MRCETCAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUSINSTITUTION–UGC,GOVT.OF INDIA)

Department of CSE

(EmergingTechnologies)

(DataScience,CyberSecurity,InternetofThings)

B.TECH(R-22 Regulation)

(II YEAR – II SEM)

(2023-24)

ObjectOrientedProgrammingthrough JAVA

(R22A0508)

LECTURE NOTES

MALLAREDDYCOLLEGEOFENGINEERING&TECHNOLOGY

(AutonomousInstitution–UGC,Govt.ofIndia)

Recognizedunder2(f)and12(B)ofUGCACT1956

(AffiliatedtoJNTUH,Hyderabad,ApprovedbyAICTE-AccreditedbyNBA&NAAC–'A'Grade- ISO9001:2015Certified)

Maisammaguda,Dhulapally(PostVia.Hakimpet),Secunderabad–500100,TelanganaState,India

DepartmentofComputerScienceandEngineering

EMERGINGTECHNOLOGIES

ObjectOrientedProgrammingthrough JAVA
(R22A0508)

LECTURE NOTES

Prepared by

Mrs.D.Kalpana, Asst.Prof.

**(First Time Prepared Faculty)*

On

20.02.2022

Updated by

Mrs.V.Bala,Asst.Prpf.

***(Second-time Prepared Faculty if updated the existed Lecture Notes)*

On

10.01.2024

DepartmentofComputerScienceandEngineering

EMERGINGTECHNOLOGIES

Vision

- "Tobe at the forefront ofEmerging Technologies andtoevolve as a Centre ofExcellence in Research, Learning and Consultancy to foster the students into globally competent professionals useful to the Society."

Mission

ThedepartmentofCSE(EmergingTechnologies) iscommitted to:

- To offer highest Professional and Academic Standards in terms of Personal growth and satisfaction.
- Makethesocietyasthehubofemergingtechnologiesandtherebycaptureopportunities in new age technologies.
- TocreateabenchmarkintheareasofResearch,EducationandPublicOutreach.
- Toprovidestudentsaplatformwhereindependentlearningandscientificstudyare encouraged with emphasis on latest engineering techniques.

QUALITYPOLICY

- To pursue continual improvement of teaching learning process of Undergraduate and Post Graduate programs in Engineering & Management vigorously.
- To provide state of art infrastructure and expertise to impart the quality education and research environment to students for a complete learning experiences.
- Developingstudentswithadisciplinedandintegratedpersonality.
- Toofferqualityrelevantandcosteffectiveprogrammestoproduceengineersasper requirements of the industry need.

Formoreinformation:www.mrcet.ac.in

SYLLABUS

II Year B.Tech. CSE (DS / CYS / IOT) - II Sem L/T/P/C 3/-/-/3

(R22A0508)OBJECTORIENTEDPROGRAMMINGTHROUGHJAVA

COURSE OBJECTIVES:

Thiscoursewillenablethestudents:

1. Tointroducetheobjectorientedprogramming concepts.
2. Tointroducetheprinciplesofinheritanceandpolymorphism;anddemonstrate how they relate to the design of abstract classes and to implementation of packages and interfaces
3. Tointroducethe conceptsofexceptionhandlingandmultithreading.
4. TointroducetheconceptsofCollectionFramework.
5. TointroducethedesignofGraphicalUserInterfaceusing applets and swingcontrols.

UNIT-I

Java Programming- History of Java, comments, Java Buzz words, Data types, Variables, Constants, Scope and Lifetime of variables, Operators, Type conversion and casting, Enumerated types, Control flow- block scope, conditional statements, loops, break and continue statements, arrays, simple java stand alone programs, class, object, and its methods constructors, methods, static fields and methods, access control, this reference, overloading constructors,recursion, exploring string class, garbage collection.

UNIT–II

Inheritance – Inheritance types, super keyword, preventing inheritance: final classes and methods.

Polymorphism – method overloading and method overriding, abstract classes and methods.

Interfaces- Interfaces Vs Abstract classes, defining an interface, implement interfaces, accessing implementations through interface references, extending interface, inner class.

Packages-Defining,creatingandaccessingapackage,importing packages.

UNIT-III

Exception handling- Benefits ofexception handling, the classificationofexceptions – exceptionhierarchy, checkedexceptionsanduncheckedexceptions, usageoftry,catch, throw, throws and finally, creating own exception subclasses.

Multithreading–Differencesbetweenmultipleprocessesand multiplethreads,threadlife cycle, creating threads, interrupting threads, thread priorities, synchronizing threads, inter-thread communication, producer consumer problem.

UNIT-IV

CollectionFrameworkinJava–Introductiontojavacollections,Overviewofjavacollection framework,Commonlyusedcollectionclasses-ArrayList,Vector,Hashtable,Stack,Lambda Expressions.

Files- Streams- Byte streams, Character streams, Text input/output, Binary input/output, File management using File class.

Connecting to Database – JDBC Type 1 to 4 drivers, Connecting to a database, querying a databaseandprocessingtheresults,updatingdatawithJDBC,DataAccessObject(DAO).

UNIT-V

GUIProgrammingwithSwing-TheAWTclasshierarchy,IntroductiontoSwing,SwingVs AWT, Hierarchy for Swing components, Overview of some Swing components – Jbutton, JLabel, JTextField, JTextArea, simple Swing applications, Layout management – Layout manager types –border, grid and flow

Event Handling- Events, Event sources, Event classes, Event Listeners, Delegation event model, Examples:HandlingMouseand Keyevents,Adapterclasses.

TEXTBOOK:

1. JavaFundamentals–AComprehensiveIntroduction,HerbertSchildtandDaleSkrien, TMH.
2. CoreJava: AnIntegratedApproach– DrRNageswara Rao

REFERENCEBOOKS:

1. Java forProgrammers,P.J.DeitelandH.M.Deitel,PEA(or)Java:HowtoProgram, P.J.Deitel and H.M.Deitel, PHI
2. ObjectOrientedProgrammingthroughJava,P.RadhaKrishna,UniversitiesPress.
3. ThinkinginJava,BruceEckel, PE
4. ProgramminginJava,S.MalhotraandS.Choudhary,OxfordUniversitiesPress.
5. DesignPatternsErichGamma,RichardHelm,RalphJohnsonandJohnVlissides.

COURSE OUTCOMES:

1. AbletounderstandtheuseofOOPsconcepts.
2. Abletounderstandtheuseofabstraction,PackagesandInterfaceinjava.
3. Abletodevelopandunderstandexceptionhandling,multithreadedapplicationswith synchronization.
4. Ableto understandtheuseofCollectionFramework.
5. AbletodesignGUI basedapplicationsanddevelopappletsforwebapplications.

INDEX

Unit-1

JavaProgramming-HistoryofJava

ThehistoryofjavastartsfromGreenTeam.Javateammembers(alsoknown as **Green Team**), initiated a revolutionary task to develop a language for digital devices such as set-top boxes, televisions etc.

Forthegreenteammembers, it wasanadvanceconcept atthattime. But, it wassuited for internet programming. Later, Java technology as incorporated by Netscape.

Currently, Java is used in internet programming, mobile devices, games, e-business solutions etc. There are given the major points that describes the history of java.

1) **JamesGosling**,**MikeSheridan**,and**PatrickNaughton**initiatedtheJavalanguageproject in June 1991. The small team of sun engineers called**Green Team**.

2) Originallydesignedforsmall, embeddedsystemsinelectronicappliancesliketset-topboxes.

3) Firstly,itwascalled**"Greentalk"**byJamesGoslingandfileextensionwas.gt.

4) After that, it was called Oak and was developed as a part of the Green project. Java Version History

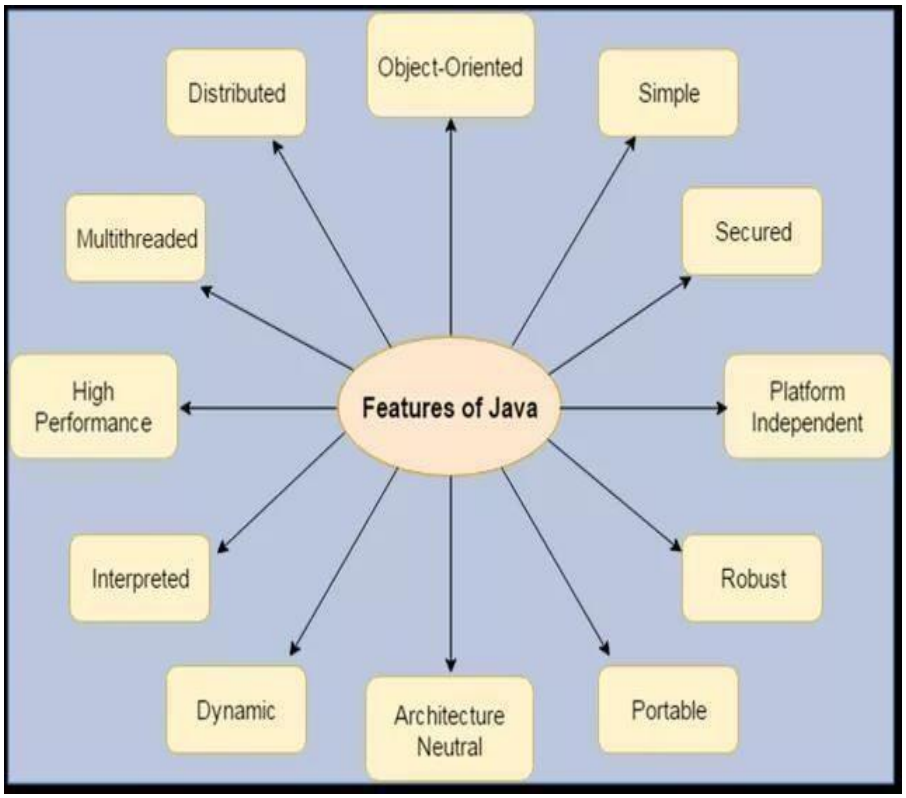
Therearemanyjavaversionsthathasbeenreleased.CurrentstablereleaseofJavaisJavaSE 8.

