandoonlyoneofthefollowingthingswithit:

¬Scheduleit forrunningusingstart( )method

¬Killitusing stop()method

ϖIf scheduled, it moves to the runnable state. If we attemps to use any other method at this stage,anexceptionwill bethrown.



SCHEDULINGANEW BORN THREAD

RUNNABLESTATE

- ϖThe runnable state means that the thread is ready for execution and is waiting for the availabilityoftheprocessor.Thatis,thethreadhasjoinedthequeueofthreadsthatarewaitingforexecution.
- ϖ Ifallthreadshaveequalpriority,then theyare giventimeslotsforexecutioninroundrobin fashion,i.e.,first-come,first-servemanner.
- ϖThisprocessofassigningtime tothreadsisknownastime-slicing.
- ϖHowever, if we want a thread to relinquish control to another thread of equal priority before itsturncomes, wecando so byusingtheyield()method.

Yield

RunningThread RunnableThreads

Relinquishingcontrolusing yield()method.

RUNNINGSTATE

- ϖRunningmeansthattheprocessorhasgivenitstime tothe threadfor itsexecution.
- ϖThethreadrunsuntilitrelinquishcontrolonitsownoritispreemptedbya higherpriority thread.
- ϖArunningthreadmayrelinquishitscontrol inoneofthefollowing situations.

1 .It has been suspended using suspend( ) method. A suspended thread can be revived by using theresume( ) method. This approach is useful when we want to suspend a thread for some time due tocertainreason, but do not wantto kill it.

. suspend

resume

Running Runnable

SuspendedRelinquishingcontrolusingsuspend(

)method

2. It has been made to sleep. We can put a thread to sleep for a specified time period using themethodsleep(time)
wheretimeisinmilliseconds.Thismeansthatthethreadisoutofqueueduring this time period. The thread re-enters the runnable state as soon as this time period iselapsed.

Sleep(t)

After(t)

Running Runnable

sleepingRelinquishingcontrolusin

gsleep()method

3.Ithasbeentoldtowaituntilsomeeventoccurs.Thisisdoneusingthewait()method.Thethreadcanbescheduled torun again usingthenotify()method.

Wait

notify

Running Runnable Waiting

BLOCKEDSTATE

ϖThe thread is said to be blocked when it is prevented from entering into the runnable state andsubsequentlytherunningstate.

ϖThishappenswhenthe threadissuspended,sleeping,orwaitinginorder tosatisfycertain requirements. A blocked thread is considered " not runnable" but not dead and therefore qualifiedtorun again.

DEADSTATE

ϖEverythreadhasalifecycle.

ϖArunningthreadendsitslifewhenithascompletedexecutingitsrun() method.

ϖItisanaturaldeath.However,we cankillitbysendingthe stopmessagetoitatanystate thus causingprematuredeathtoit.

ϖAthreadcanbe killedassoonasitisborn,orwhile itisrunning,orevenwhenitisin"not runnable"condition.

USINGTHREADMETHODS

Example

```
classAextendsThread
{
                publicvoidrun()
                {
                        for(inti=1;i<5;i++)
                        {
                                if(i==1)yield();
                                System.out.println("\nFromThreadA:i="+i);
                        }
                        System.out.println("ExitfromA");
                }
}
classBextendsThread
{
                publicvoidrun()
                {
                        for(intj=1;j<=5;j++)
                        {
                        System.out.println("\nFromThreadB:j="+j);
```

```
                    if (j==3
                    )stop();
                    }
                    System.out.println("ExitfromB");
            }
}
classCextendsThread
{
            publicvoidrun()
            {
                    for(intk=1;k<=5;k++)
                    {
                            System.out.println("\tFromThreadC:k="+k);if
                            (k===1 )
                            try
                                    {
                                    sleep(1000);
                                    }

                                catch(Exception e)
                    {
                    }
            }
            System.out.println("ExitfromC");
            }
}
classThreadMethods
{
            publicstaticvoidmain(String[]args)
            {
            A threadA =new A(
            );BthreadB =new B(
            );CthreadC=newC( );
            System.out.println("StartthreadA");
```

DBC

```
                threadA.start(
                );System.out.println("Start thread
                B");threadB.start(
                );System.out.println("Start thread
                C");threadC.start( );
                System.out.println("Endofmainthread");
                }
}
```

Output

```
        Start    thread
        AStart  thread
        BStartthreadC
                FromThreadB:j=1
                FromThreadB:j=2
                FromThreadA:i=1
                FromThreadA:i=2
        End ofmainthread
                FromThreadC:k=1
                From          Thread
                B:j=3From  Thread
                A:i=3From  Thread
                A:i=4FromThread
                A:i=5
        ExitfromA
                FromThreadC:k=2
                FromThreadC:k=3
                FromThreadC:k=4
                FromThreadC:k=5
        Exitfrom C
```

THREADEXCEPTIONS

ϖNote that the call to sleep( ) method is enclosed in a try block and followed by a catch block.

Thisisnecessarybecausethesleep()method throwsan exception,whichshouldbecaught.

ϖIfwefailtocatchtheexception,programwillnotcompile.

ϖJavarunsystemwill throwIllegalThreadStateExceptionwheneverweattempt toinvokea methodthatathreadcannothandleinthegivenstate.

ϖForexample,asleepingthreadcannotdealwiththeresume()methodbecausea sleepingthread cannotreceiveanyinstructions.Thesameistruewiththesuspend()methodwhen itis usedon ablocked(not runnable)thread.

ϖWhenever we call a thread method that is likely to throw an exception, we have to supply anappropriateexception handlerto catch it.

THREADPRIORITY

ϖIn Java, each thread is assigned a priority, which affects the order in which it is scheduled forrunning.

ϖThe threads of the same priority are given equal treatment by the Java scheduler and, thereforethey share the processor on a first-come,first-serve basis. Java permits us to set the priority of athreadusingthesetPriority()methodas follows:

ThreadName.setPriority(intnumber);

ϖThe number is an integer value to which the thread's priority is set. The thread class definesseveralpriorityconstants:

MIN_PRIORITY =

1NORM_PRIORITY=

5MAX_PRIORITY=10

ϖThe numbermayassumeoneoftheseconstantsorany valuebetween1and10.Notethatthe defaultsettingisNORM_PRIORITY.

ϖByassigningprioritiestothreads,we canensure thattheyaregiventhe attentiontheydeserve. Whenever multiple threads are ready for execution, the Java system chooses the highest prioritythreadandexecuteit.

ϖRememberthatthehighestprioritythreadalwayspreemptsanylowerprioritythreads.

Example

```
classAextendsThread
{
        publicvoidrun()
        {
            System.out.println("Thread` A
            Started");for(inti=1;i<=3;i++)
            System.out.println("\tFromthreadA:i="+i);S
            ystem.out.println("ExitfromA");
```

```
        }
}
classBextendsThread
{
        publicvoidrun()
        {
        System.out.println("ThreadBStarted");f
        or(inti=1;i<=3;i++)
        System.out.println("\tFromthreadB:j="+j);S
        ystem.out.println("ExitfromB");
        }
}
classCextendsThread
{
        publicvoidrun()
        {
        System.out.println("Thread C
        Started");for(inti=1;i<=3;i++)
        System.out.println("\tFromthreadC:i="+i);S
        ystem.out.println("ExitfromC");
        }
}
classThreadPriority
{
        publicstaticvoidmain(Stringargs[])
        {
        A threadA = new A(
        );BthreadB = new B(
        );CthreadC= newC( );
        threadC.setPriority(
        Thread.MAX_PRIORITY);threadB.setPriority(
        threadA.getPriority( )+1);threadA.setPriority(
        Thread.MIN_PRIORITY);System.out.println("
        Start thread A");threadA.start();
```

```
                System.out.println("StartthreadB");t
                hreadB.start(
                );System.out.println("StartthreadC")
                ;threadC.start( );
                System.out.println("Endofmainthread");
                }
}
```

Output

```
Start     thread
AStart  thread
BStartthreadC
ThreadBStarted
        From          thread
         B:j=1Fromthread
         B:j=2
ThreadCStarted
        From          thread
        C:k=1From  thread
        C:k=2Fromthread
        C:k=3
Exitfrom C
End ofmainthread
        FromthreadB:j=3
Exitfrom B
ThreadAStarted
        FromthreadA:i=1
        FromthreadA:i=2
        FromthreadA:i=3
ExitfromA
```

SYNCHRONIZATION

- In the above examples, we have seen threads that use their own data and methods provided insidetheirrun( )method.
- Whathappenswhenthey trytousedata andmethodsoutsidethemselves.Onsuchoccasions, theymaycompeteforthe sameresourcesandmayleadtoseriousproblems.
- For example, one thread may try to send a record from a file while another is still writing to thesamefile.

DBC

ϖDependingonthe situation,wemaygetstrangeresults.Java enablesustoovercomethisproblem usingatechniqueknown assynchronization.

ϖIn case of Java, the keyword synchronized helps to solve such problems by keeping a watch ansuch locations. For example, the method that will read information from a file and the methodthatwill updatethesamefilemaybedeclaredas synchronized.

Example

```
synchronizedvoidupdate()
 {
   ----// codehereis synchronized
   ---
  }
```

ϖWhen we declare a method synchronized, Java creates a "monitor" and hands it over to thethreadthat calls themethod first time.

ϖAs longasthe threadholdsthe monitor,nootherthreadcanenterthe synchronizedsectionof code.

ϖAmonitoris likeakeyandthethreadthatholdsthekeycanonlyopenthelock.

ϖIt is also possible to mark a block of code as synchronized as shown

```
below:synchronized(lock-object )
{
----// codehereis synchronized
---
}
```

ϖWhenever a thread has completed its work by using synchronized method, it will hand over themonitorto thenext threadthat isreadyto usethesameresource.

ϖAn interestingsituationmayoccurwhentwoor more threadsare waitingtogaincontrolofa resource.

ϖDuetosome reason,the conditiononwhichthewaitingthreadsrelyongaincontroldoesnot happen.

ϖThisresultinwhatisknownasdeadlock.Forexample,assume thatthe threadAmustaccess method1 before it can release method2, but the threadB cannot release method1 until it gets holdon method2.

ϖ Because these are mutually exclusive conditions, a deadlock occurs. The code below illustratethis:

```
ThreadA
synchronizedmethod2( )
{
```

```
        synchronizedmethod1( )
        {
            ……….
        }
    }
    ThreadB
        synchronizedmethod1( )
        {
        synchronizedmethod2( )
        {
            ……….
        }
    }
```

IMPLEMENTINGTHERUNNABLEINTERFACE

ϖTocreate threadusingthe Runnableinterface,wemustperformthe stepslistedbelow:

1. Declarethe classasimplementingtheRunnableinterface.
2. Implementtherun()method.
3. Createathreadbydefiningan objectthatisinstantiatedfromthis“runnable”classasthetargetofthethread.
4. Callthethread’sstart()methodtorunthethread.

ϖIf the direct reference to the thread threadX is not required, then we may use a shortcut as shownbelow:

newThread(newX()).start();

Example

```
classXimplementsRunnable
{
    publicvoidrun()
    {
        for(inti=1;i<3;i++)
          {
              System.out.println(“\tThreadX:”+i);
           }
        System.out.println(“EndofthreadX”);
    }
```

```
}
classRunnableTest
{
    publicstaticvoidmain(String[]args)
    {
        Xrunnable =newX( );
        ThreadthreadX=newThread( runnable);threadX.start();
        System.out.println("Endofmainthread");

    }
}
```

Output

```
EndofmainThreadT
hreadX:1ThreadX:
2
EndofThreadX
```

INTER-THREADCOMMUNICATION

- Inter-threadcommunicationcanbedefinedastheexchangeofmessagesbetweentwoormorethreads.
- Thetransferofmessagestakesplacebeforeorafterthechangeofstateofathread.
- Forexample,anactivethreadmaynotifytoanothersuspendedthreadjustbeforeswitching tothesuspendstate.
- Javaimplementsinter-threadcommunicationwiththehelpoffollowing threemethods.
- **notify():** Resumesthefirstthreadthatwentintothesleepmode.Theobjectclassdeclarationof **notify()method is**shownbelow:

```
finalvoidnotify()
```

- **notifyall():** Resumes all the threads that are in sleep mode.The execution of these threads happensasperpriority.Theobjectclass declarationof**notifyall()method is**shown below:

```
final voidnotifyall()
```

⎫ **Wait():** sendsthecalling threadintothesleepmode.Thisthreadcannowbeactivatedonlyby **notify()ornotifyall()**methods.Theobjectclassdeclaration of**wait()methodis**shownbelow:

```
finalvoidwait()
```

⎫ All the abovemethodsare declaredinthe rootclass,i.e.,Object.Since,the methodsaredeclaredasfinaltheycannotbeoverridden.Allthethreemethodsthrow**InterruptedException**.

MANAGINGERRORSANDEXCEPTIONSI

NTRODUCTION

ϖ Rarelydoesaprogram runsuccessfullyasitsveryfirstattempt. Itiscommontomakemistakes whiledevelopingaswellastypingaprogram.

ϖAmistakemightleadontoanerrorcausingtoprogramtoproduceunexpectedresults.

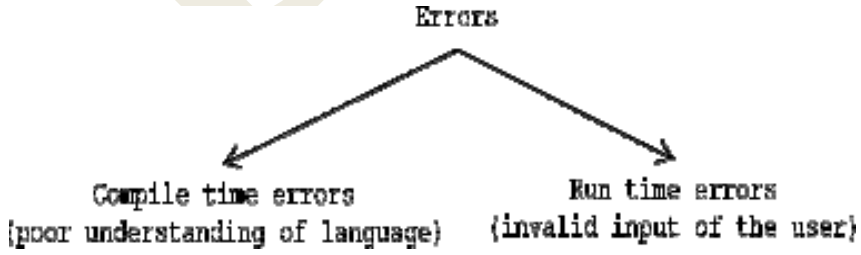
ϖErrorsarethewrongsthatcanmakeaprogramgowrong.

ϖAn error may produce an incorrect output or may terminate the execution of the programabruptlyor evencausethesystem to crash.

ϖItistherefore importanttodetectandmanageproperlyallthepossibleerrorconditionsinthe programsothattheprogram will notterminateorcrashduringexecution.

TYPESOFERRORS

ϖErrorsareof twotypes.Theyare**compile time errors**and**runtimeerrors.**



ϖ**Compile time errors** are those which are occurring because of poor understanding of thelanguage.

ϖ**Run time errors** are those which are occurring in a program when the user inputs invaliddata.

ϖThe run time errors must be always converted by the JAVA programmer into user friendlymessagesbyusingtheconcept ofexceptional handling.

COMPILE-TIMEERRORS

ϖAll syntax errors will be detected and displayed by the Java compiler and therefore these errorsareknown ascompile-timeerrors.

ϖWhenever thecompilerdisplaysanerror,itwillnotcreatethe.classfile.

ϖIt is therefore necessary that we fix all the errors beforewe can successfullycompile adruntheprogram.

ϖMost of thecompile-timeerrorsdue tootypingmistakes.Typographicalerrorsarehard to find.

ϖWemayhavetocheckthecodewordbyword,or evencharacterby character.

ϖ**Themostcommonproblemsare:**

Missingsemicolons
Missingbrackets in classesand methodsMisspelling ofidentifiersandkeywordsMissingdoubleq uotes in strings
Useof undeclaredvariables.
Incompatible types in assignments/initializationBadreferencestoobject s
Useof =in placeof==operator

RUN-TIMEERRORS

ϖSometimes,aprogrammay compile successfullycreating the **.class file** but may notrunproperly.

ϖSuchprogramsmayproducewrongresultsduetowronglogicor mayterminate duetoerrors suchasstack overflow.

ϖ**Mostcommonrun-timeerrorsare**:

¬Dividinganintegerbyzero
¬Accessingan elementthat isoutofbounds ofan array
¬Tryingtostoreavalueintoanarrayofanincompatibleclassor type
¬Tryingtocastaninstanceofa classto oneofitssubclasses.
¬Passingaparameter that is notin avalid rangeorvalueforamethod.
¬Tryingtoillegallychangethestateofathread
¬Attemptingtouseanegativesizeforanarray
¬Usinganull objectreference as alegitimateobject referencetoaccess a methodoravariable.
¬Convertinginvalidstringtoanumber
¬Accessingacharacter that isout ofbounds of astring

Example

```
classExample
{
    publicstaticvoidmain(Stringargs[])
    {
        intd=0;
        inta=42/d;
    }
}
```

When theJavarun-timesystemdetectstheattemptto dividebyzero,itconstructsanewexception object and then throws thisexception.Thiscausestheexecution ofexampletostop,because oncean exception hasbeen thrown,it must becaught by an exception handler anddealtwith immediately. In thisexample, we haven't supplied any exception handlers of ourown, so theexceptioniscaught bythedefault handler providedbytheJavarun-timesystem.

EXCEPTIONS

ϖAn**exception**isaconditionthatiscausedbya runtimeerrorintheprogram.

ϖ**Exceptional handling** is a mechanism of converting system error messages into user friendlymessages.

ϖThepurposeofexceptionhandlingmechanismistoprovideameanstodetectandreportan "exceptionalcircumstance"sothatappropriateaction canbetaken.

ϖThe mechanism suggests incorporation of a separate error handling code that performs thefollowingtasks:

1. **Findtheproblem(Hittheexception)**
2. **Informthatanerrorhasoccurred(Throwtheexception)**
3. **Receivetheerrorinformation(Catchtheexception)**
4. **Takecorrectiveactions(Handletheexceptions)**

ϖWhen writing programs, we must always be an the lookout for places in the program where anexceptioncould begenerated.

ϖSome comeexceptionsthatwe mustwatchoutfor catchingarelistedintable.

EXCEPTIONTYPE	CAUSEOFEXCEPTION
ArithmeticException	Causedbymath errorssuch as divisionbyzero
ArrayIndexOutOfBoundsException	Causedbybadarrayindexed
ArrayStoreException	Causedwhenaprogramtriestostorethewrong typedatainanarray.
FileNotFoundException	Causedbyanattempttoaccessa nonexistentfile
IOException	CausedbygeneralI/O failures,suchas inability to readfromafile.
NullPointerException	Causedbyreferencing anull object
NumberFormatException	Causedwhenaconversionbetweenstringsandnumber fails.
OutOfMemoryException	Causedwhenthere'snot enoughmemorytoallocateanew object.
SecurityException	Caused when an applet tries to perform an actionnotallowed bythebrowser'ssecuritysettings.

StackOverflowException	Causedwhenthesystemrunsoutofstackspace.
StringIndexOutOfBoundsException	Causedwhenaprogramattemptstoaccessa nonexistentcharacterpositioninastring.

CommonJavaExceptions

Exceptionsinjavacanbe categorizedinto twotypes:

- **Checkedexceptions:**These exceptionsareexplicitlyhandledinthecodeitself withthehelpoftry-catchblocks.Checkedexceptions areextended from**java.lang.Exception class.**
- **Uncheckedexceptions:**Theseexceptionsarenotessentiallyhandledintheprogramcode;instead the JVM handles such exceptions.Unchecked exceptions are extended from the**java.lang.RuntimeExceptionclass.**

SYNTAXOFEXCEPTIONHANDLINGCODE

ϖThe basicconceptofexceptionhandlingare throwinganexceptionandcatchingit.

tryblock Statementthatcausesanexception	ExceptionobjectCreator
catchblock Statementthathandlestheexception	Exceptionhandler

- Java uses a keyword **try** to preface of a block of code that is likely to cause an errorconditionand "throw"anexception.
- Acatch blockdefinedbythekeyword**catch"**catches"the exception "thrown"bythee tryblockandhandlesitappropriately.
- Thecatchblockisadded immediatelyafter thetryblock.
- Thefollowingexampleillustratestheuseofsimpletryandcatch statements.

```
……..
…….
try
 {
     statement;//generatesanexception
 }
```

```
catch(Exception-typee)
{
    statement;//processestheexception
 }
…….
……
```

Example

```
classExample

{
publicstaticvoidmain(Stringargs[])
    {
        int
        a=10;int
        b= 5;int
        c=5;intx,
        y;
      try
        {
          x=a/(b-c)
        }
        catch(ArithmeticExceptione)
        {
            System.out.println("Divisionbyzero");
        }
        y=a/ (b+c);System.out.println("y="+y);
    }
}
```

OUT PUT

Divisionbyzeroy=1

MULTIPLECATCHSTATEMENTS

ϖInsomecases,morethanoneexceptioncouldberaisedbyasinglepieceofcode.
ϖTohandle thistype ofsituation,you canspecifytwoormorecatchclause,eachcatchinga differenttypeofexception.S
yntax:

```
……
……
try
```

```
{
    statement;                  //generatesanexception
}
catch(Exception-Type-1e)
{
    statement;                  //processesexceptiontype1
}
catch(Exception-Type-2e)
{
    statement;                  //processesexceptiontype 2

}
.
catch(Exception-Type-Ne)
{
    statement;                  //processesexceptiontype N
}
……
```

- Whenanexceptionin atryblockis generated,thejavatreats themultiplecatchstatementslikecasesinaswitchstatement.
- Thefirststatementwhoseparametermatcheswiththeexceptionobjectwillbeexecuted,and theremainingstatementswillbeskipped.

Example

```
classMulticatch
```

```
{
        publicstaticvoidmain(Stringargs[])
        {
                int a[ ]=(5,10);
                int b =
                 5;try
                        {
                             intx=a[2] /b–a[1];
                        }
                 catch(ArithmeticExceptione)
                        {
                        System.out.println(“Divisionbyzero”);
                        }
                 catch(ArrayIndexOutOfBoundsExceptione)
                        {
                        System.out.println(“ArrayIndexerror”);
                        }
                  catch(ArrayStoreExceptione)
                        {
                        System.out.println(“Wrongdatatype”);
                        }
                            int y = a[1] /
                    a[0];System.out.println(“y=”+
                    y);
}               }
```

Output:

ArrayIndexerrory
=2

USINGFINALLYSTATEMENT

ϖJava supportsanotherstatementknownasfinallystatementthatcanbeused tohandlean exceptionthat isnot caughtbyanyof theprevious statement.

ϖfinallyblockcanbeusedtohandleanyexceptiongeneratedwithinatryblock.

ϖ Itmaybeaddedimmediatelyafter thetryblockorafter thelastcatchblockshown asfollows.

```
try
{
 ---
 ---
}
finally
{
        ---
}
```

```
try
{
      ---
     ---
}
catch(…. )
{
      ----
       -----
}
….
….
finally
{
 ---
  ---
    }
```

ϖwhen a finally block is defined, this is guaranteed to execute, regardless of whether or not anexceptionis thrown.

ϖAsaresult,wecanuse ittoperformcertainhouse-keepingoperationssuchasclosingfilesand releasingsystemresources.

Example

DBC

```
classFinallyDemo
{
        staticvoidproA()
          {
              try
              {
                    System.out.println("InsideProcA");
                }
              finally
              {
                    System.out.println("ProA'sfinally");
              }
           }
        publicstaticvoidmain(Stringargs[] )
              {
              procA();
              }
    }
```

Output

insideprocA
procA's finally

THROWINGOUROWNEXCEPTIONS

Theremaybewhen we would liketothrowourownexceptions.Wecando thisbyusingthekeyword **throw**asfollows:

```
t                    l   ;
```

Example:

```
 importjava.lang.Exception;
classMyException extendsException
  {
      MyException(Stringmessage)
        {
            super(message);
        }
```

```
    }
  classTestMyException
   {
      publicstaticvoidmain(Stringargs[])
       {
           intx=5,y=1000;tr
           y
            {
              floatz=(float)x/(float)y;if(z
               <0.01)
                    {
                       thrownewMyexception("Number istoosmall");
                     }
             }
             catch(MyExceptione)
             {
               System.out.println("Caught my
               exception");System.out.println(e.getMessa
               ge());
             }
           finally
            {
             System.out.println("Iamalwayshere");
            }
    }
 }
```

Output:

Caught my exceptionNumberist oosmallIam alwayshere

DBC

INTRODUCTION

- Applets aresmalljavaprogramthatareprimarilyusedin internet computing.Theycanbetransferred over the internet fromonecomputer to anotherandrun using the **appletviewer**oranyweb browserthatsupport thejavaprogram.
- An applet can perform arithmetic operations, play sounds, display graphics, accept userinput, createanimation and playinteractivegames.
- Javaapplicationsaregenerallyrunfromacommand-linepromptusingJDK.Appletsare runon anybrowsersupportingJava.
- For anapplet torunitmustbeincludedinaweb pageusingHTMLpages.
- Whenabrowserloadsawebpageincludinganapplet,thebrowserdownloadstheappletfromthew eb server andruns iton theweb owner'ssystem.
- Javainterpreterisnotrequiredspecificallyfordoingsoitis alreadybuilt-inthebrowser.

LocalApplet:

- Anappletwhichisdevelopedlocallyandstored inthelocalsystemisknown asthe localappletWhentheweb pageis tryingtofind thelocal applet.
- Itdoes'needtousetheinternetandthereforethelocalsystemdoesnotrequirethe internetconnection.Itsimplysearchesthedirectoriesinthelocalsystem andlocatesandloadthespe cifiedapplet.
- Inordertolocateandloadthelocalappletwemustknowtheappletaddressontheweb page.
- ThisaddressisknownastheURLuniformresourcelocator.andmustbespecifiedin the applet HTMLdocument as thevalueofthecodebaseattribute.

RemoteApplet:

- Itisstoredonaremotecomputerwhichisconnected tothenet. Ifconnectedwiththenet ,wecan downloadthe remoteapplet ontooursystem.
- Wecan utilizeitviatheinternet.

URL:

- UniformResourceLocator.Itspecifiestheappletsaddress.

DIFFERENCEBETWEENAPPLETSANDAPPLICATIONS

¬Applets do not use the main() method for initiating the execution of the code.Applets, whenloaded,automaticallycallcertainmethodsofappletclasstostartandexecutetheappletcode.

¬Unlikestand-aloneapplications,applets cannotrunindependently.Theyrunfrominsideaweb pageusingaspecialfeatureknownasHTMLtag.

¬Applets cannotreadfromorwritetothefilesinthelocalcomputer.

¬Applets cannotcommunicatewithotherserversonthenetwork.

¬Applets cannotrunanyprogramfromthelocalcomputer.

¬Applets arerestrictedfromusinglibrariesfromotherlanguagessuch asCorC++.

Alltheserestrictionsandlimitationsareplacedin theinterestofsecurityof systems.Theserestrictionsensurethatan applet cannot doanydamageto thelocal system.

WRITEAPPLETS

Thefollowingstepsinvolvedindevelopingandtestinginappletare:

1. Buildinganappletcode(.javafile).
2. Creatinganexecutableapplet(.classfile).
3. DesigningaWebpageusingHTMLtags.
4. Preparing<APPLET>tag.
5. Incorporating<APPLET>tagintotheWebpage.
6. Creating HTMLfile.
7. Testingtheappletcode.

BUILDINGAPPLETCODE

- Applet code uses the service of two classes namely, **Applet** and **Graphics** from the Javalibrary. The **applet** class is contained in the **java.applet** package provides life and behavior to theappletthroughits methodssuchas**init(),start(),andpaint().**

- The Applet class maintains the lifecycle of an applet. The **paint()** method of the appletclass actually displays the result of the applet code on the screen. The output may be text, **graphics**objectas anargument,isdefined asfollows:

i i

The applet code general format as follows:importjava.awt.*;

importjava.applet.*;

publicclassappletclassnameextendsApplet

{

……..

…….

publicvoid paint(Graphics g)

{

…….

…….

…….

}

……

……

}

Example:

```
import
java.awt.*;importjav
a.applet.*;
publicclasshelloworldextendsapplet
{
publicvoidpaint(Graphicsg)
{
g.drawstring“HelloJava”, 10,100);
}
}
```

Chainofclassesinheritedbyappletclass

java.lang.objectjava

.awt.componentjava

.awt.containerjava.a

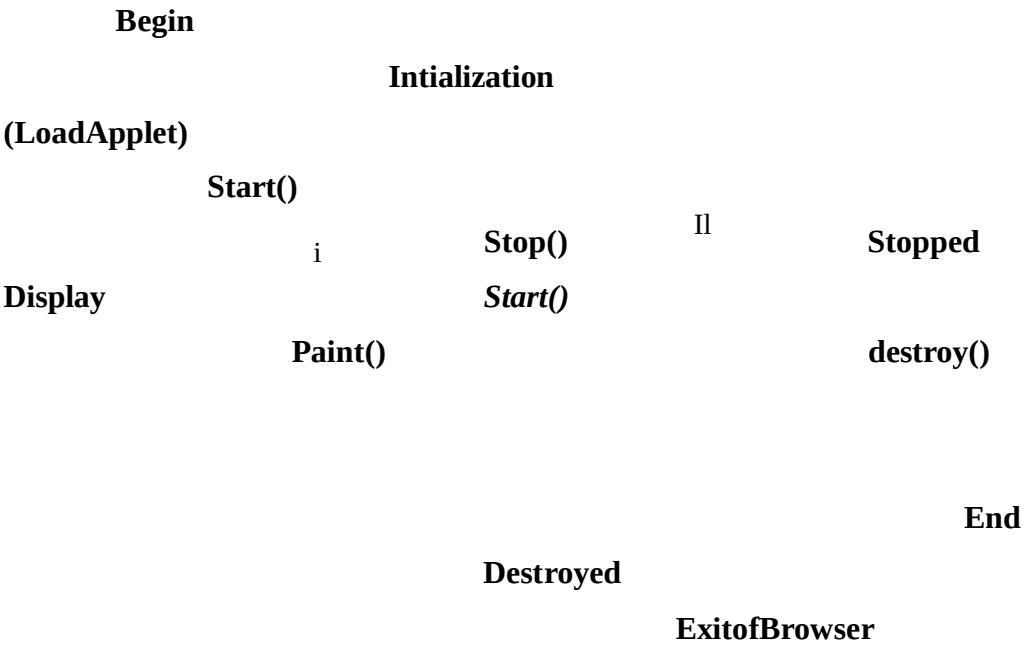
wt.pannel

java.applet.Applet

APPLETLIFECYCLE

- Everyjava applet inherit a set a default behavior from the Appletclass. Astheresultapplet is loaded,it undergoes a series of changes in its stateasshown in the above figure the appletstateswill be

* Bornorinitialization state
* Runningstate
* Idlestate
* Deadordestroyedstate



InitializationState:

Itisachievedby callinginit()methodoftheappletclass. Wecanperformthefunctions like

- Createobjectsneededbytheapplet
- Setupinitial values
- Load imagesorfonts
- SetupcolorsThi

smethodoccursonlyonce

Syntax:

```
publicvoid init()
{
```

……..(Action)

}

Runningstate:

- Applet enter the running state when the system calls the start() method of the applet class.Thisoccurs automatically aftertheappletis initialized.
- Startingcanalsoiftheappletisalreadyintheidlestoppedstate.Thestart()method maybecalledbymorethanonetime.

publicvoidstart()

{

……(Action)

}

StoppedOrIdlestate:

- Anappletbecomesidlewhenitisstoppedfromrunning.Stoppingoccursautomaticallywhenwel eavethepagecontaining thecurrentlyrunningapplet.
- Wecanalsodosobycallingthestop()methodexplicitly.Ifweusethethreadtorunthe appletthatwemust usethestop()method toterminatethethread.
- Wecanachievebyoverriddingthestop()method.pub

licvoidstop()

{

…….(Action)

}

DeadState

- An applet issaidtobedeadwhenitisremoved fromthe memory.This occursautomaticallybyinvoking thedestroy()method.
- Whenwequitthebrowser. Liketheinitializationdestroyingstageoccuronlyonceinthe appletlifecycle.

publicvoiddestroy()

{

.....(Action)

}

DisplayState

- Appletmovestothedisplaystatewheneverithastoperformsomeoutputoperationonthescreen.This happensimmediatelyaftertheapplet entersinto therunningstate.
- Thepaint()method iscalledtoaccomplishthistask.Almosteveryappletwillhavea paint()methodlikeothermethodinthelifecycle.

```
publicvoidpaint(Graphicsg)
  {
      …….(Displaystatements)
    }
```

CREATINGANEXECUTABLEAPPLET

Letusconsiderthe HelloJavaappletcreated.Thisapplethasbeen storedin afilecalledHelloJava.java.Hereare thesteps requiredforcompilingtheHelloJavaapplet.

1. Movetothedirectorycontaining thesourcecodeandtypethefollowingcommandjavac HelloJava.java
2. Thecompiledo/pfilecalledHelloJava.classisplacedinthesamedirectoryasthesource.
3. Ifanyerrormessageisreceived,thenwemust checkforerrorscorrect themandcompiletheappletagain.