1. JDKAlphaand Beta(1995)
2. JDK1.0(23rdJan,1996)
3. JDK1.1(19thFeb,1997)
4. J2SE1.2(8thDec,1998)
5. J2SE 1.3(8thMay,2000)
6. J2SE1.4(6th Feb,2002)
7. J2SE5.0(30thSep,2004)
8. JavaSE6(11thDec,2006)
9. JavaSE7(28thJuly,2011)
10. JavaSE8(18thMarch,2014)

FeaturesofJava

There isgivenmanyfeaturesofjava. Theyarealso knownasjava buzzwords.TheJavaFeatures given below are simple and easy to understand.

1. Simple
2. Object-Oriented
3. Portable
4. Platformindependent
5. Secured
6. Robust
7. Architectureneutral
8. Dynamic
9. Interpreted
10. HighPerformance
11. Multithreaded
12. Distributed



JavaComments

The java comments are statements that are not executed by the compiler and interpreter. The comments canbe used to provide informationor explanationaboutthe variable, method, class or any statement. It can also be used to hide program code for specific time.

TypesofJavaComments

Thereare 3typesofcommentsinjava.

1. SingleLineComment
2. MultiLineComment
3. DocumentationComment

JavaSingleLineComment

Thesinglelinecommentisusedtocommentonlyone line.

Syntax:

```
1.//Thisissingle linecomment
```

Example:

```
publicclassCommentExample1{
publicstaticvoidmain(String[]args){ int
   i=10;//Here, i is a variable
   System.out.println(i);
}
}
```

Output:

```
10
```

JavaMultiLineComment

Themultilinecommentisusedtocommentmultiplelinesofcode.

Syntax:

```
 /*
 This
 is
 multiline
 comment
 */
```

Example:

```
publicclassCommentExample2{
publicstaticvoidmain(String[]args) {
/* Let's declare and
 printvariableinjava.*/
   inti=10;
   System.out.println(i);
}}
```

Output:

```
10
```

JavaDocumentationComment

ThedocumentationcommentisusedtocreatedocumentationAPI.TocreatedocumentationAPI, youneed to use **javadoc tool**.

Syntax:

```
/**
This
is
documentation
comment
*/
```

Example:

```
/**TheCalculatorclass providesmethodstogetadditionandsubtractionofgiven2 numbers.*/
publicclassCalculator{
/**Theadd()methodreturnsadditionofgivennumbers.*/
publicstaticintadd(inta,intb){returna+b;}
/**Thesub()methodreturnssubtractionofgivennumbers.*/
publicstaticintsub(inta,intb){returna-b;}
}
```

Compileitbyjavactool:

```
javacCalculator.java
```

CreateDocumentationAPIbyjavadoctool:

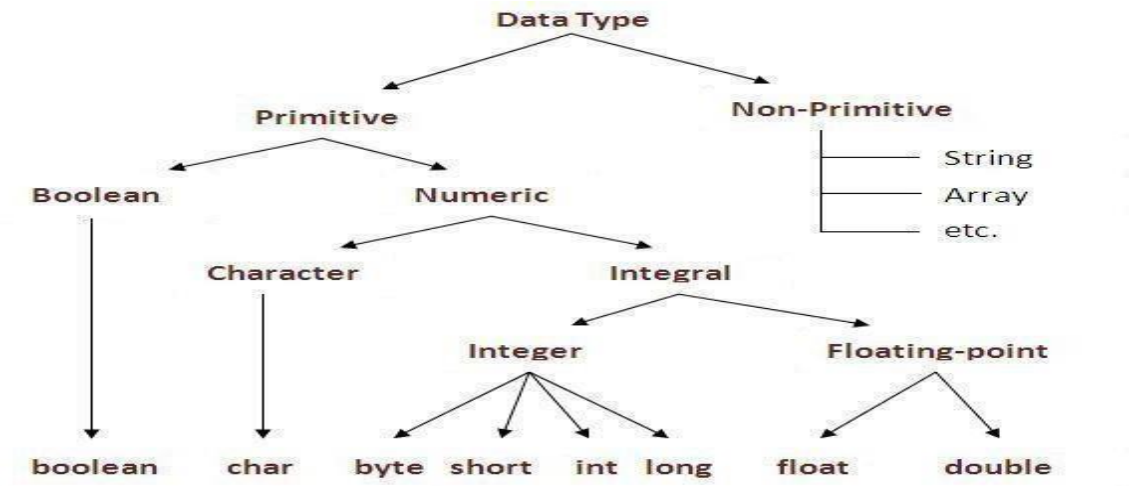
```
javadocCalculator.java
```

Now,therewillbeHTMLfilescreatedfor your Calculator classinthecurrentdirectory. OpentheHTML files and see the explanation of Calculator class provided through documentation comment.

DataTypes

Datatypesrepresentthedifferentvaluestobestoredinthevariable.Injava,therearetwotypes ofdatatypes:

- Primitivedatatypes
- Non-primitivedatatypes



DataType	DefaultValue	Defaultsize
boolean	False	1bit
char	'\u0000'	2byte
byte	0	1byte
short	0	2byte
int	0	4byte
long	0L	8byte
float	0.0f	4byte
double	0.0d	8byte

JavaVariableExample:AddTwoNumbers

```
classSimple{
publicstaticvoidmain(String[]args){
inta=10;
intb=10;
intc=a+b;
System.out.println(c);
}}
```

Output:20

VariablesandDataTypesinJava

Variableisanameofmemorylocation.Therearethreetypesofvariables injava: local, instance and static.

Therearetwotypesofdatatypesinjava:primitiveandnon-primitive.

TypesofVariable

Therearethreetypesofvariablesin java:

- localvariable
- instancevariable
- staticvariable

1) LocalVariable

Avariablewhichisdeclaredinsidethemethodiscalledlocalvariable.

2) InstanceVariable

Avariablewhichisdeclared insidetheclassbutoutsidethe method,iscalled instancevariable.It is not declared as static.

3) Staticvariable

Avariablethatisdeclaredasstaticiscalledstaticvariable.Itcannotbelocal. We

will have detailed learning of these variables in next chapters.

Exampletounderstandthetypesofvariablesin java

```
class A{
int data=50;//instance variable
static int m=100;//staticvariable
void method(){
int n=90;//localvariable
}
}//endofclass
```

ConstantsinJava

Aconstantisavariablewhichcannothave itsvaluechangedafterdeclaration. It usesthe**'final'keyword.**

Syntax

```
modifier final dataTypevariableName=value;//globalconstant

modifier staticfinal dataTypevariableName=value;//constantwithinac
```

ScopeandLifeTimeof Variables

The scope of a variable defines the section of the code in which the variable is visible. As a general rule, variables that are defined within a block are not accessible outside thatblock. The lifetime of a variable refers to how long the variable exists before it isdestroyed. Destroying variables refers to deallocating the memorythatwasallotted to the variables when declaring it. We have written a few classes till now. You might have observed that not all variables are the same. The ones declared in the body of a method were different from those that were declared in the class itself. There are three types of variables: instance variables, formal parameters or local variables and localvariables.

Instancevariables

Instance variables are those that are defined within a class itself and not in any method or constructor of the class. They are known as instance variables because every instance of the class(object) contains acopyofthese variables. The scopeofinstance variablesisdetermined bythe access specifier that is applied to these variables. We have alreadyseen aboutit earlier. The lifetime of these variables is the same as the lifetime of the object to which it belongs. Object once created do not exist for ever. They are destroyed bythe garbage collector of Java when there are no more reference to that object. We shall see about Java's automatic garbage collector later on.

Argumentvariables

These are the variables that are defined in the header oaf constructor or a method. The scope of these variables is the method or constructor in which they are defined. The lifetime is limited to the time for which the method keeps executing. Once the methodfinishes execution, these variables aredestroyed.

Localvariables

A localvariable is the one that is declared within a method or a constructor (not in the header). The scope and lifetime are limited to the methoditself.

One important distinction between these three types of variables is that access specifiers can be applied to instance variables onlyand not to argument or local variables.

In addition to the local variables defined in a method, we also have variables that are defined in bocks life an if block and an else block.The scope and is the same as that of the block itself.

Operatorsinjava

Operatorinjava isasymbolthat isusedto performoperations. Forexample:+,-,*,/etc. There are manytypes of operators in java which are given below:

- UnaryOperator,
- ArithmeticOperator,
- shiftOperator,
- RelationalOperator,
- BitwiseOperator,
- LogicalOperator,
- TernaryOperatorand
- AssignmentOperator.