DESIGININGAWEBPAGE

- AwebpageisbasicallymadeupoftextandHTMLtags.ItisalsoknownasHTMLpageordocument.
- AwebpageismarkedbyanopeningHTMLtag<HTML>anda closingtag</HTML>
- It isdividedintothreemajor sections.

1. Commentsection(optional)
2. Headsection(optional)
3. Bodysection

HTMLtags format:

<HTML>

<!
.....
>

CommentSection

<HEAD>
Titletag
</HEAD>

HeadSection

<BODY>
Applettag
</BODY>

BodySection

</HTML>

1. CommentSection:

- This sectioncontainscommentsaboutthewebpage.
- It tells whatisgoingonthewebpage.
- Thecommentlinebegins with<!And endswith>

2. HeadSection:

- This sessioncontainsthetitleforthewebpage.Starting <HEAD>and endingwith</HEAD>

<HEAD>
<TITLE>WELCOME TOJAVAAPPLETS</TITLE>
</HEAD>

3.BodySection:

- This sectioncontainstheentireinformationaboutthewebpage

<BODY>
<CENTER>
<H1>APPLETS</H1>

</CENTER>

```
<APPLET
  CODE=
"HELLO.CLASS"WIDTH=
300HEIGHT=200>
</APPLET>
</BODY>
```

APPLETTAG

The<APPLET…>tagsuppliesthenameoftheapplettobeloaded andtellsthebrowserhowmuchspacetheapplet requires.

Example:

```
<APPLET
    CODE=helloJava.classW
    IDTH=400
     HEIGHT=200>
  </APPLET>
```

Notethat<APPLET>tag discussedabovespecifies threethings:

1. Nameoftheapplet.
2. Widthoftheapplet.(inpixels)
3. Heightoftheapplet.(inpixels)

ADDINGAPPLETTOHTMLFILE

```
<HTML>
<HEAD>
    <TITLE>WELCOME TOJAVAAPPLETS</TITLE>
</HEAD>
<BODY>
<CENTER>
    <H1>APPLETS</H1>
<BR>
</CENTER>
<APPLET
  CODE="HelloJava.class"
WIDTH=300
```

HEIGHT=200>

</APPLET>

</BODY>

</HTML>

Wemustnamethis fileasHelloJava.htmlandsaveitinthesamedirectoryasthecompiledapplet.

RUNNINGTHEAPPLET

- WemusthavethefollowingfilesinourcurrentdirectoryHell

  oJava.java

  HelloJava.class

  HelloJava.html

- Torunanappletwerequireoneofthefollowingtools:
  1. Java-enabled web browser(suchasHotJavaorNetscape)
  2. Javaappletviewer

- IfweuseaJava-

  enabledwebbrowser,wewillbeabletoseetheentirewebpagecontainingtheapplet.
- If weusethe appletviewertool,wewillonlyseetheapplet
  output.Ex:appletviewerHellojava.html

  l i ll .ll
  ll

  l l .

APPLETTAGS

Thesyntaxofthe<APPLET>taginfullformisshownasfollows:

<APPLET

[CODEBASE =

codebase_URL]CODE=

AppletFileName.class[ALT=alte

rnate_text]

[NAME=applet_instance_name]

WIDTH=pixels

HEIGHT =

pixels[ALIGN =

alignment][VSPACE

=

pixels][HSPACE=pix

els]

>

[<PARAMNAME=name1VALUE=value1>][<P

ARAMNAME=name2VALUE=value2>]

…….

[TexttobedisplayedintheabsenceofJava]

</APPLET>

AttributesofAPPLETTag

Attribute	MeaningSpeci
CODE=AppletFileName.classC	fiesthenameoftheapplet to belaoded.
ODEBASE=codebase_URL	Specifies the URL of the directory in which the applet
WIDTH=pixels	resides.Theseattributesspecifythewidthand heightofthespaceon
HEIGHT=pixels	the
NAME=applet_instance_name	HTMLpagethatwill bereservedfortheapplet.
ALIGN=alignment	Anameforthe applet mayoptionallybespecified.
	This optional attribute specifies where on the page the applet
	willappear.Possible valuesfor
HSPACE =	alignmentare:TOP,BOTTOM,LEFT,
	RIGHT,MIDDLEETC.
pixelsVSPACE =	Thisattributespecifiestheamountofhorizontalblankspacethe
	browsershouldleavesurroundingtheapplet.
pixelsALT=alternate_t	Thisattributespecifiestheamountofverticalblankspacethebrowser
	shouldleavesurrounding theapplet.
ext	Non-javabrowserswilldisplaythistextwherethe
	appletwouldnormallygo.This attributeis optional.

PASSINGPARAMETERSTOAPPLETS

¬Wecansupplyuser-definedparametersto anappletusing<PARAM..>tags.
¬Each<PARAM..>taghasanameattributesuch ascolor,and avalueattributesuchas red.
¬Forexample, wecanchangethecolor ofthetextdisplayed to redbyanapplet byusinga <PARAM..>tagasfollows:

```
<APPLET>
 <PARAM=colorVALUE="red">
 </APPLET>
```

Tosetupand handleparameters, weneedtodotwothings:

1. Include appropriate<PARAM..> tagsinthe HTMLdocument.
2. Provide code in the applet to parse these parameters.Example:

AppletHelloJavaParam

```
import

java.awt.*;importjav

a.applet.*;

publicclassHelloJavaParamextendsApplet

{

        Stringstr;

         publicvoidinit( )

        {

        str=

       getParameter("string");if

       (str==null)

              str="Java";

               str="Hello"+str;

         }

         publicvoidpaint(Graphicsg)

       {

            g.drawString(str,10,100);
```

```
        }
    }
```

NowletuscreateHTMLfilethatcontainsthisapplet.

TheHTMLfileforHelloJavaParamapplet

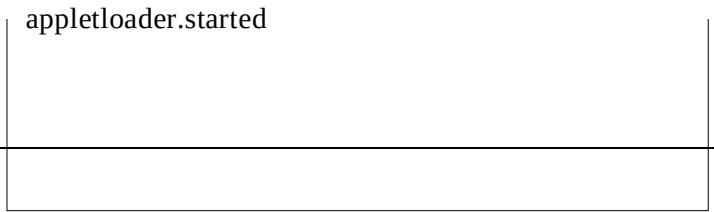
```
<HTML>
        <!parameterizedHTMLfile>
    <HEAD>
    <TITLE>WelcometoJavaApplets</TITLE>
     </HEAD>
    <BODY>
      <APPLETCODE=HelloJavaParam.class
            WIDTH=400
             HEIGHT=200>
            <PARAMNAME="string"
                VALUE="Applet!">
         </APPLET>
    </BODY>
  </HTML>
```

Savethisfileas**HelloJavaParam.html**andthen runtheappletusingtheappletviewer as follows:

AppletviewerHelloJavaParam.html

This willproducetheresultasshownbelow:

l i ll .ll
ll l



APPLETTAGS

Now,removethe<PARAM>tagfromtheHTMLfileandthenruntheappletagain.Theresultwillbeas shown below

l i ll .ll
ll

l l .

APPLETTAGS

ALIGNINGTHEDISPLAY

We can align the output of the applet using the ALIGN attribute.This attribute can have one oftheninevalues:

LEFT,RIGHT,TOP,TEXTTOP,MIDDLE,ABSMIDDLE, BASELINE,BOTTOM,ABSBOTTOM

ForexampleALGN=LEFTwilldisplaytheoutputattheleftmargin ofthepage.Example:

```
<HTML>
    <HEAD>
        <TITLE>Hereisanapplet</TITLE>
    </HEAD>
    <BODY>
```

```
<APPLETCODE=HelloJava.class
 WIDTH=400
 HEIGHT =
 200ALIGN=RIGH
 T>
</APPLET>
</BODY>
</HTML>
```

applet

Text

An applet aligned rightMOREABOUTHTMLTAGS

HTMLTagsand TheirFunctions

Tag		**Function**
<HTML> … </HTML>	…	Indicatesstarting&endingof aHTMLfile
<HEAD> … </HEAD>	…	Containsthedetailsofthewebpage
<TITLE> … </TITLE>	…	Containsthetitleofthebrowser
<BODY> ….</BODY>		It containsthemaintext
<H1> … </H1>	…	Itcontainstheheadingtag,where1to7are itssize

…..	H1ⓅLargestFont
<H7> …… </H7>	H7ⓅSmallest Font
<CENTER> …. </CENTER>	PlacesthetextintheCenterPosition
<APPLET> … </APPLET>	Indicates the Applet tag for graphicalcreations
<PARAM….>	Userdefinedparameters
<B>…</B>	Boldface
<U>…</U>	Underline
<I>….</I>	Italics
 	IndicatesBreak intextposition
<P>	IndicatestheParagraph tag
<IMG….>	Used fortheinsertion oftheimage
<HR>	Drawsthehorizontalruler
<A… >…. </A>	Indicates the Anchor tag (i.e) Hyperreference
<FONT> …. </FONT>	IndicatesthesettingofFont,Size,Color…..
<!..>	Indicatesthecomment position

DISPLAYINGNUMERICALVALUES

¬In applets, we can display numerical values by first converting them into strings and thenusingthedrawstring()method ofGraphics class.

¬Wecando this easilybycallingtheValueOf()methodofStringclass.importjava.awt.*;
importjava.applet.*;
publicclassNumValuesextendsApplet
{
publicvoid paint(Graphics g)
{
int value1 =
10;intvalue2=2
0;
intsum=value1+value2;

```
            Strings="sum:"+String.valueOf(sum);g.d
            rawString(s,100,100);
        }
    }
    <html>
    <applet
        Code =
          Numvalues.classWidth
          =300
           Height=300>
     </applet>
     </html>
```

Output:

l i l l
l

l l .

GETTINGINPUTFROMTHEUSER

- Appletworksin agraphicalenvironment.
- Applettreatinputasastring.
- Wemust firstcreateanareaofthescreen.
- Thisisdobyusing theTextFieldclassoftheappletpackage.
- Oncetext fields arecreatedfor receivinginput,wecantypethevaluesin the fieldsandeditthem.
- NextstepistoretrievetheitemsfromthefieldsExa

mple:

```
import

java.awt.*;importjav

a.applet.*;

publicclassUserInextendsApplet;

{

TextFieldtext1,text2;

publicvoidinit()

{

text1=new

TextField(8);text2=new

TextField(8);add(text1);
```

```
add(text2);text1.s
ettext(“0”);
text2.settext(“0”);
}
publicvoid paint(Graphics g)
{
int
x=0,y=0,z=0;Str
ings1,s2,s;
g.drawstring :Input a number in each
box”,10,50);try
{
s1=text1.gettext();x=int
eger,parseINT(S1);s2=te
xt2.getText();y=Integer.
parseint(s2);
}
catch (Exception
ex){}z=x+y;s=String.val
ueof(z);
g.drawstring(“Thesumis”,10,75);g.
drawstring(s,100,75);
}
publicBooleanaction(Eventevent,objectobject)
{
repaint();re
turntrue;
}
```

DBC

}

RuntheappletUserInusingthefollowingsteps:1.T

ype and save the program(.java

file)2.Compiletheapplet(.class file)

3.WriteaHTMLdocument(.htmlfile)

<html>

<applet

code

=userIn.classwidt

h=300

height=200>

</applet>

</html>

4.Usetheappletviewertodisplaytheresult.

l i I.l
l

I i
i

l l .

EVENTHANDLING

ActionEventis triggeredwheneverauserinterfaceelementisactivated,suchasselectionofamenuitem.

- **ItemEvent**istriggeredattheselectionordeselectionofanitemizedorlistelement,suchascheckbox.
- **TextEvent**istriggeredwhenatextfieldismodified.
- **WindowEvent**istriggeredwheneverawindow-relatedoperationisperformed,suchasclosing oractivatingawindow.
- **KeyEvent**istriggeredwheneverakeyispressedonthekeyboard.

EventSources

- The registrationof alistenerobjectwithaneventensuresthatonoccurrence of theevent,thecorrespondinglistener objectisnotifiedfortakingappropriateaction.

Followingisthesyntaxforregisteringalistenerforanevent.

```
publicvoidadd<Type>Listener(<Type>ListenerEveList)
```

EventListeners

- Theeventlistenerobjectcontainsmethodsforreceivingandprocessingeventnotificationsse ntbythesourceobject.
- Thesemethodsareimplementedfromthecorrespondinglistenerinterfacecontainedinthe**java. awt.event**package.

EventClasses

- Alltheeventsinjavacorrespondingeventclassesassociatedwiththem.
- Eachoftheseclassesisderivedfromonesinglesuperclass,i.e**.,EventObject**.
- Itiscontainedinthe**java.util.package**.
- The**EventObject**classcontainsthefollowingtwoimportantmethodsfor handlingevents:
  - **getSource():**Returnstheeventsource**.**
  - **toString():**Returnsastringcontaininginformation abouttheeventsource.

QUESTIONS

2 Marks

1. Whatarethetwowaysto createa newthread?
2. Definemultithreading.
3. DefineException.
4. WhatarethetwotypesofExceptions?
5. Writedownthesyntaxformultiplecatchstatements.
6. Howwill youbuild anappletcode?
7. Whatarethekeyevents usedinjava?
8. Whatdoyoumeanbylocaland remoteapplet?

5Marks

1. Howwill you extendthethreadclass?
2. Explainaboutthreadpriority.
3. Whatarethetwotypesoferrors?Explainindetail.
4. Explainindetailabouttryandcatch mechanism.
5. Howwill youdesign awebpage?
6. Whatarethe attributes usedinapplettag?
7. Howwillyoupassparameterstoapplets?
8. Write shortnoteson adding appletto HTMLfile.

10Marks

1. Explainindetailabout LifecycleofThread.
2. DescribeindetailaboutAppletlifecycle.

Unit IVcomple ted

DBC

DBC

DBC

UNITVGRAPHICSPROGRAMMI

NGINTRODUCTION

- Oneofthemostimportant featuresofJavaisitsabilityto drawgraphics.
- We can write java applets that draw lines, figures of different shapes, images, and text in different fontsandstyles.
- Wecanalsoincorporatedifferentcoloursindisplay.
- Everyapplethasitsownareaofthescreenknownascanvas,whereitcreatesitsdisplay.
- AJavaappletdrawsgraphicalimageinsideitsspaceusingthecoordinatesystem.
- Java'scoordinatesystemhastheorigin(0,0)intheupper-leftcorner.
- Positivexvaluesaretotheright,andpositiveyvaluesaretothebottom.
- The valuesofcoordinatesxandyareinpixels.

THEGRAPHICSCLASS

- Java'sgraphicsclassincludesmethodsfor drawingmanydifferenttypesofshapes.
- To drawa shapeonthescreen,wemaycalloneofthemethodsavailableinthe**Graphics**class.
- Thefollowingarethemostcommonlyusedmethodsin**Graphics**class.

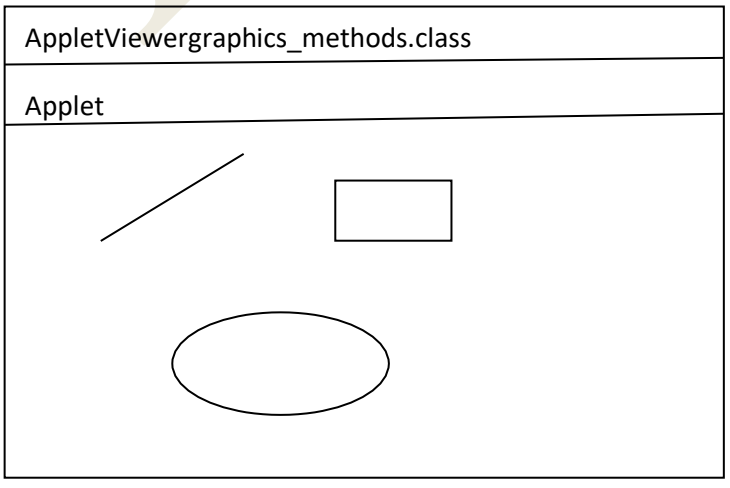
Drawingmethodsofthegraphicsclass

Method	Description
clearRect()	Erasesarectangular areaofthecanvas.
copyArea()	Copiesarectangulararea ofthecanvastoanotherarea.
drawArc( )	Drawsahollowarc.
drawLine()	Drawsastraightline.
drawOval()	Drawsahollowoval.
drawPolygon( )	Drawsahollowpolygon.
drawRect()	Drawsahollowrectangle.
drawRoundRect()	Drawsahollow rectanglewithroundedcorners.
drawstring()	Displaysatextstring.
fillArc()	Drawsafilledarc.
fillOval()	Drawsafilledoval.
fillPolygon()	Drawsafilled polygon

fillRect()	Drawsafilledrectangle.
frillRoundRect()	Drawsafilled rectanglewithroundedcorners.
getcolor()	Retrievesthecurrentdrawingcolour.
getFont()	Retrievesthecurrentlyusedfont.
getFontMetrics()	Retrieves informationaboutthecurrentfont.
setColor()	Setsthedrawingcolour.
setFont()	Setsthefont.