OperatorsHierarchy

Operator Precedence

Operators	Precedence
postfix	`expr++ expr--`
unary	`++expr --expr +expr -expr ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Expressions

ExpressionsareessentialbuildingblocksofanyJavaprogram,usuallycreatedtoproduceanew value, although sometimes an expression simply assigns a value to a variable. Expressions are built using values, variables, operators and method calls.

TypesofExpressions

Whileanexpressionfrequentlyproducesaresult,it doesn't always.Therearethreetypesof expressions in Java:

- Thosethatproduceavalue,i.e.theresultof(1 +1)
- Thosethatassignavariable,forexample(v=10)
- Thosethathavenoresult butmight havea"sideeffect"becauseanexpressioncaninclude a wide range ofelements suchas method invocations or increment operatorsthat modify the state (i.e. memory) of aprogram.

JavaTypecastingandTypeconversion

WideningorAutomaticTypeConversion

Wideningconversiontakesplacewhentwodatatypesareautomaticallyconverted.Thishappens when:

- Thetwodatatypes arecompatible.
- Whenweassignvalueofasmallerdatatypetoabigger datatype.

For Example, in java the numeric data types are compatible with each other but no automatic conversion is supported from numeric type to char or boolean. Also, char and boolean are not compatible with each other.

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

NarrowingorExplicitConversion

Ifwewantto assignavalueoflargerdatatypetoasmallerdatatypeweperformexplicit type casting or narrowing.

- Thisisusefulforincompatibledatatypeswhereautomaticconversioncannot bedone.
- Here,target-typespecifiesthedesiredtypetoconvertthespecifiedvalueto.

Double –> Float –> Long –> Int –> Short –> Byte

- Narrowing or Explicit Conversion

JavaEnum

Enuminjavaisadatatype thatcontainsfixed setofconstants.

It can be used for days of the week (SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY,FRIDAYandSATURDAY),directions(NORTH,SOUTH,EASTandWEST) etc.Thejavaenumconstantsarestaticand finalimplicitly. It isavailable fromJDK1.5. Java

Enums can be thought of as classes that have fixed set of constants.

Simpleexampleofjavaenum

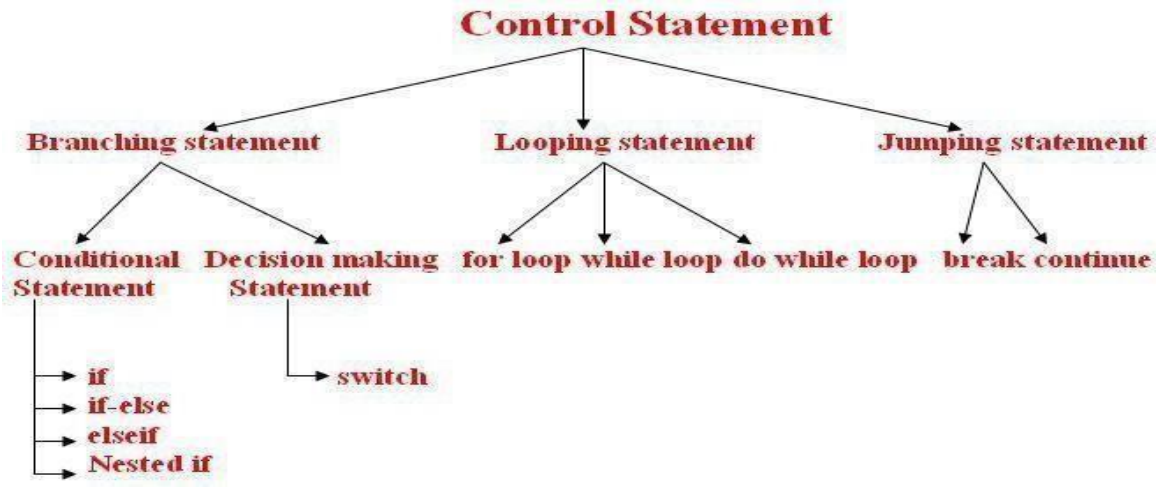
```
classEnumExample1{
publicenumSeason{WINTER,SPRING,SUMMER,FALL}

publicstaticvoidmain(String[]args){ for
(Season s : Season.values())
System.out.println(s);
}}
```

Output:

```
WINTER
SPRING
SUMMER
FALL
```

ControlFlowStatements



Java'sSelectionStatements

Java supports two selection statements: **if** and **switch**. These statements allow you to control the flow of your program‟s execution based upon conditions known only during run time.

if

The **if** statementis Java‟s conditional branch statement. It can be used to route programexecution through two different paths. Here is the general form of the **if** statement:

Syntax

```
if(condition)
    statement1;
else
    statement2;
```

Here,eachstatement maybeasingle statement oracompound statement enclosed in curlybraces (that is, a block). The condition is anyexpression that returns a **boolean** value. The **else** clause is optional.
The **if** works like this: Ifthe *condition* is true, then *statement1* is executed. Otherwise, *statement2* (if it exists) is executed. In no case will both statements be executed. For example, consider the following:

```
inta,b;
//
...if(a<b
)
   a=0;
else
    b=0;
```

Nestedifs

A*nested* **if** is an**if** statement that is the targetofanother **if**or **else**. Nested **if**s are verycommon in programming. When you nest **if**s, the main thing to remember is that an **else** statement always refersto the nearest **if** statement that is within the same block as the **else**and that is not already associated with an **else**. Here is an example:

```
if(i== 10)
{
    if(j<20)
        a = b;
    if(k>100)
         c=d;//thisifis else
     a=c;//associated withthiselse
}
else
    a= d;//thiselse referstoif(i== 10)
```

Theif-else-ifLadder

Acommonprogrammingconstructthatisbaseduponasequenceofnested**if**sisthe***if-else-if ladder***.Itlookslikethis:

Syntax:
if(*condition*)
statement;
elseif*(condition*)
statement;
elseif(*condition*)
statement;
...
else
statement;

switch

The **switch** statement is Java‟s multiway branch statement. It provides an easy way to dispatch execution to different parts of your code based on the value of an expression. As such, it often provides a better alternative than a large series of **if-else-if** statements.

Hereisthegeneralformofa**switch**statement:

switch (*expression*)

{

case*value1*:
//statementsequencebreak;
case *value2*:
//statementsequencebreak;
...
case*valueN*:
//statementsequencebreak; default:
//defaultstatementsequence
}

Example:

```
classSampleSwitch{
publicstaticvoidmain(Stringargs[]){for(int i=0;
i<6; i++)
switch(i) {case 0:
System.out.println("iiszero.");break;
case 1:
System.out.println("iisone.");break;
case 2:
System.out.println("iistwo.");break;
case 3:
System.out.println("iisthree.");break;
default:
System.out.println("iisgreaterthan3.");
}
}
}
```

Theoutputproducedbythisprogramisshownhere:iiszero. I is
one.i is two.
iisthree.

iisgreaterthan3.iis
greater than 3.

IterationStatements

Java'siterationstatementsare

for
while
do-while

Thesestatementsareusedtorepeatsamesetofinstructionsspecifiednumberoftimescalled loops.Alooprepeatedlyexecutesthesameset ofinstructionsuntilaterminationcondition ismet.

- **while Loop:** while loop repeats a group ofstatements as long as condition is true. Once the condition is false, the loop is terminated. In while loop, the condition is tested first; if it is true, then only the statements are executed. while loop is called as entry control loop.

Syntax:while(condition)
{
statements;
}

Program: Write aprogramtogeneratenumbersfrom1to20.

```
//Programto generatenumbersfrom1to20. class
Natural
{
    publicstaticvoidmain(Stringargs[])
     {
        inti=1;
        while(i<=20)
         {
          System.out.print(i+"\t"); i++;
         }
      }
}
```

do…while Loop: do…while loop repeats a group of statements as long as condition is true. In do...while loop, the statements are executed first and then the condition is tested. do…while loop is also called as exit control loop.

Syntax:

```
do
{
statements;
}while(condition);
```

Program: Write a program to generate numbers from 1 to 20.

```
//Programto generatenumbers from1to20. class
Natural
{
publicstaticvoidmain(Stringargs[])
{
inti=1;
do
{
System.out.print(i+"\t"); i++;
}while(i<= 20);
}
}
```

for Loop: The for loop is also same as do…while or while loop, but it is more compactsyntactically. The for loop executes a group ofstatements as long as a condition is true.

Syntax: for(expression1;expression2;expression3)
{statements;
}
Here,expression1isusedtoinitializethevariables,expression2isusedforcondition checking and expression3 is used for increment or decrement variable value.

Program: Write a program to generate numbers from 1 to 20.

```
//Programtogeneratenumbersfrom1to20. class
Natural
{
publicstaticvoidmain(Stringargs[])
{
inti;
for (i=1; i<=20; i++)
System.out.print(i+"\t");
}
}
```

JUMPSTATEMENTS

Java supports three jump statements: break, continue and return. These statementstransfer control to another part of the program.

- ***break:***

- breakcanbeusedinsidealoopto comeoutofit.
- breakcanbeusedinsidetheswitchblockto comeoutoftheswitchblock.
- breakcanbeusedinnestedblockstogototheendofablock.Nestedblocksrepresent a block written within another block.

Syntax: break;(or)breaklabel;//here labelrepresentsthenameoftheblock.

Program: Write aprogramtousebreak asacivilizedformofgoto.

```
//usingbreakasacivilizedformofgoto class
BreakDemo
{
 publicstaticvoidmain(Stringargs[])
{


booleant=true;
first:
{
second:
{
third:
{
System.out.println(“Beforethebreak”);
if(t)breaksecond;//breakoutofsecondblock
System.out.println (“This won‟t execute”);
}
System.out.println(“Thiswon‟texecute”);
}
System.out.println(“Thisisaftersecondblock”);
}
}
}
```

continue: Thisstatementisusefultocontinuethenextrepetitionofaloop/iteration.When continue is executed, subsequent statements inside the loop are not executed.

Syntax:continue;

Program: Write aprogramtogeneratenumbersfrom1to20.

```
//Programtogeneratenumbersfrom1to20. class
Natural
{
publicstaticvoid main(Stringargs[])
{
 int i=1;
while(true)
{
System.out.print(i+“\t”); i++;
if(i<=20)
```

continue;
else
break;
}
}
}

returnstatement:

- returnstatementisusefultoterminateamethodandcomebacktothecallingmethod.
- returnstatementinmainmethodterminatestheapplication.
- returnstatementcanbeusedtoreturnsomevaluefromamethodtoacallingmethod.

Syntax:return;(or)
returnvalue;//valuemaybe ofanytype

Program: Write aprogramtodemonstratereturnstatement.

```
//Demonstratereturn
class ReturnDemo
{
publicstaticvoidmain(Stringargs[])

{
 booleant =true;
System.out.println("Beforethereturn");
if (t)
return;
System.out.println("Thiswon"texecute");
}
}
```

CreatingaStand-AloneJavaApplication

1. Writeamainmethodthatruns yourprogram. Youcanwritethis methodanywhere. Inthis example, I'llwrite my main method in a class called Main that has no other methods. **For example:**

```
2.  publicclassMain
3.  {
4.    publicstaticvoidmain(String[]args)
5.    {
6.      Game.play();
7.    }}
```

8. Makesure yourcodeiscompiled, andthatyouhavetesteditthoroughly.
9. Ifyou're using Windows, you willneed to set yourpathto include Java, ifyou haven't done so already. This is a delicate operation. Open Explorer, and look inside C:\ProgramFiles\Java, and you should see some version ofthe JDK. Open this folder, and thenopenthebinfolder. Select thecompletepathfromthetopoftheExplorerwindow,and press Ctrl-C to copyit.

Next, find the "MyComputer" icon(on your Startmenu or desktop), right-click it, and select properties.Clickon theAdvancedtab,andthen clickon theEnvironmentvariablesbutton. Lookatthevariableslistedforallusers,andclickonthePathvariable.Donotdeletethe contentsofthisvariable!Instead,editthecontentsbymovingthecursortotherightend, enteringasemicolon(;),andpressingCtrl-Vtopastethepathyoucopiedearlier.Thengo aheadandsaveyourchanges.(Ifyouhaveany Cmdwindowsopen,youwillneedtoclose them.)

10. Ifyou'reusing Windows, gototheStart menu and type"cmd"torunaprogramthat brings up a command prompt window. Ifyou're using a Mac orLinux machine, runthe Terminal program to bring up a command prompt.
11. InWindows,typedir atthecommand prompt to listthecontentsofthecurrent directory. On a Mac or Linux machine, type ls to do this.
12. Now we want to change to the directory/folder that contains your compiled code. Look at the listing of sub-directories within this directory, and identify which one contains your code. Type cd followed by the name of that directory, to change to that directory. For example, to change to a directory called Desktop, you would type:

cdDesktop

Tochangetotheparentdirectory,type:

cd..

Everytimeyouchangetoanewdirectory, listthecontentsofthat directorytoseewhereto go next. Continuelisting and changing directories until you reach the directorythatcontains your .class files.

13. Ifyoucompiledyour programusingJava1.6,butplantorunitonaMac, you'llneedto recompile your code from the command line, bytyping:

javac-target1.5*.java

14. Nowwe'llcreateasingleJARfilecontaining allofthefilesneeded torunyourprogram.

Arrays

Java provides a data structure, the **array**, which stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and...,numbers[99] to represent individual variables.

This tutorial introduces how to declare array variables, create arrays, and process arrays using indexed variables.

DeclaringArrayVariables:

To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable:

```
dataType[]arrayRefVar;//preferredway. or
dataTypearrayRefVar[];//worksbutnotpreferredway.
```

Note: Thestyle**dataType[]arrayRefVar** ispreferred.Thestyle**dataTypearrayRefVar[]** comesfromtheC/C++languageandwasadoptedinJavatoaccommodateC/C++programmers.

Example:
Thefollowingcodesnippetsareexamplesofthissyntax:

```
double[]myList;ordoublemyList[];
CreatingArrays:
/preferredway.
```

//worksbutnotpreferredway.

Youcancreateanarraybyusingthenewoperatorwiththefollowingsyntax:

```
arrayRefVar=newdataType[arraySize];
```

Theabovestatementdoestwothings:

- Itcreatesanarrayusingnew dataType[arraySize];
- Itassignsthe referenceofthenewlycreated arraytothe variablearrayRefVar.

Declaringanarrayvariable,creatinganarray,and assigning thereferenceofthearraytothe variable can be combined in one statement, as shown below:

```
dataType[]arrayRefVar=newdataType[arraySize];
```

Alternativelyyou cancreatearraysasfollows:

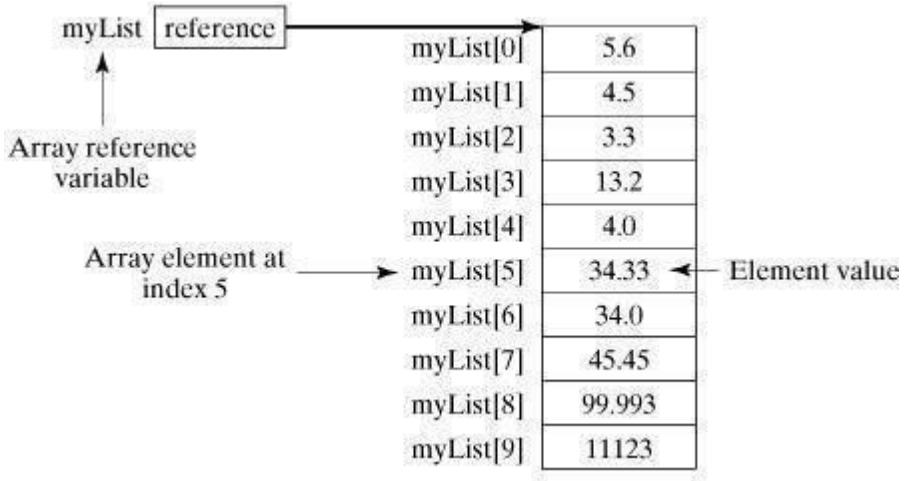
```
dataType[]arrayRefVar={value0,value1,...,valuek};
```

Thearrayelementsareaccessedthroughthe**index**.Arrayindicesare0-based;that is, theystart from 0 to **arrayRefVar.length-1**.

Example:
Followingstatement declaresanarrayvariable, myList, createsanarrayof10elementsof double type and assigns its reference tomyList:

```
double[]myList=newdouble[10];
```

Followingpicturerepresentsarraymy List.Here,myList holdstendoublevaluesandtheindices are from 0 to 9.



ProcessingArrays:

Whenprocessingarrayelements, weoftenuseeither for loopor foreachloopbecauseallofthe elements in an arrayare ofthe same type and the size of the array is known.

Example:

Hereisacompleteexampleofshowinghowtocreate,initializeand processarrays:

```
publicclassTestArray
{
  public static void main(String[] args) {
    double[]myList={1.9,2.9,3.4,3.5};
          // Print all the array elements
    for(inti=0;i<myList.length;i++){
      System.out.println(myList[i]+"");
    }
    //Summingallelements double
    total = 0;
    for(inti=0;i<myList.length;i++){ total
      += myList[i];
    }
    System.out.println("Totalis"+total);
    //Findingthelargestelement
    double max = myList[0];
    for(inti=1;i<myList.length;i++){ if (myList[i]
      >max) max = myList[i];
    }
    System.out.println("Maxis"+max);
  }
}
```

Thiswouldproducethefollowing result:

```
1.9
2.9
3.4
3.5
Totalis 11.7
Maxis3.5
```

```
publicclassTestArray{
public static void main(String[] args) {
    double[]myList={1.9,2.9,3.4,3.5};
 // Print all the array elements
    for(doubleelement:myList){
       System.out.println(element);
}}}
```

JavaConsoleClass

TheJavaConsoleclassisbeusedtogetinput fromconsole.Itprovides methodstoreadtextsand passwords.

IfyoureadpasswordusingConsoleclass,itwillnotbedisplayedtotheuser.

Thejava.io.Consoleclassisattachedwithsystemconsole internally.TheConsoleclassis introduced since 1.5.