Example:

```
<html>
<body>
<appletcode=graphics_methods.classwidth=200height=200>
</applet>
</body>
</html>
importjava.awt.*;im
portjava.applet.*;
publicclassgraphics_methodsextendsApplet
{
publicvoidpaint(GraphicsGA)
{
GA.drawRect(160,5,60,60);GA.
drawLine(380,100,200,180);GA
drawOval(10,120,155,95);
}
}
```



LINESANDRECTANGLES

- Thesimplestshapewecandrawwith Graphicsclassisaline.
- The**drawline()**methodtakestwopair ofcoordinates(x1,y1) and(x2,y2).

Ex:

```
g.drawLine(10,10,50,50);
```

- Thegisthe**Graphics**objectpassedto **paint()** method.
- Wecandrawarectangleusingthedraw Rect()method.
- This methodtakesfourarguments.
- Thefirsttworepresentthexandy coordinatesoftheleftcorneroftherectangle,andtheremainingtworepresentthewidthandtheheightofthe rectangle.

Ex:

```
g.drawRect(10,60,40,30)
```

- Wecandraw asolidboxbyusingthemethodfillRect()

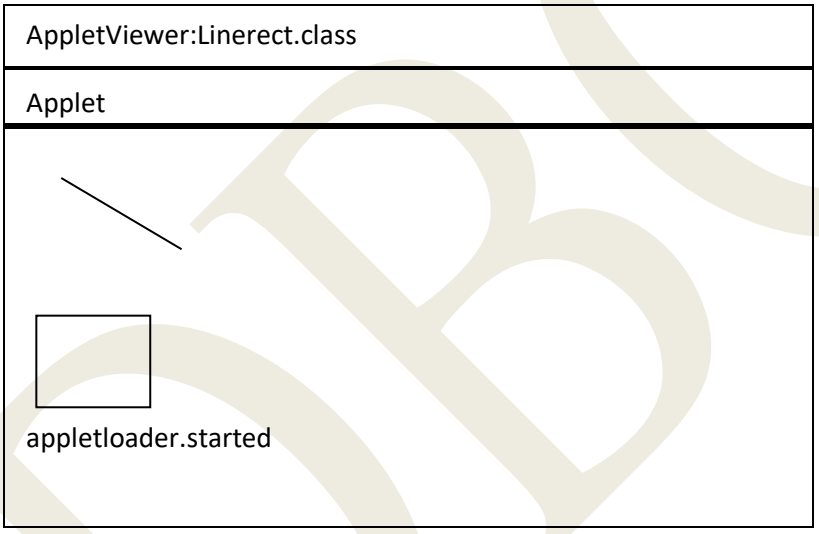
Ex:

```
g.fillRect(60,10,30,80)
```

Example:

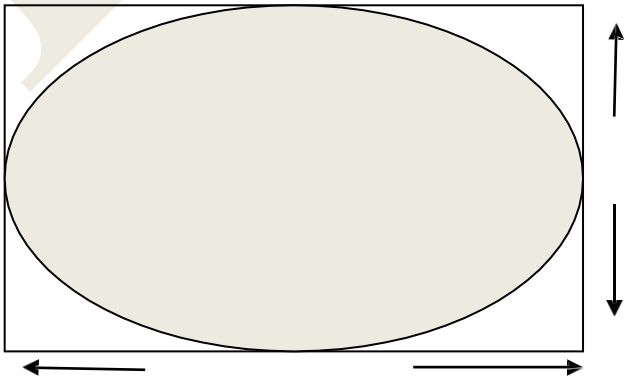
```
importjava.awt.*;imp
ortjava.Applet.*;
publicclassLineRectextendsApplet
{
publicvoid pain(Graphicsg)
{
g.drawLine(10,10,50,50);
g.drawRect(10,60,40,30);
```

```
}

}

<Applet

Code =

LineRect.classWidth=

250

Height=200>

</Applet>
```



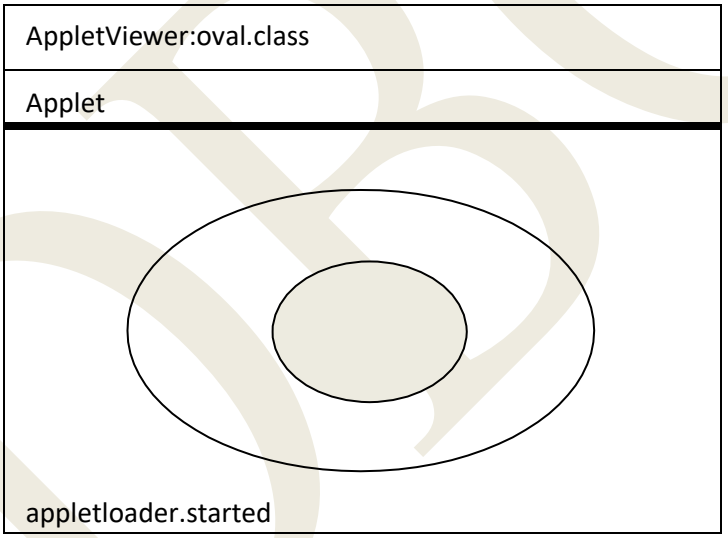
CIRCLESANDELLIPSES

ItisachievedbyusingdrawOval()andfillOval()



Ex:

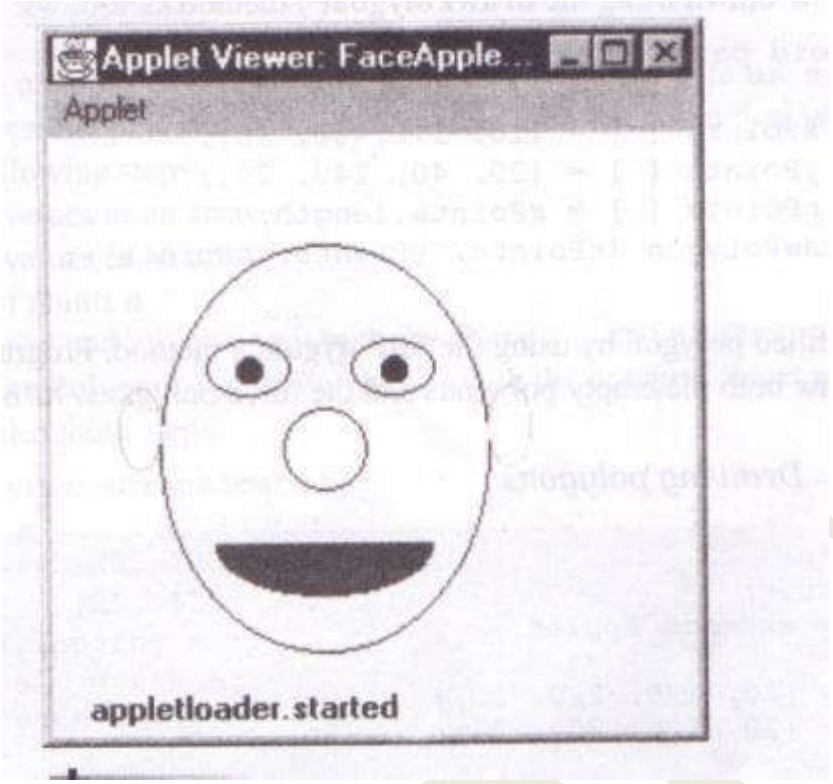
```
importjava.awt.*;im

portjava.applet.*;

/*<Appletcode="cir.class"height=100width=200>

</Applet>*/

publicclasscirextendsApplet
{
        publicvoidpaint(Graphicsg)
        {
                g.drawOval(20,20,200,120);
                g.setColor(color.Green);g.fil
                lOval(70,30,100,100);
        }
```



DRAWINGARCS

- ✓ Anarcisapart ofanoval.
- ✓ The**drawArc( )**designedtodrawarcstakessixarguments.
- ✓ The first four of are the same as arguments for **drawOval( )**method and the last tworepresentthe startingangleofthearcandthenumberofdegreesaround thearc.
- ✓ The**fillArc()** methodtofillthearc.

```
import
 java.awt.*;importjav
 a.applet.*;
 publicclassFaceextendsApplet
   {
       publicvoidpaint (Graphicsg)
        {
                    g.drawOval(40,40,120,150);
                   g.drawOval(57,75,30,20);
                   g.drawOval(110,75,30,20);
                   g.fillOval(68, 81, 10,10);
                   g.fillOval(121, 81,10,10);
                   g.drawOval(85,100,30,30);
                   g.fillArc(60,125,80,40,180,180);
                   g.drawOval(25,92,15,30);
                   g.drawOval(160,92,15,30);
        }
}
```



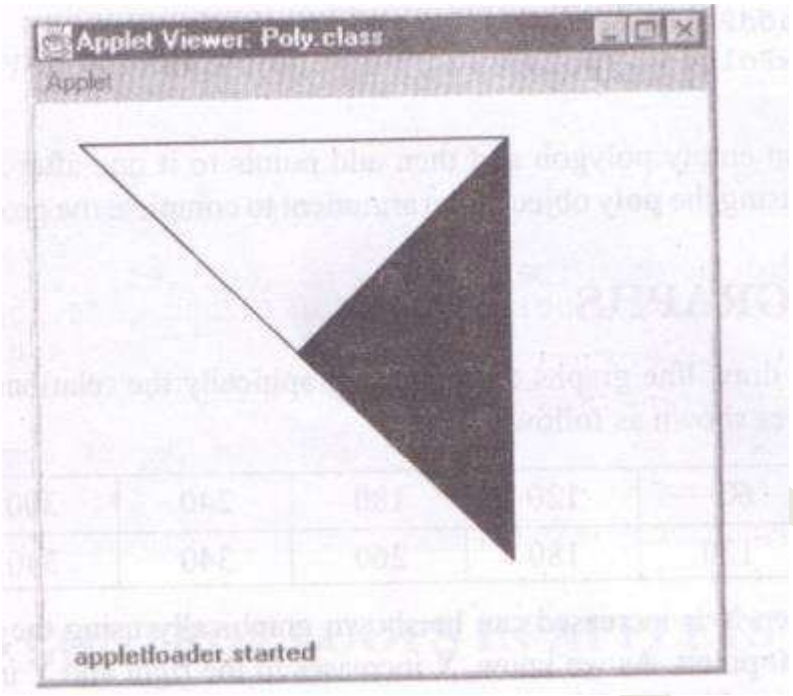
DRAWINGPOLYGONS

The**drawPolygon( )**methodtakes3arguments.

- ✓ Anarrayofintegers containing x coordinates
- ✓ Anarrayofintegers containingycoordinates
- ✓ An integer for the total number of

pointsThe**fillPolygon(**

)methodusedtodrawafilledpolygon.



Ex:

```
importjava.awt.*;im

portjava.applet.*;

publicclassPolyextendsApplet

{

    intx1[]={20,120, 220,20};
    inty1[] ={20, 120,20,20};
    intn1=4;
    intx2[]={120,220,220,120};
    inty2[]={120,20,220,120};
    intn2=4;
   public voidpaint(Graphicsg)
      {
          g.drawPolygon(x1,y1,n1);g
          .fillPolygon(x2,y2,n2);
       }
  }
```

LINEGRAPHS

We can design applets to draw line graphs to illustrate graphically the relationship between twovariables.

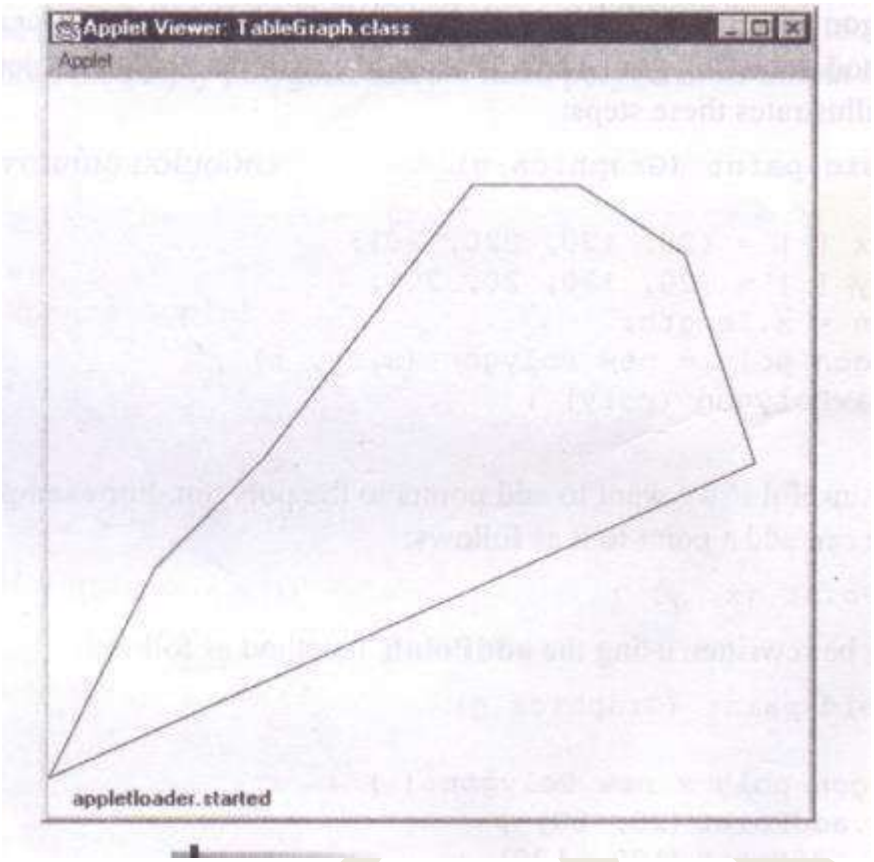
Considerthetableofvaluesshownasfollows:

X	0	60	120	180	240	300	360	400
Y	400	280	220	140	60	60	100	220

Ex:

```
importjava.awt.*;im
portjava.applet.*;
publicclassTableGraphextendsApplet
{
    intx []={ 0,60,120,180,240,300,360,400}
    inty[]={400,280,220,140,60,60,100, 220};
    intn=x.length;
    publicvoidpaint(Graphicsg)
    {
        g.drawPolygon(x,y,n);
    }
}
```

DBC



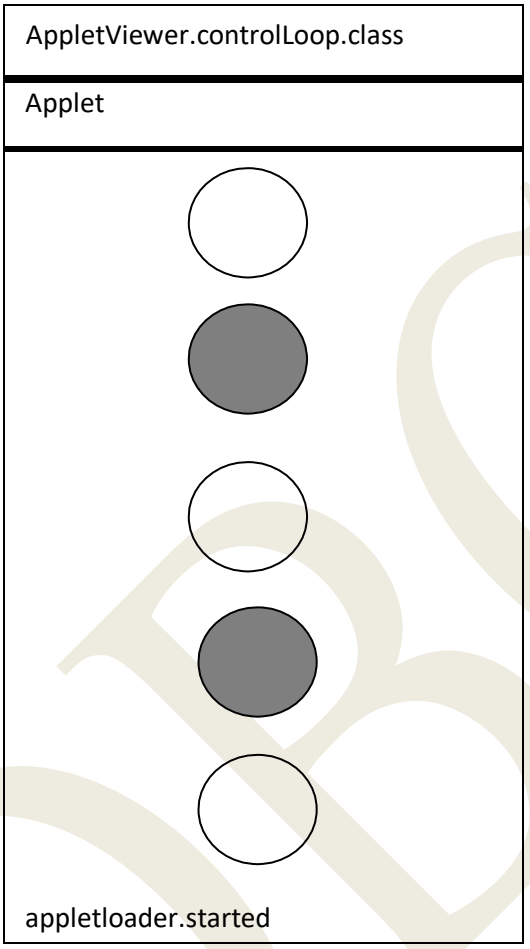
USINGCONTROLLOOPSINAPPLETS

- Wecanuseallcontrolstructures inanapplet.
- Theprogramusesa **for**loopfordrawingcirclesrepeatedly.