Let'sseeasimpleexampleto readtextfromconsole.

1. Stringtext=System.console().readLine();
2. System.out.println("Textis:"+text);

JavaConsoleExample

```
importjava.io.Console;
classReadStringTest{
public static void main(String args[]){
Console c=System.console();
System.out.println("Enter your name: ");
String n=c.readLine();
System.out.println("Welcome"+n);     }    }
```

Output

```
Enteryourname:NakulJain
WelcomeNakulJain
```

Constructors

Constructorinjavaisa*specialtypeofmethod*thatisusedtoinitializetheobject.

Javaconstructoris*invokedatthetimeofobjectcreation*.Itconstructsthevalues i.e. providesdata for the object that is why itis known as constructor.

Therearebasicallytworulesdefinedfortheconstructor.

1. Constructorname must besameasitsclassname
2. Constructormusthavenoexplicitreturntype

Typesofjavaconstructors

Therearetwotypesofconstructors:

1. Defaultconstructor(no-argconstructor)
2. Parameterizedconstructor

JavaDefaultConstructor

Aconstructorthathavenoparameterisknownasdefaultconstructor.

Syntaxofdefaultconstructor:

```
1.<class_name>(){}
```

Exampleofdefaultconstructor

Inthisexample,wearecreatingtheno-argconstructorintheBikeclass. Itwillbe invokedat the time of object creation.

```
classBike1{
Bike1(){System.out.println("Bikeiscreated");}
publicstaticvoidmain(Stringargs[]){
Bike1 b=new Bike1();
}}
```

```
Output:Bikeiscreated
```

ExampleofparameterizedconstructorInthisexample,wehavecreatedtheconstructorofStudentclassthat havetwoparameters. We can have any number of parameters in the constructor.

```
classStudent4{
intid;
Stringname;

Student4(inti,Stringn){
id = i;
name=n;
}
voiddisplay(){System.out.println(id+""+name);}

public static void main(String args[]){
Student4s1=newStudent4(111,"Karan");
Student4s2=newStudent4(222,"Aryan");
s1.display();
s2.display();
}}
```

Output:

```
111Karan
222Aryan
```

ConstructorOverloadinginJava

Constructor overloading is a technique in Java in which a class can have any number of constructorsthatdiffer inparameterlists.Thecompilerdifferentiatestheseconstructorsby taking into account the number ofparameters in the list and their type.

ExampleofConstructorOverloading

```
classStudent5{
intid; String
name;
intage;
Student5(inti,Stringn){
id = i;
name=n;
}
Student5(inti,Stringn,inta){
id = i;
name=n;
age=a;
}
voiddisplay(){System.out.println(id+""+name+""+age);}

public static void main(String args[]){
Student5 s1 = new Student5(111,"Karan");
Student5s2=newStudent5(222,"Aryan",25);
s1.display();
```

```
s2.display();
}}
```

Output:

```
111Karan0
222Aryan25
```

JavaCopyConstructor

Thereis no copyconstructorinjava.But,wecancopythevaluesofoneobjectto anotherlike copy constructor inC++.

Therearemanywaystocopythevaluesofoneobjectintoanotherin java.Theyare:

- Byconstructor
- Byassigningthe valuesofoneobjectintoanother
- Byclone()methodofObjectclass

Inthisexample,wearegoingtocopythe valuesofoneobject into anotherusing java constructor.

```
classStudent6{
intid;
Stringname;
Student6(inti,Stringn){
id = i;
name=n;
}

Student6(Student6s){
id = s.id;
name=s.name;
}
voiddisplay(){System.out.println(id+""+name);}

public static void main(String args[]){
Student6s1=newStudent6(111,"Karan");
Student6 s2 = new Student6(s1);
s1.display();
s2.display();
}}
```

Output:

```
111Karan
111Karan
```

Java-Methods

A Java method is a collection of statements that are grouped together to perform an operation. When you call the System.out.**println()** method, for example, the system actually executes several statements in order to display a message on the console.

Nowyou willlearnhowto createyourownmethodswithorwithout returnvalues, invoke a method with or without parameters, and apply method abstraction in the program design.

CreatingMethod

Consideringthefollowingexampletoexplainthesyntaxofamethod−

Syntax

```
publicstaticintmethodName(inta,intb){
  //body
}
```

Here,

- **publicstatic**−modifier
- **int**−returntype
- **methodName**−nameofthemethod
- **a,b**−formalparameters
- **inta,intb**−list ofparameters

Methoddefinitionconsistsofamethodheaderandamethodbody.Thesame isshowninthe following syntax −

Syntax

```
modifierreturnTypenameOfMethod(ParameterList) {
  //method body
}
```

Thesyntaxshownaboveincludes−

- **modifier**−Itdefinestheaccesstypeofthemethodanditisoptionaltouse.
- **returnType**−Methodmayreturnavalue.
- **nameOfMethod**−Thisisthe methodname.Themethodsignatureconsistsofthemethod name and the parameter list.

- **ParameterList**−The list ofparameters, it isthe type, order, and number ofparameters of a method. These are optional,method may contain zeroparameters.
- **methodbody**−Themethod bodydefineswhatthemethod doeswiththestatements.

CallbyValueandCallbyReferenceinJava

There isonlycallbyvalue in java, not callbyreference. Ifwecalla method passing avalue, it is known as call by value. The changes being done in the called method, is not affected in the calling method.

Exampleofcallbyvalue injava

Incaseofcallbyvalueoriginalvalueisnotchanged.Let'stakeasimpleexample:

```
classOperation{
 intdata=50;
 voidchange(intdata){
 data=data+100;//changeswillbeinthelocalvariableonly
 }
 public static void main(String args[]){
   Operation op=new Operation();
   System.out.println("before change "+op.data);
   op.change(500);
   System.out.println("afterchange"+op.data);
 }
}
```

```
Output:beforechange50
       afterchange 50
```

InJava,parametersarealwayspassedbyvalue.Forexample,followingprogramprints i = 10, j = 20.

```
//Test.java
classTest{
   //swap()doesn'tswapiandj
   publicstaticvoidswap(Integeri,Integerj){
      Integer temp = new Integer(i);
      i=j;
      j=temp;
   }
   publicstaticvoidmain(String[]args){
      Integer i = new Integer(10);
      Integerj=newInteger(20);
      swap(i, j);
      System.out.println("i=" +i+ ",j="+j);
```

```
  }
}
```

StaticFieldsandMethods

The **static keyword** in java is used for memory management mainly. We can apply java static keywordwithvariables, methods, blocksandnestedclass. Thestatickeywordbelongstotheclass than instance of the class.

Thestatic canbe:

1. variable(alsoknownasclassvariable)
2. method(alsoknownasclassmethod)
3. block
4. nestedclass

Javastaticvariable

Ifyoudeclareanyvariableasstatic,itisknownstatic variable.

- Thestaticvariablecanbeusedtorefer thecommonpropertyofallobjects (that is not uniquefor each object) e.g. company name of employees,college name of studentsetc.
- Thestaticvariablegetsmemoryonlyonceinclassareaatthetimeof classloading.

Advantageofstaticvariable

Itmakesyour program**memoryefficient**(i.eitsavesmemory).

Understandingproblemwithoutstaticvariable

```
1. class Student{
2.     int rollno;
3.     String name;
4.     String college="ITS";
5.}
```

Exampleofstaticvariable

```
//Programofstaticvariable
class Student8{
  int rollno;
```

```
  Stringname;
  staticStringcollege="ITS";
  Student8(int r,String n){
  rollno =r;
  name=n;
  }
voiddisplay(){System.out.println(rollno+""+name+""+college);}
public static void main(String args[]){
Student8s1=newStudent8(111,"Karan");
Student8s2=newStudent8(222,"Aryan");

s1.display();
s2.display();
}}
```

```
Output:111KaranITS
       222AryanITS
```

Javastaticmethod

Ifyouapplystatickeywordwithanymethod,itisknownasstaticmethod.

- Astaticmethod belongstotheclassratherthanobjectofaclass.
- Astaticmethodcanbeinvokedwithouttheneedforcreating aninstance ofaclass.
- staticmethodcanaccessstaticdatamemberandcanchangethevalueofit.

Exampleofstaticmethod

```
//Programofchangingthecommonpropertyofallobjects(staticfield).

classStudent9{
    int rollno;
    Stringname;
    staticStringcollege="ITS";
    static void change(){
    college = "BBDIT";
    }
    Student9(intr,Stringn){
    rollno =r;
    name=n;
```

```
}
 voiddisplay(){System.out.println(rollno+""+name+""+college);}
publicstaticvoidmain(Stringargs[]){
Student9.change();
Student9 s1 = new Student9 (111,"Karan");
Student9 s2= new Student9 (222,"Aryan");
Student9s3  =new  Student9(333,"Sonoo");
s1.display();
s2.display();
s3.display();
}}
```

```
Output:111KaranBBDIT 222
    Aryan BBDIT
     333SonooBBDIT
```

Javastaticblock

- Isusedtoinitializethestatic datamember.
- Itisexecuted beforemainmethod atthetimeofclassloading.

Exampleofstaticblock

```
class A2{
 static{System.out.println("static block is invoked");}
 public static void main(String args[]){
 System.out.println("Hello main");
 }}
```

```
Output:staticblockisinvoked Hello
    main
```

AccessControl

AccessModifiersinjava

Therearetwotypesofmodifiersinjava:**accessmodifiers**and**non-accessmodifiers**.

Theaccessmodifiers injavaspecifiesaccessibility(scope)ofadatamember, method,constructor or class.

Thereare4typesofjavaaccessmodifiers:

1. private
2. default
3. protected
4. public

privateaccessmodifier

Theprivateaccessmodifierisaccessibleonlywithinclass.

Simpleexampleofprivateaccessmodifier

In this example, we have created two classes A and Simple. A class contains private data member and private method. We are accessing these private members from outside the class, so there is compile time error.

```
classA{
privateintdata=40;
privatevoidmsg(){System.out.println("Hellojava");}}
publicclassSimple{
 publicstaticvoidmain(Stringargs[]){
  A obj=new A();
  System.out.println(obj.data);//Compile Time Error
  obj.msg();//Compile Time Error
  }}
```

2) defaultaccessmodifier

Ifyoudon't useanymodifier, it istreatedas**default** bydefault. Thedefault modifier is accessible only withinpackage.

Exampleofdefaultaccessmodifier

In this example, we have created two packages pack and mypack. We are accessing theA class from outside its package, since A class is not public, so it cannot be accessed from outside thepackage.

```
//savebyA.java
package pack;
class A{
  voidmsg(){System.out.println("Hello");}
}

//save by B.java
packagemypack;
import pack.*;
```

```
class B{
  public static void main(String args[]){
   Aobj=new      A();//CompileTimeError
   obj.msg();//Compile Time Error }}
```

Intheaboveexample,thescopeofclassAand itsmethodmsg() isdefault so it cannot be accessed from outside thepackage.

3) protectedaccessmodifier

The**protectedaccessmodifier**isaccessiblewithinpackageandoutsidethepackagebutthrough inheritance only.

Theprotectedaccessmodifiercanbeappliedonthedatamember,methodandconstructor.Itcan't be applied on the class.

Exampleofprotectedaccessmodifier

In this example, we have created the two packages pack andmypack. The A class of pack package is public, so can be accessed fromoutside the package. Butmsg method ofthis package is declared as protected, so it can be accessed fromoutside the class onlythroughinheritance.

```
//savebyA.java
package pack;
publicclassA{
protectedvoidmsg(){System.out.println("Hello");}}
//savebyB.java
package mypack;
import pack.*;
classBextendsA{
  publicstaticvoidmain(Stringargs[]){ B
   obj = new B();
   obj.msg();
  }}
```

Output:Hello

4) publicaccessmodifier

The**publicaccessmodifier**isaccessibleeverywhere.Ithasthewidest scopeamongallother modifiers.

Exampleofpublicaccessmodifier

```
//savebyA.java
package pack;
publicclassA{
publicvoidmsg(){System.out.println("Hello");}}
//save by B.java
packagemypack;
import pack.*;
class B{
  publicstaticvoidmain(Stringargs[]){ A
   obj = new A();
   obj.msg();
  }}
```

Output:Hello

Understandingalljavaaccessmodifiers

Let'sunderstand theaccessmodifiersbyasimpletable.

Access Modifier	within class	within package	outsidepackageby subclassonly	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

thiskeywordinjava

Usageofjavathiskeyword

Hereisgiventhe6usageofjavathiskeyword.

1. thiscanbeusedtorefercurrentclassinstancevariable.
2. thiscanbeusedtoinvokecurrentclassmethod(implicitly)
3. this() canbeusedto invokecurrentclassconstructor.

4. thiscanbepassedasanargumentinthemethodcall.
5. thiscanbepassedasargumentintheconstructorcall.
6. thiscanbeusedtoreturnthecurrentclassinstancefromthemethod.

```
classStudent{
int rollno;
String name;
float fee;
Student(introllno,Stringname,floatfee){
this.rollno=rollno;
this.name=name;
this.fee=fee;
}
voiddisplay(){System.out.println(rollno+""+name+""+fee);}
}
classTestThis2{
public static void main(String args[]){
Student s1=new Student(111,"ankit",5000f);
Students2=newStudent(112,"sumit",6000f);
s1.display();
s2.display();
}}
```

Output:
111 ankit5000
112 sumit6000

ConstructorOverloadinginJava

Constructor overloading is a technique in Java in which a class can have any number ofconstructors that differ in parameter lists.The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

ExampleofConstructorOverloading

```
classStudent5{int
id; String name;
intage;
Student5(inti,Stringn){id= i;
name=n;
}
Student5(inti,Stringn,inta){id= i;
name=n;age=a;
}
voiddisplay(){System.out.println(id+""+name+""+age);}
```

```
  publicstaticvoidmain(Stringargs[]){Student5s1
  =newStudent5(111,"Karan");Student5s2=
  newStudent5(222,"Aryan",25); s1.display();
  s2.display();
 }
}
```

Output:

```
111Karan0
222Aryan25
```

MethodOverloadinginjava

Ifaclasshasmultiplemethodshavingsamenamebut differentinparameters,it isknown as **Method Overloading**.

Ifwehavetoperformonlyoneoperation,havingsame nameofthemethodsincreasesthe readability of the program.

MethodOverloading:changingno.ofarguments

Inthisexample,wehavecreatedtwomethods,first add()methodperformsadditionoftwo numbers and second add method performs addition of three numbers.

Inthisexample,wearecreatingstaticmethodssothat wedon't needto createinstance forcalling methods.

```
classAdder{
staticintadd(inta,intb){returna+b;}
staticintadd(inta,intb,intc){returna+b+c;}
}
classTestOverloading1{
public static void main(String[] args){
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(11,11,11));
}}
```

Output:

```
22
33
```

MethodOverloading:changingdatatypeofarguments

Inthisexample, wehavecreatedtwomethodsthatdiffers indatatype. The first addmethod receives two integer arguments and second add method receives two double arguments.

RecursioninJava

Recursioninjava isaprocessinwhicha methodcalls itselfcontinuously.Amethodinjavathat calls itself is called recursive method.

JavaRecursionExample1:FactorialNumber

```
publicclassRecursionExample3{
  staticintfactorial(intn){
   if(n==1)
     return1;
     else
    return(n*factorial(n-1));
    }}
public static void main(String[] args) {
System.out.println("Factorialof5is:"+factorial(5));
}}
```

Output:

```
Factorialof5is:120
```

JavaGarbageCollection

Injava,garbagemeansunreferencedobjects.

GarbageCollectionisprocessofreclaimingtheruntimeunused memoryautomatically. Inother words, it is a way to destroy the unused objects.

Todo so,wewereusing free()functioninClanguageanddelete()inC++.But,injava it is performed automatically. So, java provides better memory management.

AdvantageofGarbageCollection

- Itmakesjava**memoryefficient**becausegarbagecollectorremovestheunreferenced objects from heapmemory.
- Itis**automaticallydone**bythegarbagecollector(apartofJVM) so wedon't needtomake extraefforts.

gc()method

Thegc()method isusedto invokethegarbagecollectorto performcleanupprocessing.The gc() is found in System and Runtime classes.

```
publicstaticvoidgc(){}
```

SimpleExampleofgarbagecollectioninjava

```
public class TestGarbage1{
 publicvoidfinalize(){System.out.println("objectisgarbagecollected");}
 public static void main(String args[]){
  TestGarbage1 s1=new TestGarbage1();
  TestGarbage1 s2=new TestGarbage1();
  s1=null;
  s2=null;
  System.gc();
 }}
```

```
objectisgarbagecollected
objectisgarbagecollected
```

JavaString

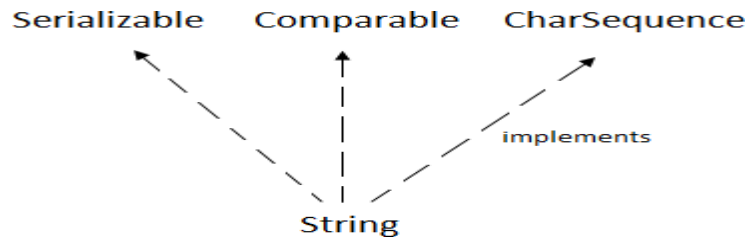
stringisbasicallyanobjectthat representssequenceofchar values. Anarrayofcharactersworks same as java string. For example:

```
1. char[]ch={'j','a','v','a','t','p','o','i','n','t'};
2. Strings=newString(ch);
```

ssame as:

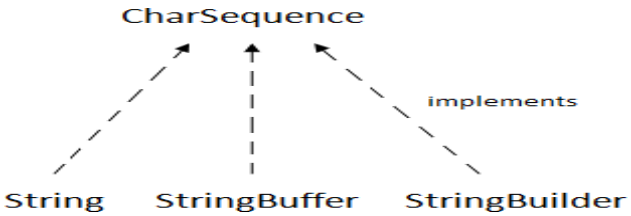
```
1.  Strings="javatpoint";
```

2. **Java String** class provides a lot of methods to perform operations on string such as compare(), concat(),equals(),split(),length(),replace(),compareTo(),intern(),substring() etc.
3. Thejava.lang.String
4. classimplements*Serializable*,*Comparable*and*CharSequence*interfaces.