Example:

```
importjava.awt.*;im
portjava.applet.*;
publicclassControlLoopextendsApplet
{
publicvoidpaint(Graphicsg)
{
for(inti=0;i<=4;i++)
{if(i%2)==
0
g.drawOval(120,i*60+10,50,50);
```

```
else
g.fillOval(120,1*60+10,50,50);
}
}
}
```



DRAWINGBARCHARTS

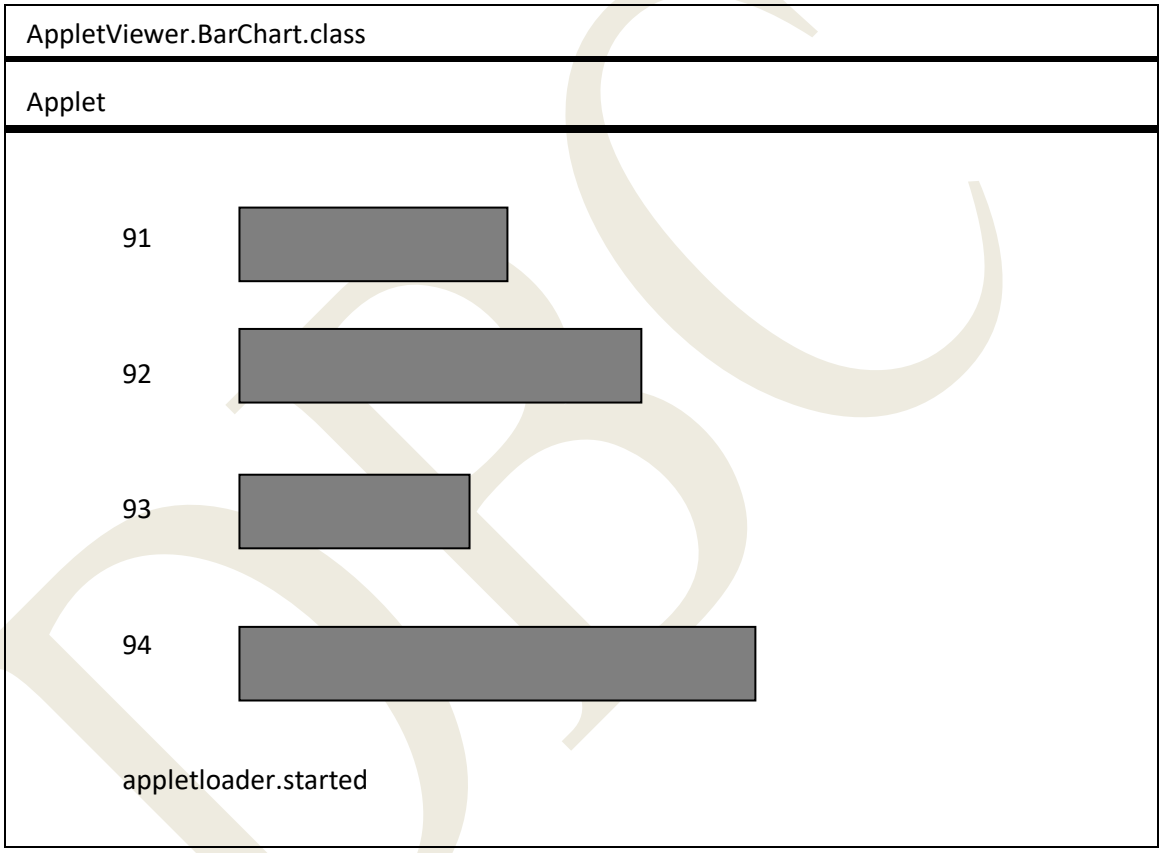
- Appletscanbedesignedtodisplaybarcharts,whicharecommonlyusedincomparativeanalysisof data.
- Thetablebelow showstheannualturnoverofa companyduringtheperiod1991–1994.
- Thesevalues maybeplacedinaHTMLfileasPARAMattributesandthenusedinappletfordisplayingabarchart .

Year	1991	1992	1993	1994
Turnover(RsCrores)	110	150	100	170

Example:

```
importjava.awt.*;im
portjava.applet.*;
publicclassBarChartextendsApplet
{
intn=0;
String label
[];intvalue[];pub
licvoidinit()
{
try
{
n=integer.parsInt
(getParameter("columns"));label=new
String[n]; value= new
int[n];label[0]=getParameter("lab
el1");
label[1]=getParameter("label2");
label[2]=getParameter("label3);
label[3]=getParameter("label4");
value [0] =Integer.parseInt (getParameter
("c1"));value [1] =Integer.parseInt (getParameter
("c2"));value [2] =Integer.parseInt (getParameter
("c3"));value[3]=Integer.parseInt(getParameter("c
4"));
}
catch(NumberFormatExceptione)
{
}
}
publicvoidpaint(Graphicsg)
{
for(inti=0;i<n;i++)
{
g.setcolor(color.red);g.drawstring(label
[i],20,i*50+30);g.fillRect(50,i*50+10,v
alue[i],40);
}
}
}
<html>
<applet
Code =
BarChart.classWidth=
300
Height=250>
```

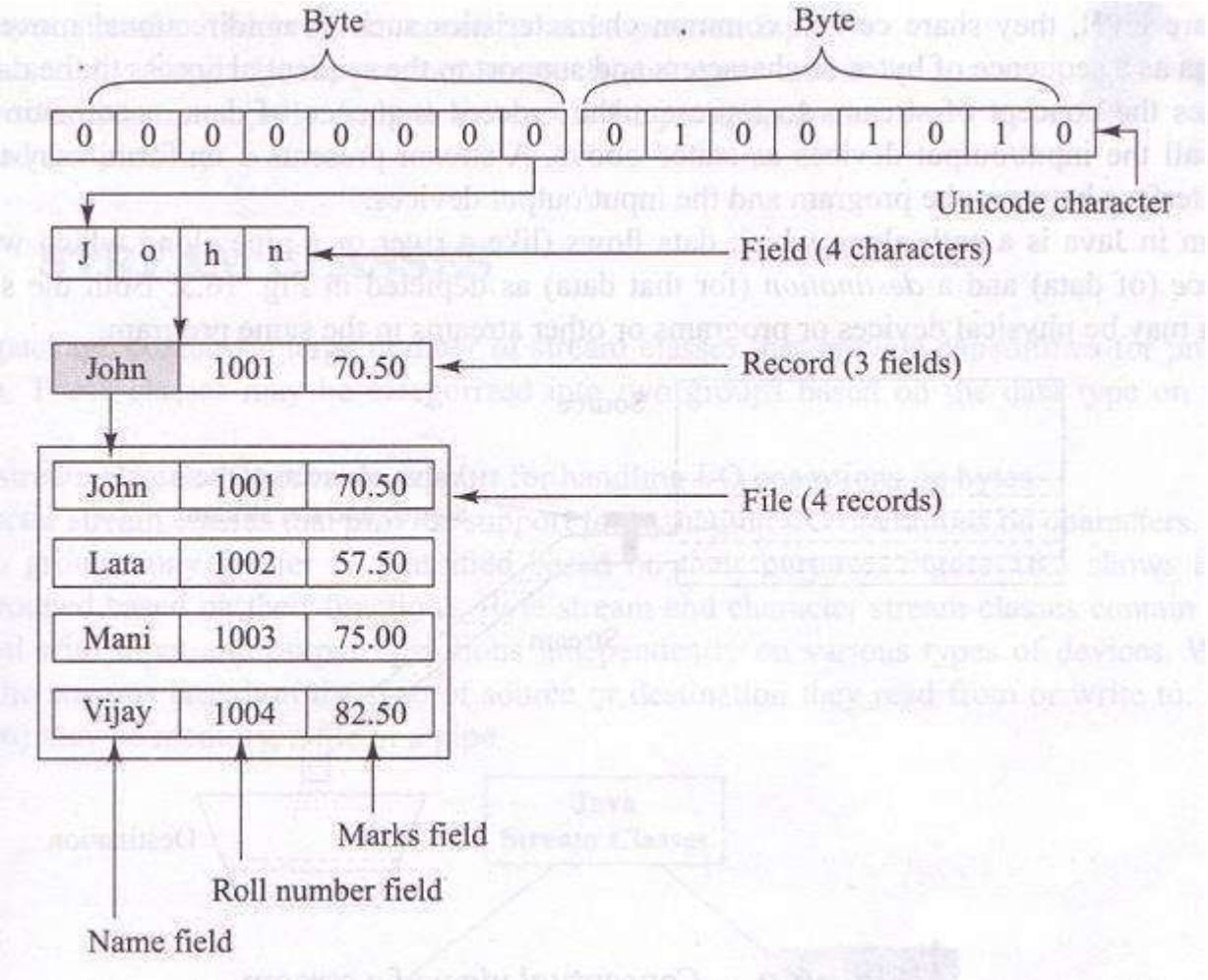
<PARAMNAME =“columns”VALUE=“4”>
<PARAMNAME=“c1”VALUE=”110”>
<PARAMNAME=“c2”VALUE=”150”>
<PARAMNAME=“c3”VALUE=”100”>
<PARAMNAME=“c4”VALUE=”170”>
<PARAMNAME =“label1”VALUE=”91”>
<PARAMNAME =“label2”VALUE=”92”>
<PARAMNAME =“label3”VALUE=”93”>
<PARAMNAME =“label4”VALUE=”94”>
</applet>
</html>



MANAGINGINPUT/OUTPUTFILESINJAVAINTRODUCTION

- ✓ Afileisacollectionofrecordsplacedinaparticulararеaonthedisk.
- ✓ Arecordiscomposedofseveralfieldsis a groupofcharacters.
- ✓ CharactersinjavaareUnicodecharacterscomposedoftwobytes,eachbytecontainingeightbinarydigits,1or0.

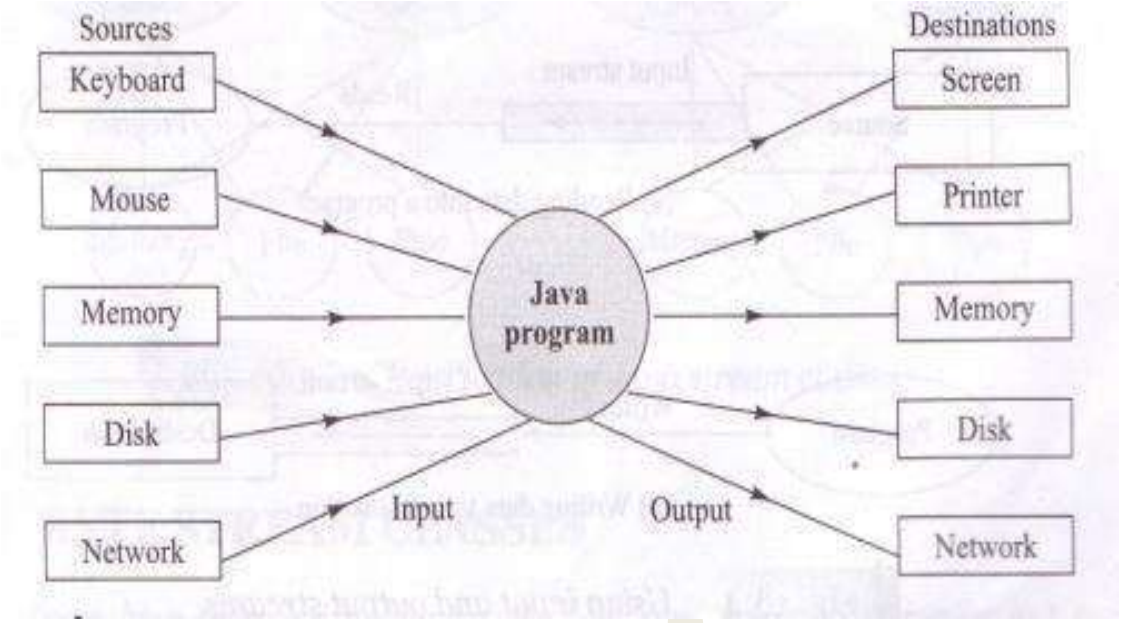
- ✓ Storing and managing data using files is known as file processing which includes taskssuchascreatingfiles,updatingfilesandmanipulationofdata.
- ✓ Javasupportsmanypowerfulfeaturesformanaginginputandoutputofdatausing files.
- ✓ Theprocessofreadingandwritingobjectsiscalledobjectserialization.



DatarepresentationinJavafiles

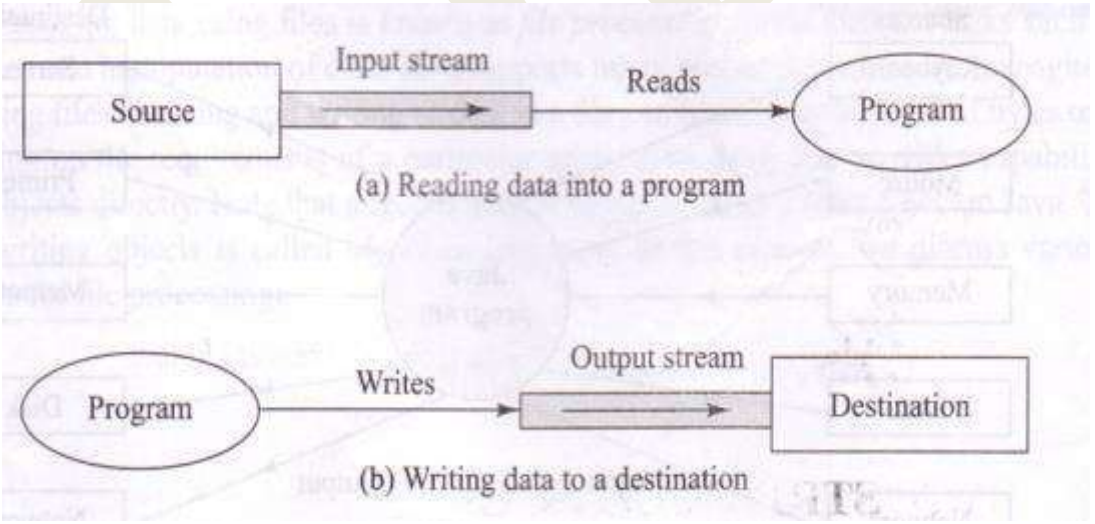
CONCEPTOFSTREAMS

- ✓ Infileprocessing,inputrefersto theflowofdataintoaprogramandoutputmeanstheflowofdataoutofaprogram.
- ✓ Input to a program may come from the keyboard, the mouse, the memory, the disk, a network, oranotherprogram.
- ✓ Outputfromaprogrammaygotothescreen,theprinter,thememory,thedisk,oranotherprogram.