CharSequenceInterface

The CharSequence interface is used to represent sequence of characters. It is implemented by String, StringBuffer and StringBuilder classes. It means, we can create string in java by using these 3 classes.



The java String is immutable i.e. it cannot be changed. Whenever we change any string,anewinstanceiscreated.Formutablestring,youcanuseStringBufferandStringBuilder classes. Therearetwowaystocreate Stringobject:

1. Bystringliteral
2. Bynewkeyword

StringLiteral

JavaStringliteraliscreatedbyusingdoublequotes.ForExample:

```
1. Strings="welcome";
```

Each time you create a string literal, the JVM checks the string constant pool first. If the string already exists in the pool, a reference to the pooled instance is returned. If string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

```
1.  Strings1="Welcome";
2.  Strings2="Welcome";//willnotcreatenewinstance
```

Bynew keyword

```
1. Strings=newString("Welcome");//createstwoobjectsandonereferencevariable
```

In such case, JVM will create a new string object in normal (non pool) heap memory and the literal"Welcome" willbe placed in the string constant pool. The variable s will refer to theobject in heap (non pool).

```
JavaStringExample
publicclassStringExample{
publicstaticvoid main(Stringargs[]){
Strings1="java";//creatingstringbyjavastring literal
charch[]={'s','t','r','i','n','g','s'};
Strings2=newString(ch);//convertingchararraytostring
Strings3=newString("example");//creatingjavastringbynewkeyword
System.out.println(s1);
System.out.println(s2);
System.out.println(s3);
}}
```

```
java

strings
example
```

ImmutableStringinJava

Injava,**stringobjectsareimmutable**.Immutablesimplymeansunmodifiableorunchangeable.

Oncestring object iscreated itsdataor statecan't bechanged but anewstring object iscreated. Let's

tryto understand the immutability concept by the example given below:

```
classTestimmutablestring{
  publicstaticvoidmain(Stringargs[]){
    String s="Sachin";
    s.concat(" Tendulkar");//concat() method appends the string at the end
    System.out.println(s);//willprintSachinbecausestringsareimmutableobjects
  }}
```

Output:Sachin

```
classTestimmutablestring1{
  publicstaticvoidmain(Stringargs[]){
    String s="Sachin";
    s=s.concat("Tendulkar");
    System.out.println(s);
  }}
```

Output:Sachin Tendulkar

Unit-2

InheritanceinJava

Inheritanceinjavaisa mechanisminwhichoneobject acquiresalltheproperties and behaviors ofparent object. Inheritance represents the **IS-A relationship**, also known as *parent-child*relationship.

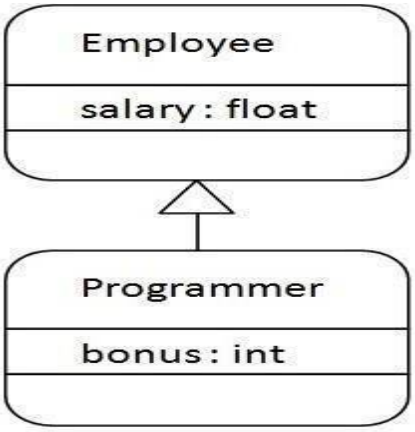
Whyuseinheritanceinjava

- ForMethodOverriding(so runtimepolymorphismcanbeachieved).
- ForCodeReusability.

SyntaxofJavaInheritance

```
1. classSubclass-nameextendsSuperclass-name
2. {
3.     //methodsand
fields4. }
```

The**extendskeyword** indicatesthat youaremakinganewclassthat derives fromanexisting class. The meaning of "extends" is to increase the functionality.

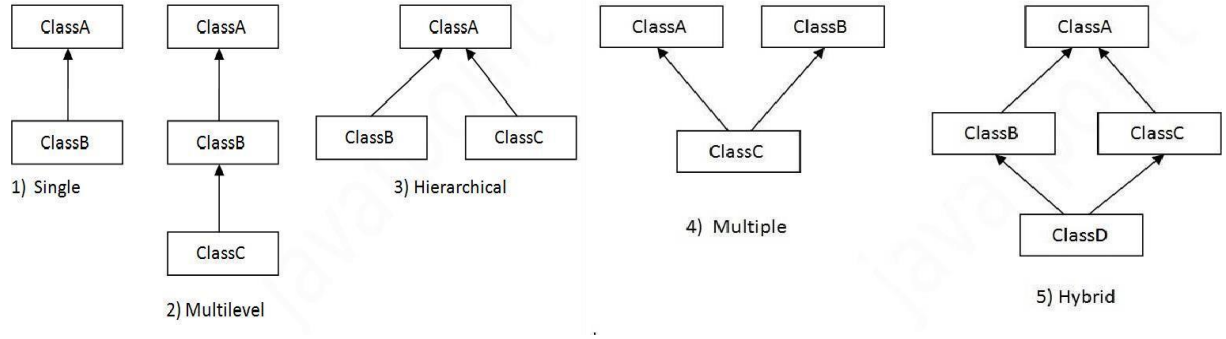



```
classEmployee{
 floatsalary=40000;
}
classProgrammerextendsEmployee{
 intbonus=10000;
 publicstaticvoidmain(Stringargs[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus ofProgrammer is:"+p.bonus);
}}
```

```
Programmersalaryis:40000.0
```

Bonusofprogrammeris:10000

Typesofinheritanceinjava



SingleInheritanceExample

File: TestInheritance.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextendsAnimal{
voidbark(){System.out.println("barking...");}
}
classTestInheritance{
publicstaticvoidmain(Stringargs[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:
barking...
eating...

MultilevelInheritanceExample

File: TestInheritance2.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextendsAnimal{
voidbark(){System.out.println("barking...");}
}
classBabyDogextendsDog{
voidweep(){System.out.println("weeping...");}
}
classTestInheritance2{
```

```
publicstaticvoidmain(Stringargs[]){
BabyDog d=new BabyDog();
d.weep();
d.bark();
d.eat();
}}
```

Output:

```
weeping...
barking...
eating...
```

HierarchicalInheritanceExample

File: TestInheritance3.java

```
classAnimal{
voideat(){System.out.println("eating...");}
}
classDogextendsAnimal{
voidbark(){System.out.println("barking...");}
}
classCatextendsAnimal{
voidmeow(){System.out.println("meowing...");}
}
classTestInheritance3{
publicstaticvoidmain(Stringargs[]){
Cat c=new Cat();
c.meow();
c.eat();
//c.bark();//C.T.Error
}}
```

Output:

```
meowing...
eating...
```

MemberaccessandInheritance

Asubclassincludesallofthe membersofitssuper classbut it cannot accessthose membersof the super class that have been declared as private. Attempt to access a private variable would causecompilationerrorasit causesaccessviolation.Thevariablesdeclaredasprivate,isonly accessible by other members of its own class. Subclass have no access to it.

superkeywordinjava

The**super**keywordinjava isareferencevariablewhichisusedtorefer immediateparent class object.

Wheneveryoucreatetheinstanceofsubclass, aninstanceofparent class iscreatedimplicitly which is referred by super reference variable.

UsageofjavasuperKeyword

1. supercanbeusedtoreferimmediateparentclass instancevariable.
2. supercanbeusedtoinvokeimmediateparentclassmethod.
3. super()canbeusedtoinvokeimmediateparentclassconstructor.

superisusedtoreferimmediateparentclassinstancevariable.

```
classAnimal{
Stringcolor="white";
}
classDogextendsAnimal{
Stringcolor="black";
voidprintColor(){System.out.println(color);//prints
color of Dogclass
System.out.println(super.color);//printscolorofAnimalclass
}
}
classTestSuper1{
publicstaticvoidmain(Stringargs[]){ Dog
d=new Dog();
```

```
d.printColor();
}}
```

Output:

```
black
white
```

FinalKeywordinJava

The **final keyword** injava is usedtorestrict theuser.Thejava final keywordcanbeusedin manycontext. Final can be:

1. variable
2. method
3. class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only. The blank final variable can be static also which will be initialized in the static block only.

ObjectclassinJava

The**Objectclass** istheparent classofalltheclassesinjavabydefault.Inotherwords,it isthe topmost class of java.

TheObjectclass isbeneficialifyouwanttoreferanyobject whosetypeyoudon't know.Notice that parent class reference variable can refer the child class object, know as upcasting.

Let'stakeanexample, thereisgetObject()methodthat returnsanobject butit canbeofanytype like Employee,Student etc, we canuse Object class reference to refer that object. For example:

```
1. Objectobj=getObject();//wedon'tknowwhatobjectwillbereturnedfromthismethod
```

TheObject classprovidessomecommonbehaviorsto alltheobjectssuchasobject canbe compared, object can be cloned, object can be notified etc.

MethodOverridinginJava

Ifsubclass(childclass)hasthesamemethodasdeclaredintheparent class, it isknown as **method overriding in java**.

UsageofJavaMethodOverriding

- Methodoverridingisusedtoprovidespecific implementationofa methodthatisalready provided by its superclass.
- Methodoverridingisused forruntimepolymorphism

RulesforJavaMethodOverriding

1. methodmusthavesamenameas intheparentclass
2. methodmusthavesameparameterasintheparentclass.
3. mustbeIS-Arelationship(inheritance).

Exampleofmethodoverriding