Relationship ofJavaprogramwith I/Odevices

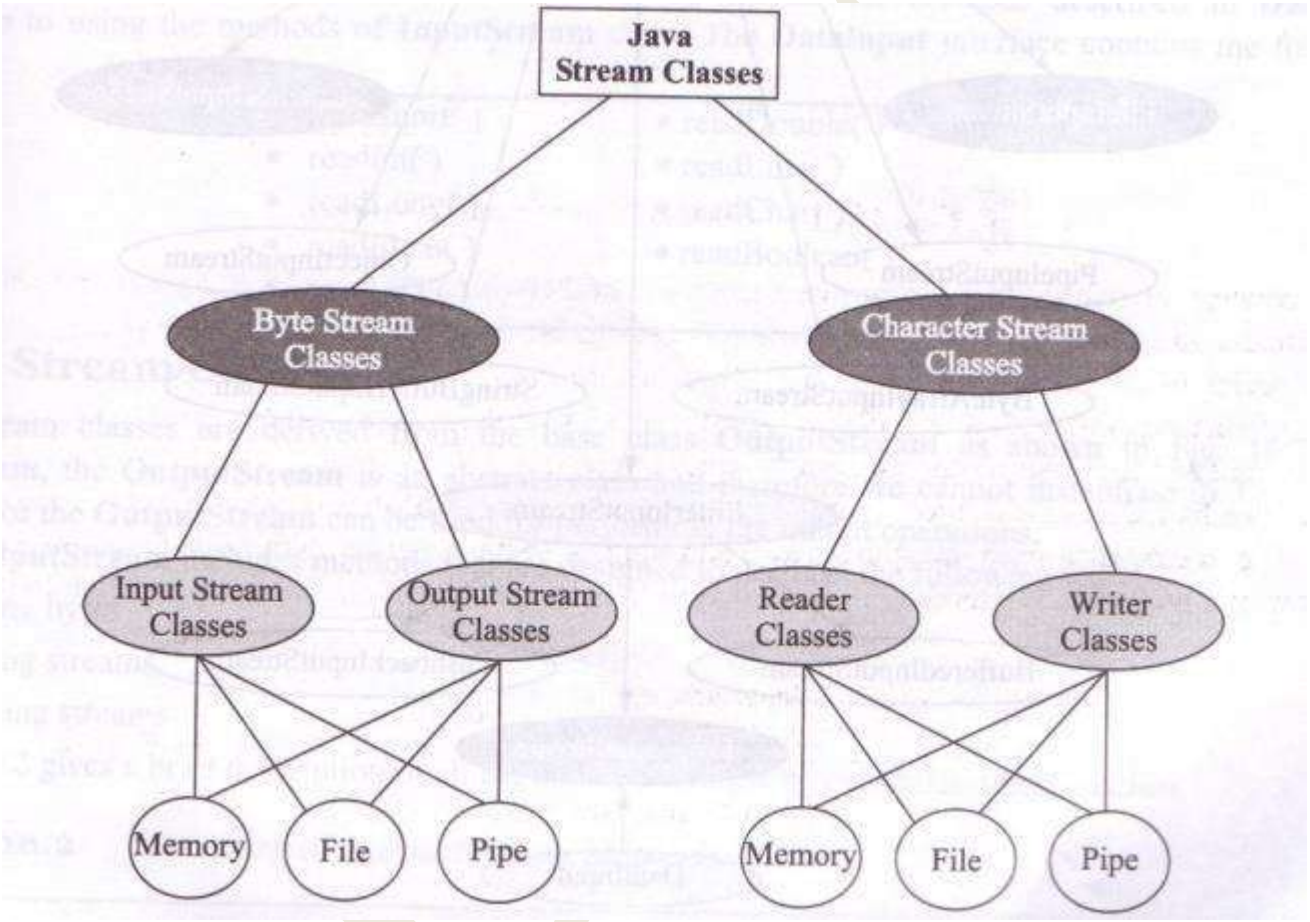
- ✓ Astreamin javaisa pathalongwhichdataflows.
- ✓ Ithasasourceandadestination.
- ✓ Both the source and the destination may be physical devices or programs or other streams inthesameprogram.
- ✓ Javastreamsareclassifiedintotwobasictypes, namely**input streamandoutput stream.**
- ✓ Aninputstreamextractsdatafromthesource(file) andsendsittotheprogram.
- ✓ Anoutputstreamtakesdatafromtheprogramandsendsittothedestination(file).Thefollowi ngfigureillustratestheuseofinputandoutputstreams.



Usinginputandoutputstreams

STREAMCLASSES

- The **java.io** package contains a large number of stream classes that provide capabilities forprocessingalltypes ofdata.
- Theseclassesmaybecategorizedintotwogroupsbasedonthedatatypeonwhichtheyoperate.
  1. Bytestreamclassesthatprovidesupportforhandling I/Ooperationsonbytes.
  2. Character streamclassesthatprovidesupportformanaging I/Ooperationsoncharacters.



Classificationofjavastreamclasses

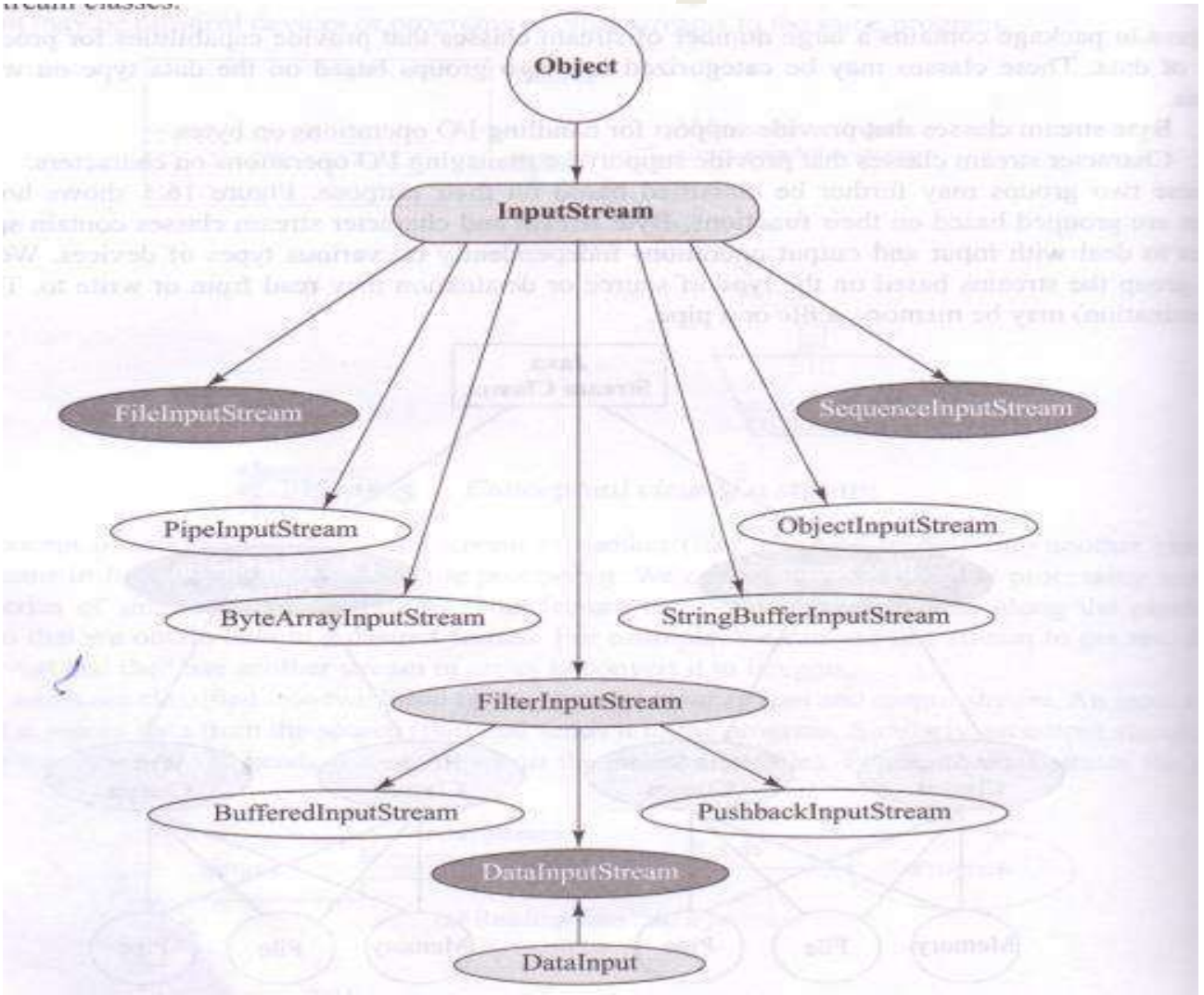
BYTESTREAMCLASSES

- ✓ Bytestreamclasseshavebeendesignedtoprovidefunctionalfeaturesforcreatingandmanip ulatingstreamsandfilesforreadingandwritingbytes.
- ✓ Sincethestreamsareunidirectional,theycantransmitbytesinonlyonedirection.
- ✓ Javaprovidestwo kindsofbyte streamclasses:
    - Inputstreamclasses
    - Outputstreamclasses

Inputstreamclasses

Input stream classes that are used to read 8-bit bytes include a super class known as **Inputstream**and a number of subclasses for supporting various input-related functions. The following figure showsthehierarchyofinputstreamclasses.

Hierarchyofinputstreamclasses



The**Inputstream**classdefinesmethodsfor performing inputfunctionssuchas

- Readingbytes
- Closingstreams
- Makingpositionsinstreams
- Skippingaheadinastream
- Findingthenumberofbytesinastream

Inputstreammethods

Method	*Description*
1. read ( )	Reads a byte from the input stream
2. read (byte b[ ])	Reads an array of bytes into b
3. read (byte b[ ], int n, int m)	Reads m bytes into b starting from nth byte
4. available( )	Gives number of bytes available in the input
5. skip(n)	Skips over n bytes from the input stream
6. reset( )	Goes back to the beginning of the stream
7. close( )	Closes the input stream

DataInputStreamextends**FilterInputStream**andimplementstheinterface**DataInput.**The **DataInput** interfacecontainsthefollowingmethods.

- readShort( )
- readInt( )
- readLong( )
- readFloat( )
- readUTF( )
- readDouble( )
- readLine( )
- readChar( )
- readBoolean( )

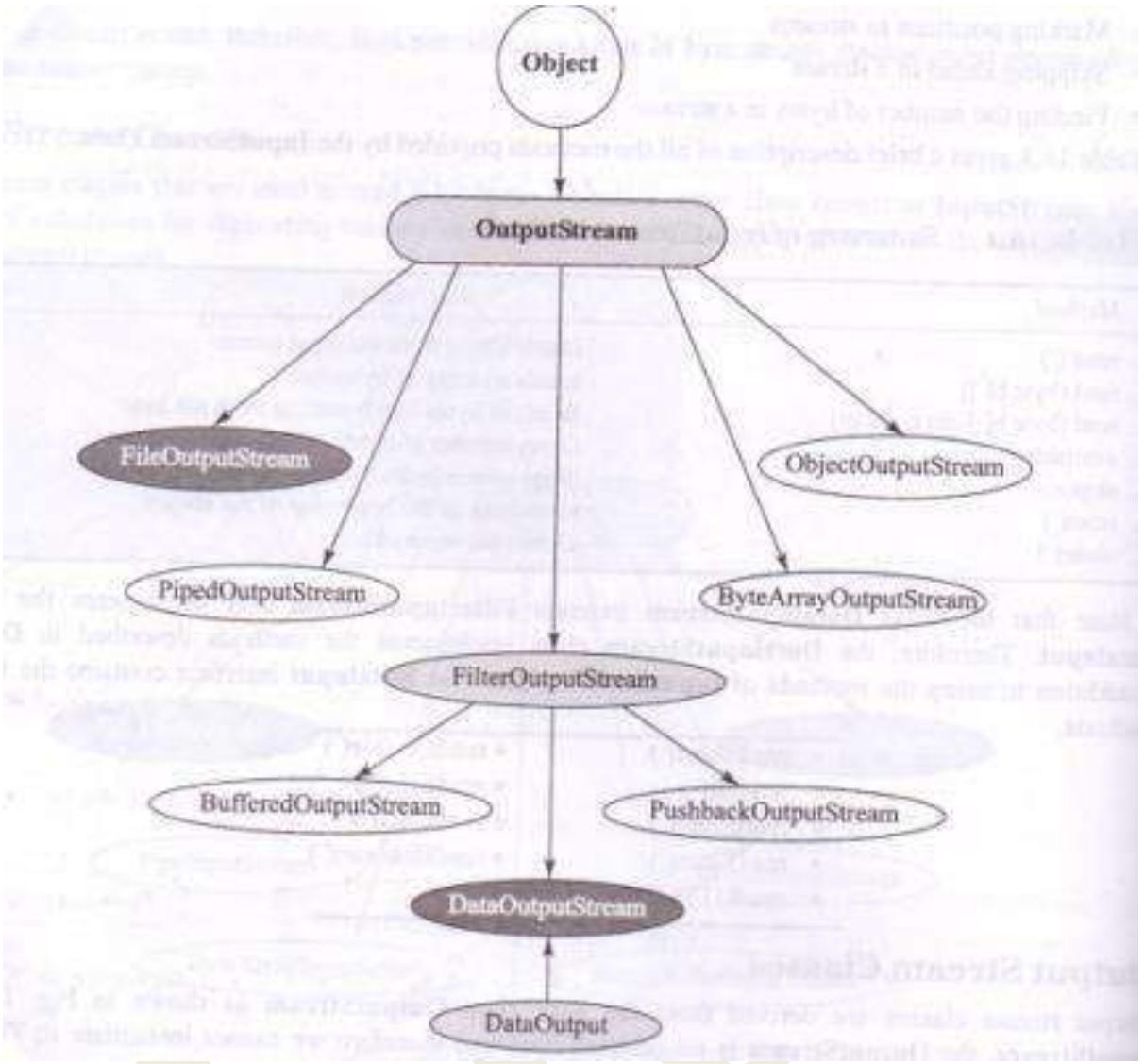
The**OutputStream** includes methodsthataredesignedtoperformthefollowingtasks:

- Writingbytes
- Closingstreams
- Flushingstreams

OuputStreamMethods

Method	*Description*
1. write ( )	Writes a byte to the output stream
2. write (byte b[ ])	Writes all bytes in the array b to the output stream
3. write (byte b[ ], int n, int m)	Writes m bytes from array b starting from nth byte
4. close( )	Closes the output stream
5. flush( )	*Flushes the output stream*

Hierarchyofoutputstreamclasses



The**DataOutputStream**,acounterpartof**DataInputStream**,implementstheinterface **DataOutput** and,therefore,implementsthefollowing methodscontainedin**DataOutput**interface.

- writeShort( )
- writeInt( )
- writeLong( )
- writeFloat( )
- writeUTF( )
- writeDouble( )
- writeBytes( )
- writeChar( )
- writeBoolean( )

CHARACTERSTREAMCLASSES

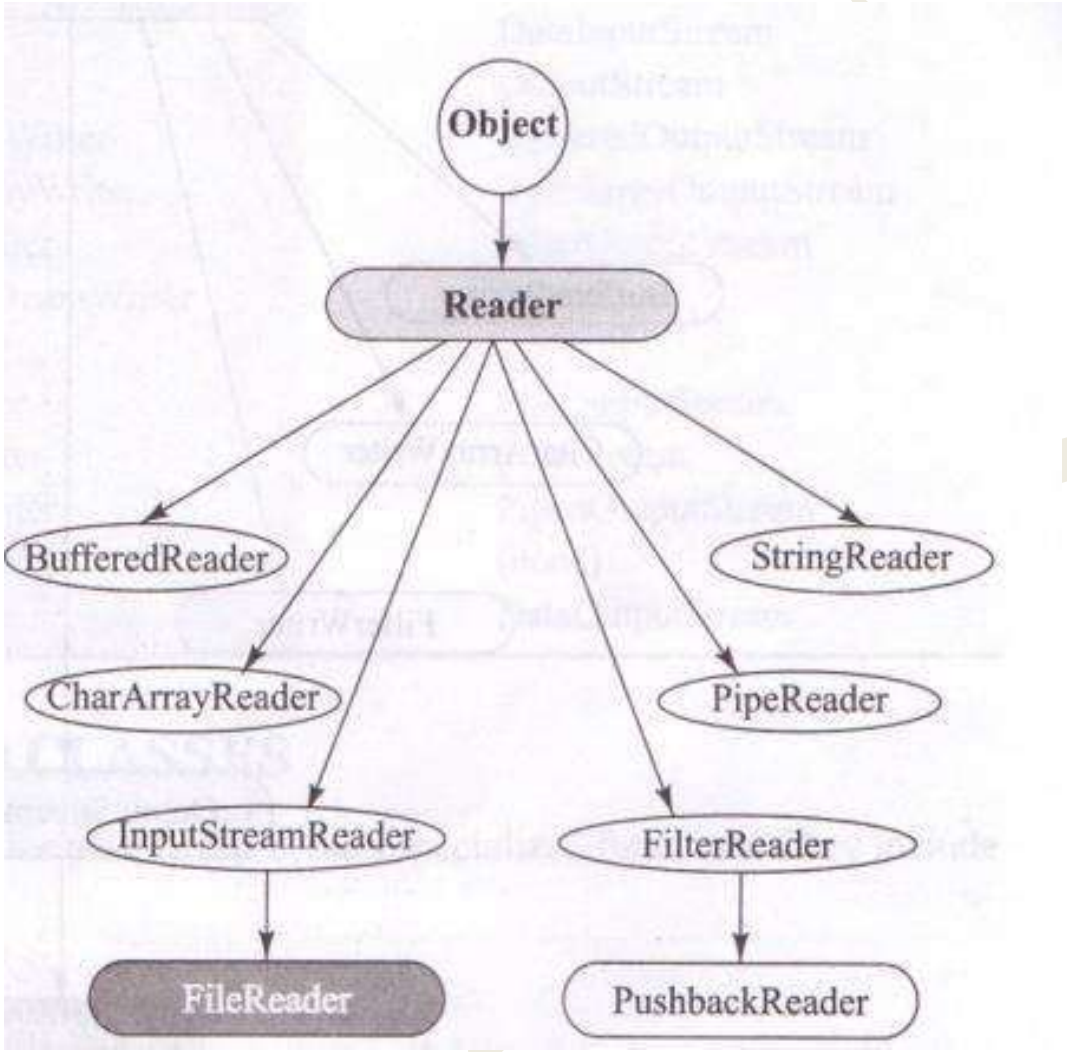
✓ Character streamclassescanbeusedtoreadandwrite16-bitUnicodecharacters.

- ✓ There are two kinds of character stream classes, namely, reader stream classes and writer streamclasses.

ReaderStreamclasses

- ❖ Reader streamclassesaredesignedtoread character fromthefiles.
- ❖ **Reader**classisthebaseclassforallotherclassesinthisgroup.
- ❖ The**Reader**classcontainsmethodsthat areidenticaltothoseavailableinthe**InputStream** class,except**Reader**isdesignedtohandlecharacters.

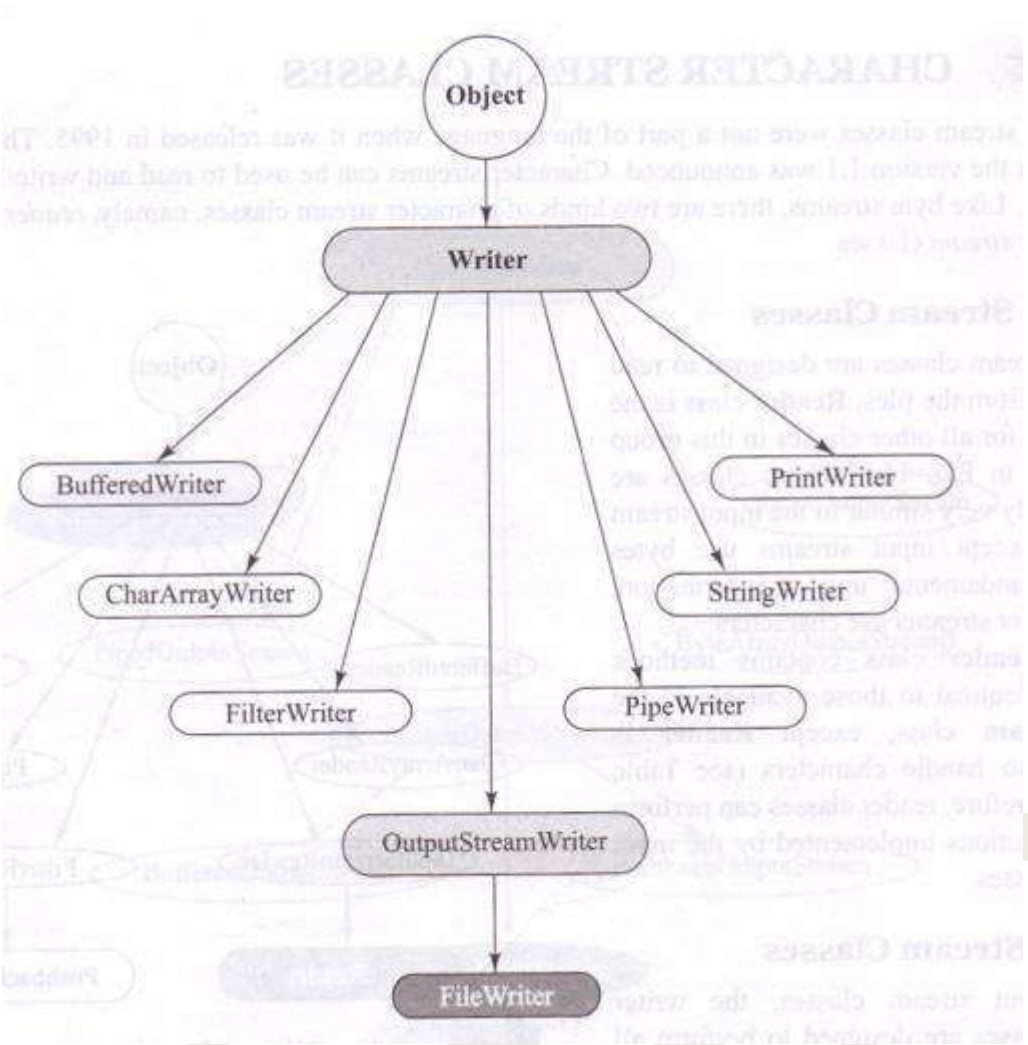
Hierarchyofreaderstreamclasses



WriterstreamClasses

- ❖ Thewriterstreamclassesaredesigned toperformalloutputoperations onfiles.
- ❖ Only difference is that while output stream classes are designed to write bytes, the writerstreamclasses aredesignedtowritecharacters.

Hierarchyofwriterstreamclasses



USINGSTREAMS

- ✓ The various types of input and output stream classes used for handling both 16-bit characters and8-bitbytes.
- ✓ Alltheclassesareknownasi/oclasses,notallofthemareusedforreadingandwritingoperations only.
- ✓ Someperformoperationssuchas buffering,filtering,dataconversion,countingandconcatenationwhilecarrying outi/otasks.

ListofTasksandClassesImplementingThem

Task	*Character Stream Class*	*Byte Stream Class*
Performing input operations	Reader	InputStream
Buffering input	BufFeredReader	BufferdInputStream
Keeping track of line numbers	LineNumberReader	LineNumberInputStream
Reading from an array	CharArrayReader	ByteArrayInputStream
Translating byte stream into a character stream	InputStreamReader	(none)
Reading from files	FileReader	FileInputStream

Task	*Character Stream Class*	*Byte Stream Class*
Filtering the input	FilterReader	FilterInputStream
Pushing back characters/bytes	PushbackReader	PushbackInputStream
Reading from a pipe	PipedReader	PipedInputStream
Reading from a string	StringReader	StringBufferInputStream
Reading primitive types	(none)	DataInputStream
Performing output operations	Writer	OutputStream
Buffering output	BufferedWriter	BufferedOutputStream
Writing to an array	CharArrayWriter	ByteArrayOutputStream
Filtering the output	FilterWriter	FilterOutputStream
Translating character stream into a byte stream	OutputStreamWriter	(none)
Writing to a file	FileWriter	FileOutputStream
Printing values and objects	PrintWriter	printStream
Writing to a pipe	PipedWriter	PipedOutputStream
Writing to a string	String Writer	(none)
Writing primitive types	(none)	DataOutputStream

USINGTHEFILECLASS

- ✓ The **java.io** package includes a class known as the **File** class that provides support for creatingfilesanddirectories.
- ✓ Theclassincludesseveralconstructorsforinstantiatingtheobjects.
- ✓ Thisclassalsocontainsseveralmethodsforsupportingtheoperationssuchas
  - Creatingafile
  - Openingafile
  - Closingafile
  - Deletingafile
  - Gettingthenameofafile
  - Gettingthesizeofafile
  - Checkingtheexistenceofafile
  - Renaminga file
  - Checkingwhetherthefile iswritable
  - Checkingwhetherthefileisreadable

CREATIONOFFILES

Ifwewanttocreateanduseadiskfile,weneedtodecidethefollowingaboutthefileandits intendedpurpose:

- Suitablenameforthefile
- Datatypetobestored
- Purpose(reading,writing,orupdating)
- Methodofcreatingthefile

✓ Afilenameisauniquestringofcharactersthathelpsidentifyafileonthedisk.

✓ Afilenamemaycontaintwoparts,aprimarynameandanoptionalperiodwithextension.

✓ Examples:

input.data	salary
test.doc	student.txt
inventory	rand.dat

✓ Datatypeisimportanttodecidethetypeoffilestreamclassestobeusedforhandlingthedata.

✓ Weshoulddecidewhetherthedatatobehandledisintheformofcharacters,bytesorprimitivetype.

✓ The purpose of using a file must also be decided before using it.For example, we know whetherthefileiscreated forreadingonly,orwritingonly,orboththe operations.

✓ For using a file, it must be opened first.This is done by creating a file stream and then linking itto thefilename.

✓ Thecommonstreamclassesusedfor variousi/ooperationsaregiveninthefollowingtable:

Commonstreamclassesused forI/Ooperations

Source or Destination	Characters		Bytes	
	Read	*Write*	*Read*	*Write*
Memory	CharArrayReader	CharArrayWriter	ByteArrayInputStream	ByteArrayOutputStream
File	FileReader	FileWriter	FileInputStream	FileOutputStream
Pipe	PipedReader	PipedWriter	PipedInputStream	PipedOutputStream

❖ Therearetwowaysofinitializingthefilestreamobjects.

❖ All of the constructors require that we provide the name of the file either directly, or indirectly bygivinga file object that has already been assigned a filename.The following code segmentillustratestheuseofdirectapproach.

```
FileInputStreamfis;try
   {
        fis=newFileInputStream("test.dat");
        ……..
    }
     catch    (IOExceptione)
    …..
```

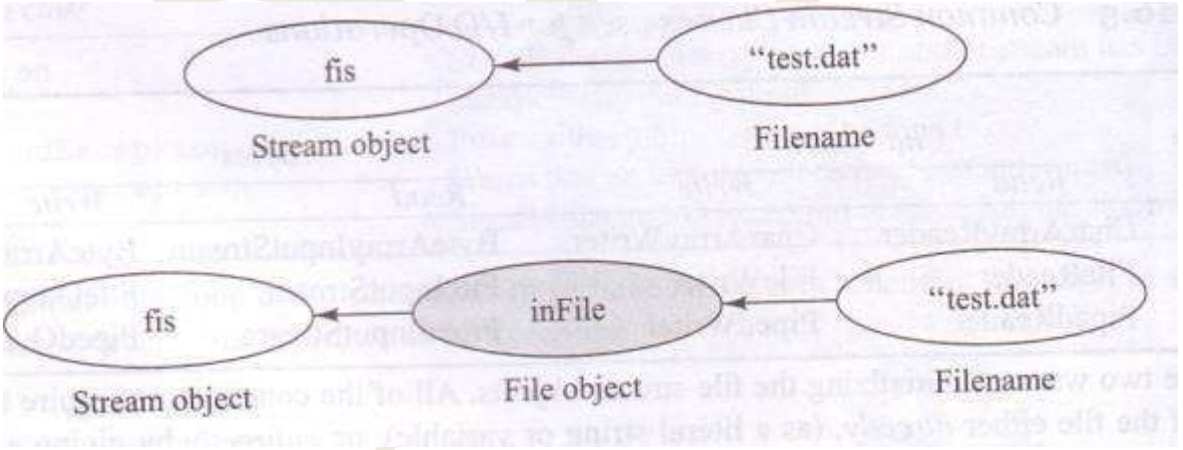
The indirectapproachusesafileobjectthathas beeninitializedwiththedesiredfilename. Thisisillustratedbythefollowingcode.

```
……
……..
FileinFile;
InFile = new File
 (“test.dat”);FileInputStream
 fis;
  try
 {
        fis=newFileInputStream(inFile);
        ……..
  }
    catch    (.....)
   …..
```

Thecodeaboveincludesfivetasks:

- Selectafilename
- Declarethefileobject
- Givetheselectednametothefileobjectdeclared
- Declareafilestreamobject
- Connect thefiletothefilestreamobject
- Boththeapproachesareillustratedinthefollowingfigure:

Hierarchyofwriterstreamclasses



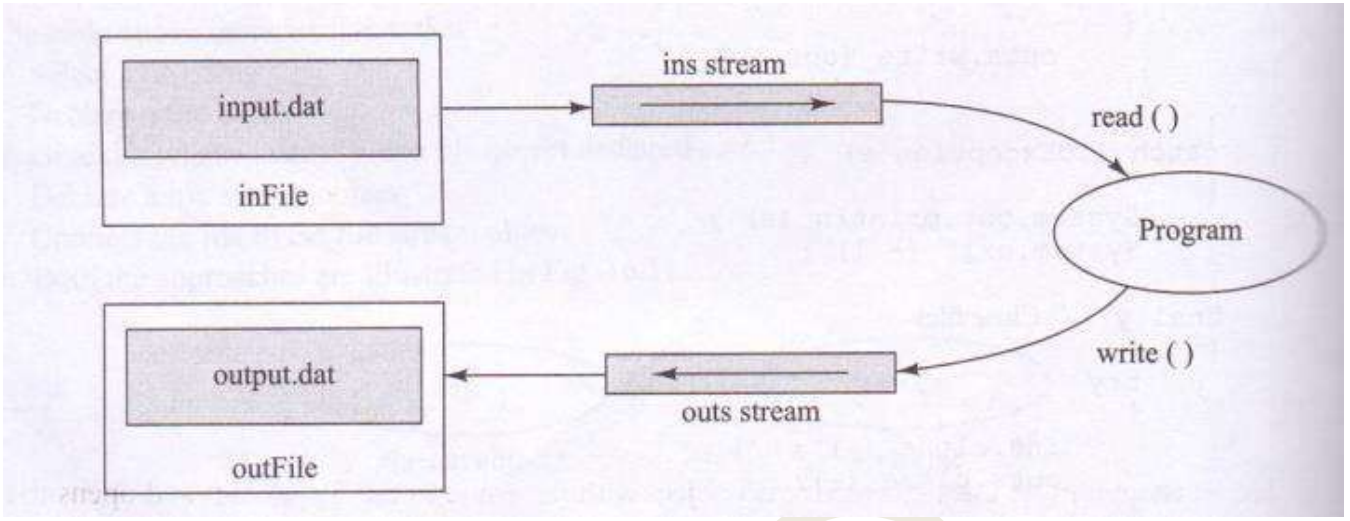
READING/WRITINGCHARACTERS

The two subclasses used for handling characters in files are FileReader(for reading characters) andFileWriter(forwritingcharacters).Thefollowingprogramusesthesetwofilestreamclassestocopythecontentsofafilenamed“input.dat”intoafilecalled“output.dat”.

```
importjava.io.*;class
CopyCharacters
  {
```

```
publicstaticvoidmain(Stringargs[ ])
  {
      FileinFile=newFile("input.dat");Fileou
      tFile=new File("output.dat");
      FileReaderins=null;                          //createsfile streamins
     FileWriter outs=null;                        // creates file stream
     outstry
   {
     ins=newFileReader(inFile);                   //opens inFile
     outs=newFileWriter(outFile);
                                                  //opensoutFilei
     ntch;
      while(( ch=ins.read())!=-1)
        {
            outs.write(ch);
        }
  }
   catch(IOExceptione)
    {
      System.out.println(e);
       System.exit(-1);
     }
     finally     //closefiles
       {
            try
              {
                ins.close();
                outs.close();
               }
             catch(IOExceptione){}
        }
    }
}
```

Theconceptofusing filestreamsandfileobjectsforreadingandwritingcharactersintheaboveprogramisillustratedinthefollowingfigure:



Readingfromand writingtofiles

READING/WRITINGBYTES

Thefollowingprogramdemonstrateshow**FileOutputStream**class isusedfor writingbytestoafile. The program writes the names of some cities stored in a byte array to a new file named "city.txt".wecanverifythecontentsofthelifebyusing thecommand

typecity.txt

Example:

```
//writingbytestoafileim
port
java.io.*;classWrite
Bytes
  {
    publicstaticvoidmain(Stringargs[])
      {
        //declareandinitializeabytearray
        bytecities[ ] ={'D','E','L','H','I',*\n\'M','A','D','R','A','S',*\n\,
                                'L','O','N','D','O','N'}
         // create an output file
          streamFileOutputStreamOutfile=n
          ull;try
                {
                  //connecttheoutfilestreamto"city.txt"
                    outfile=newFileOutputStream("city.txt");
                    //writedatatothestreamoutfi
                      le.write
                      (cities);outfile.close ();
                }
            catch(IOExceptionioe)
```

```
                {
                    System.out.println(ioe);
                    System.exit(-1);
                }
            }
        }
```

Output:

typecity.txt

DELHI

MADRAS

LONDON

The following program shows how **FileInputStream**class is used for reading bytes from afile.The program reads an existing file and displays its bytes on the screen.We run this program, wemust first create a file for it to read. We may use this program to read the file city.txt created in aboveprogram.

Example:

```
// readingbytes from a
   fileimportjava.io.*;
   classReadBytes
     {
       publicstaticvoidmain(Stringargs[])
         {
           // create an input file
            streamFileInputStream infile
            = null;intb;
            try
                {
                  //connecttheinfilestreamto
                    therequiredfileinfile=newFileInputStream(ar
                    gs[0]);
                    //readanddisplaydata
                    While((b=infile.read())!=-1)
                    {
                        System.out.println((char)b);
                     }
                       Infile.close();
                }
             catch(IOExceptionioe)
                {
                    System.out.println(ioe);
                 }
            }
        }
```

Theprogramdisplaysthefollowingwhenwesupplythefilename“city.txt”.

Prompt>javaReadBytescity.txt

DELHI

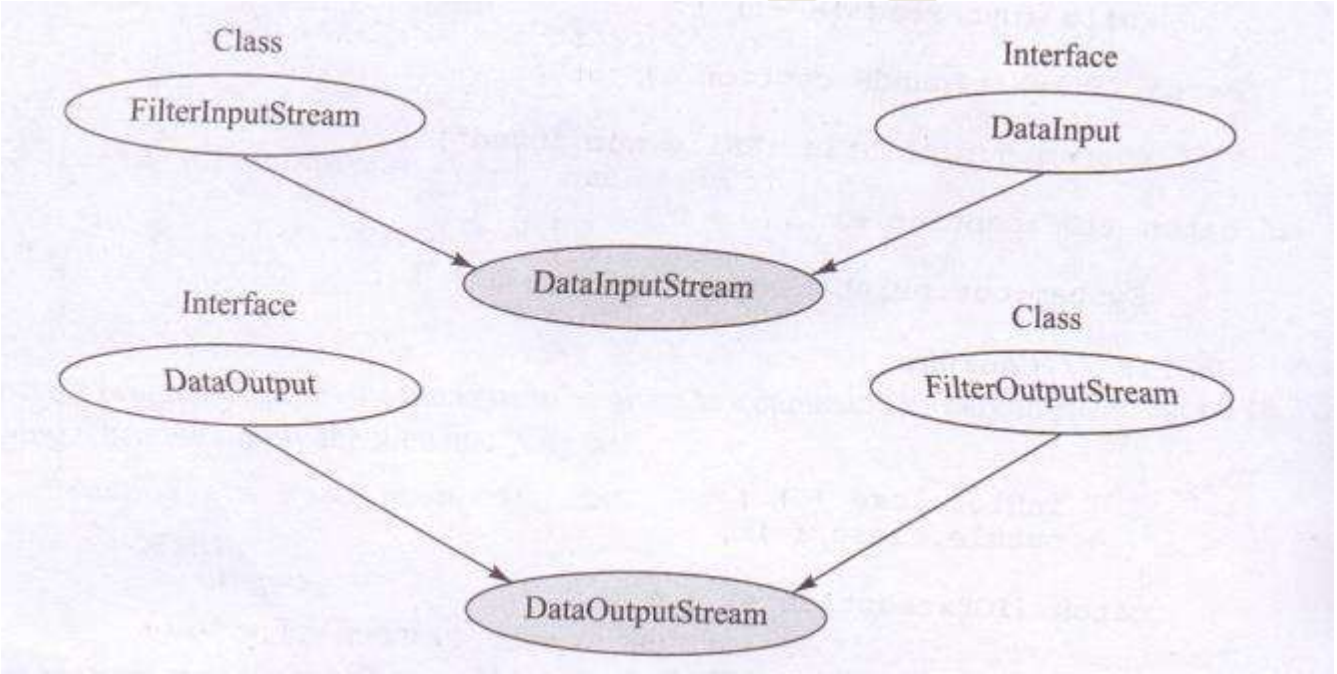
MADRAS

LONDON

HANDLINGPRIMITIVEDATATYPES

- ✓ Two filter classes used for creating "data streams" for handling primitive types are**DataInputStream** and **DataOutputStream.**These classes use the concepts of multipleinheritanceas showninthefollowingfigure.

Hierarchyofdatastreamclasses



Adatastreamforinputcanbecreatedasfollows:

FileInputStream fis = new FileInputStream (infile);DataInputStreamdis=newDataInputStream(fis);

These statements first create the input file stream fis and then create the input data streamdis.These statementsbasically wrap dis on fis and use it as a "filter".Similarly, the followingstatementscreatethe outputdata streamdosandwrapitovertheoutputfilestreamfos.

FileOutputStreamfos=newFileOutputStream(outfile);DataOutputSt reamdos=newDataOutputStream(fos);

Example:

// reading and writing primitive dataimportjava.io.*;

classReadWritePrimitive

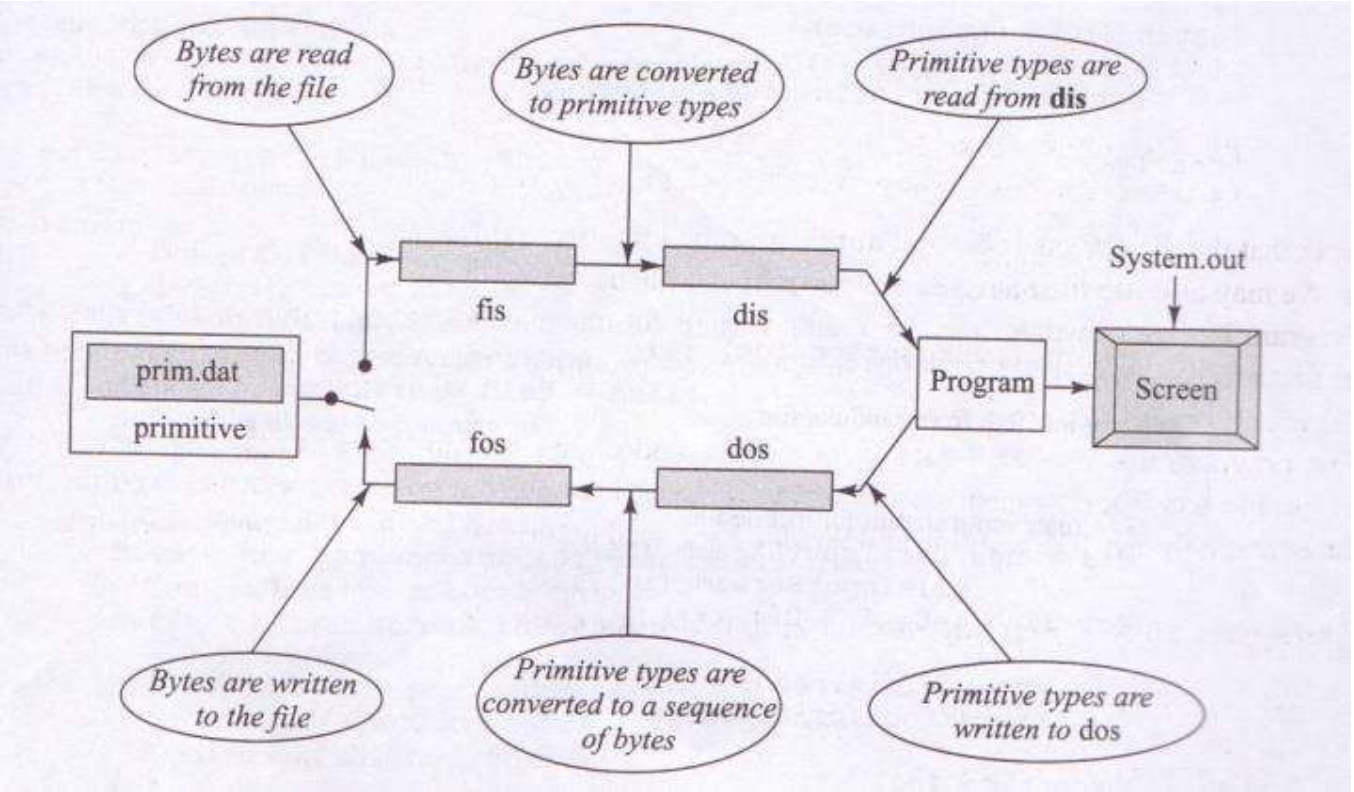
```
{
        publicstaticvoidmain(Stringargs[])throwsIOException
          {
              Fileprimitive=newFile("prim.dat");
              FileOutputStream fos = new FileOutputStream
               (primitive);DataOutputStreamdos
               =newDataOutputStream(fos);
               //writeprimitivedatatothe"prim.dat"filedos.
               writeInt(1999);dos.writeDouble(375.85);do
               s.writeBoolean(false);dos.writeChar('X');
                 fos.close();
                 //readdatafromthe"prim.dat"file
                  FileInputStreamfis=newFileInputStream(primitive);DataInput
                  Streamdis=newDataInputStream(fis);
                    System.out.println(dis.readInt());System
                  .out.println(dis.readDouble());System.out
                  .println(dis.readBoolean());System.out.pri
                  ntln(dis.readChar());dis.close();
                   fis.close();
          }
    }
```

Output:

```
1999
375.75
false
X
```

Thedatastreamsusedinaboveprogramandtheirfunctionsareillustratedinthefollowing figure:

DBC



CONCATENATINGANDBUFFERINGFILES

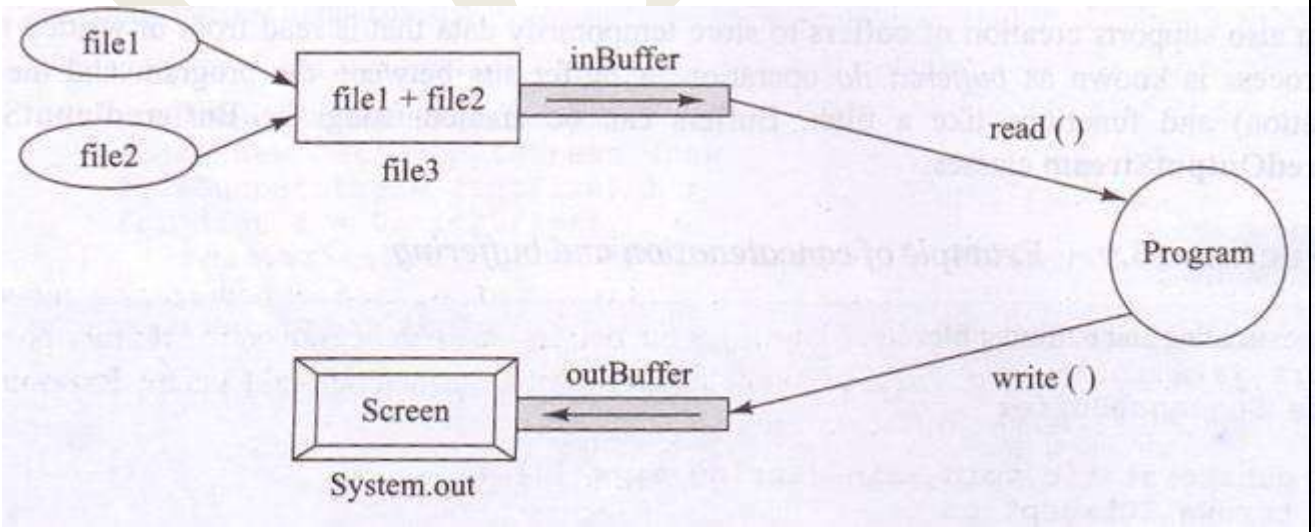
- ✓ Itispossibletocombinetwoormoreinputstreams intoasingle inputstream.
- ✓ Thisprocessisknownasconcatenationoffilesandachieved usingthe**SequenceInputStream** class.
- ✓ OneoftheconstructorsofthisclasstakestwoInputStreamobjectsasargumentsandcombinesthemtoc onstructasingleinputstream.
- ✓ Java also supports creation of buffers to stroe temporarily data that is read from or written to astream.
- ✓ Theprocessisknownasbufferedilooperation.
- ✓ Abuffer sitsbetweentheprogramandthesourceandfunctionslikeafilter.
- ✓ Buffercanbecreatedusingthe **BufferedInputStream**and**BufferedOutputStream.**

Example:

```
importjava.io.*;
classSequenceBuffer
  {
    public static void main (String args[
    ])throwsIOException
    {
          //declarefilestreamsFileInput
            Stream
            file1=null;FileInputStream
            file2=null;
```

```
            //declarefile3tostorecombinedfiles
          SequenceInputStreamfile3=null;
              //openthefilestobeconcatenatedfile1=ne
            w FileInputStream("text1.dat");
             file2=new FileInputStream("text2.dat");
         //concatenatefile1and file2into file3
          file3=newSequenceInputStream(file1,file2);
           // create buffered input and output
       streamsBufferedInputStreaminBuffer=
                newBuffered
        InputStream(file3);BufferedOutputStrea
        mOutBuffer=
                newBufferedOutputStream(System.out);
  //read and writetilltheend
  ofbuffersintch;
   while((ch=inBuffer.read())!=-1)
      {
        outBuffer.write((char)ch);
       }
     inBuffer.close();ou
      tBuffer.close();fil
      e1.close();file2.cl
      ose();
   }
}
```

Theentireprocessofconcatenation,bufferinganddisplayingthecontentsoftwoindependentfilesisillustratedinthefollowingfigure:



RANDOMACCESSFILES

- ✓ Files can be used either for “read only” or for “write only” operations and not for both purposessimultaneously.
- ✓ Thesefilesare readorwrittenonlysequentiallyand,therefore,areknownassequential files.
- ✓ **Random Access File** class supported by the **Java.io** package allows us to create files that can beusedforreadingandwritingdatawithrandomaccess.
- ✓ suchfilesareknownasrandomaccessfiles.
- ✓ A file can be created and opened for random access by giving a mode string as a parameter to theconstructorwhenweopenthefile.Wecanuseoneofthefollowingtwomodestrings:
- ✓ “r”for readingonly
- ✓ “rw” forbothreadingandwriting
- ✓ Anexistingfilecanbeupdatedusingthe“rw”mode.

Example:impor
t
java.io.*;classR
andomIO
{
publicstaticvoid main(stringargs[])
{
RandomAccessFile
file=null;try
{
file=newRandomAccessFile(“rand.dat”,”rw”);
//Writing to the
filefile.WriteChar(‘X’);file.writeInt
(555);file.writeDouble(3.1412);file
.seek(0);**//Gotothebeginning**
//Reading from the
fileSystem.out.println(file.readChar());
System.out.println(file.readInt());
`
System.out.println(file.readDouble
());file.seek(2);//Go to the second
itemSystem.out.println(file.readInt());
//Go to the end and append false to the
filefile.seek(file.length());file.writeBoolean(f
alse);
file.seek(4);System.out.println(file.read
Boolean());file.close();
}
catch(IOExceptione){System.out.println(e);}
}
}

Theprogramopensarandomaccessfileandthenperformsthefollowingoperations.

1. Writes three items of
   datax
   555
   3.1412
2. Brings the file pointer to the beginning3.Readsanddisplaysallthethree items
   x55
   5
   3.1412
4. Takesthepointertotheseconditemandthenreadsand displaysthesecondtemin thefile.
   555
5. Placesthepointerat theendusingthemethod length()and thenadds aBooleanitemto thefile.
6. Finally, takesthepointertothefourthitemand displaysit.
7. Attheend,closesthefile.

Theoutputonthescreenwouldappearasfollows:x
555
3.1412
555
false

QUESTIONS

2Marks

1. Writeanytwomethodsofgraphics class.
2. Defineobjectserialization.
3. Whatdoyoumeanbyfileprocessing?
4. Whataretwo typesofstreams?
5. Whataretheinputfunctionsperformedbytheinputstreamclass?
6. Writedowntheinputstreammethods.
7. Whataretheoutputstreammethodsusedinjava?
8. Definerandomaccess file.

5Marks

1. Howwill youdrawlinesandrectanglesusinggraphicsclass method?
2. Explainaboutcontrolloopsinapplets.
3. Writeshortnotesonstreamclasses.
4. Explainabouttypesofbytestreamclasses.
5. Describeaboutreader andwriterstreamclasses.
6. Whataretheusesoffileclass?
7. Writeindetailaboutcommonstreamclassesused forI/Ooperations.

10Marks

1. Explainindetailaboutrandomaccessfiles.
2. Writeshortnotesonhandlingprimitivedatatypes.

UNITVCOMPLETED

DBC