```
Class Vehicle{
voidrun(){System.out.println("Vehicleisrunning");}
}
classBike2 extends Vehicle{
voidrun(){System.out.println("Bike is running safely");}
public static void main(String args[]){
Bike2obj=newBike2();
obj.run();
}
```

Output:Bikeisrunningsafely

```
1.classBank{
intgetRateOfInterest(){return0;}
}
classSBIextendsBank{
intgetRateOfInterest(){return8;}
}
 classICICIextendsBank{
intgetRateOfInterest(){return7;}
}
classAXISextends Bank{
intgetRateOfInterest(){return9;}
}
classTest2{
publicstaticvoidmain(Stringargs[]){
SBI s=new SBI();
ICICIi=newICICI();
AXISa=newAXIS();
System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
System.out.println("ICICIRateofInterest:"+i.getRateOfInterest());
System.out.println("AXISRateofInterest:"+a.getRateOfInterest());
}}
```

```
Output:
SBIRateofInterest:8
```

```
ICICI RateofInterest:7
AXISRateofInterest:9
```

AbstractclassinJava

A class that is declared with abstract keyword is known as abstract class in java. It can have abstract and non-abstract methods (method with body). It needs to be extended and its method implemented. It cannot be instantiated.

Exampleabstractclass

```
1.abstractclassA{}
```

abstractmethod

```
1.abstractvoidprintStatus();//nobodyandabstract
```

Exampleofabstractclassthathasabstractmethod

```
abstract classBike{
  abstractvoidrun();
}
classHonda4 extends Bike{
voidrun(){System.out.println("runningsafely..");}
public static void main(String args[]){
 Bikeobj=newHonda4();
 obj.run();
}
1.}
```

```
runningsafely..
```

InterfaceinJava

An**interfacein java**isablueprintofaclass.Ithasstaticconstantsandabstractmethods.

The interface in java is **a mechanismto achieveabstraction**. Therecanbeonlyabstract methods in the java interface not method body. It is used to achieve abstractionand multiple inheritance in Java.

JavaInterfacealso**representsIS-Arelationship**. It

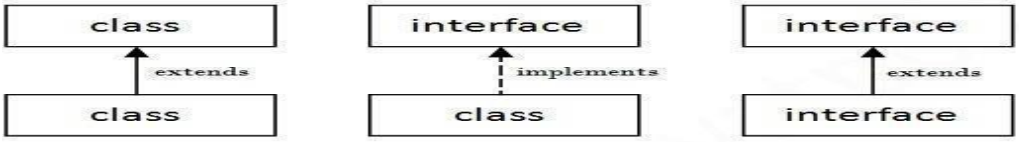
cannot be instantiated just like abstract class.

Therearemainlythreereasonstouseinterface.Theyaregivenbelow.

- Itisusedto achieveabstraction.
- Byinterface,wecansupportthe functionalityofmultipleinheritance.
- Itcanbeusedtoachieveloosecoupling.

Internaladditionbycompiler

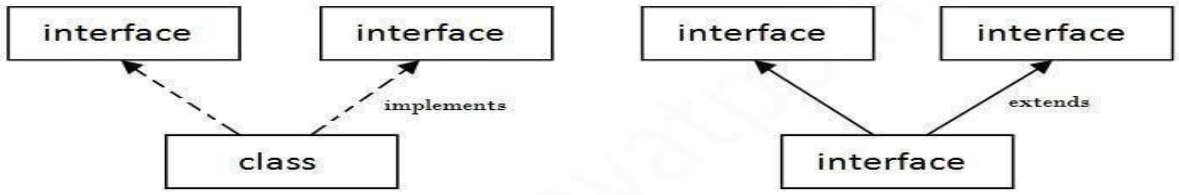
Understandingrelationshipbetweenclassesandinterfaces




```
//Interfacedeclaration:byfirstuser
interfaceDrawable{
voiddraw();
}
//Implementation:byseconduser
classRectangleimplementsDrawable{
publicvoiddraw(){System.out.println("drawingrectangle");}
}
classCircleimplementsDrawable{
publicvoiddraw(){System.out.println("drawingcircle");}
}
//Usinginterface:bythirduser
classTestInterface1{
publicstaticvoidmain(Stringargs[]){
Drawabled=newCircle();//Inrealscenario,objectisprovidedbymethode.g.getDrawable() d.draw();
}}
```

Output:drawingcircle

MultipleinheritanceinJavabyinterface



Multiple Inheritance in Java

```
interfacePrintable{
```

```
voidprint();
}
interfaceShowable{
voidshow();
}
classA7implementsPrintable,Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}
public static void main(String args[]){
A7obj=newA7();
obj.print();
obj.show();
 }}
```

Output:Hello
Welcome

Abstract class	Interface
1)Abstractclasscan**haveabstract andnon-abstract**methods.	Interfacecanhave**onlyabstract**methods.Since Java 8, it can have **default and static methods**also.
2) Abstract class**doesn't support multipleinheritance**.	Interface **supportsmultipleinheritance**.
3)Abstractclass**canhavefinal,non-final,staticandnon-staticvariables**.	Interfacehas**onlystaticandfinalvariables**.
4) Abstract class**can provide the implementationofinterface**.	Interface**can'tprovidetheimplementationof abstractclass**.
5)The**abstractkeyword**isusedto declareabstractclass.	The**interface keyword**is used to declare interface.

6)**Example:**
public abstract class Shape{
public abstract void draw()
}
	Example:
	public interface Drawable{
	void draw()
	}

JavaInnerClasses

Javainnerclassornestedclassisaclasswhichisdeclaredinsidetheclassorinterface.

Weuseinnerclassesto logicallygroupclassesandinterfaces inoneplacesothat it canbe more readable and maintainable.

SyntaxofInnerclass

```
1. classJava_Outer_class{
2.  //code
3.  classJava_Inner_class{
4.   //code
5.  }}
```

Advantageofjavainnerclasses

Therearebasicallythreeadvantagesofinnerclassesinjava.Theyareasfollows:

1) Nestedclassesrepresent aspecialtypeofrelationshipthat is**itcanaccessallthemembers (data members and methods) of outer class** includingprivate.

2) Nestedclassesareused**todevelopmorereadableandmaintainablecode**becauseit logically group classes and interfaces in one placeonly.

3) **CodeOptimization**:Itrequireslesscodetowrite.

Differencebetweennestedclassand innerclassinJava

Innerclassisapartofnestedclass.Non-staticnestedclassesareknownasinnerclasses.

TypesofNestedclasses

Therearetwotypesofnestedclassesnon-staticandstaticnestedclasses.The non-staticnested classes are also known as inner classes.

- Non-staticnestedclass(innerclass)
    1. Memberinnerclass
    2. Anonymousinnerclass
    3. Localinnerclass
- Staticnestedclass

JavaPackage

A **java package** is a group ofsimilar types ofclasses,interfaces and sub-packages. Package

injavacanbecategorized intwoform,built-inpackageanduser-definedpackage. There are

many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql

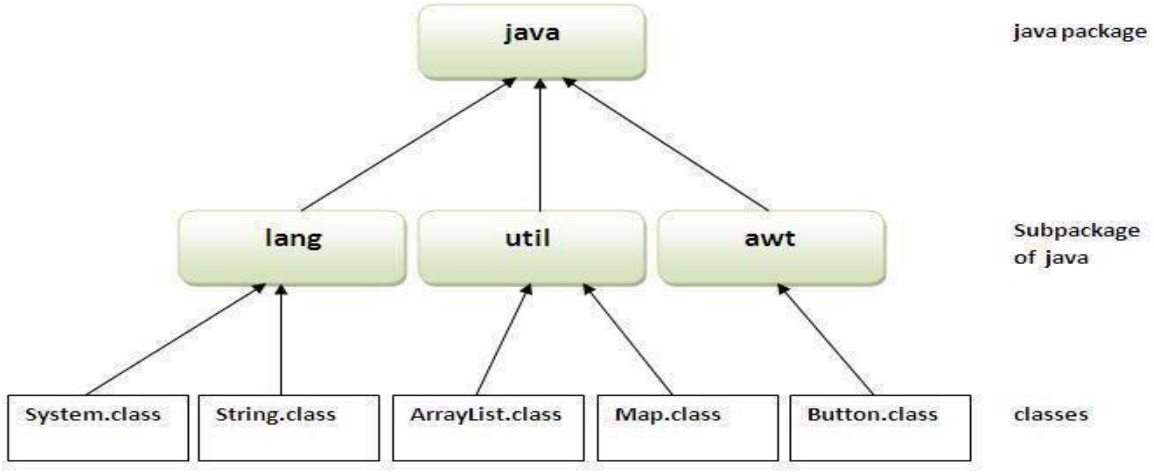
etc.**Advantage of JavaPackage**

1) Javapackage isusedto categorizetheclassesand interfacesso thattheycanbeeasily maintained.

2) Javapackageprovidesaccessprotection.

3) Javapackageremovesnamingcollision.

```
packagemypack;
publicclassSimple{
 public static void main(String args[]){
   System.out.println("Welcome to package");
  }}
```



Howtocompilejavapackage

IfyouarenotusinganyIDE, youneedtofollowthe**syntax**givenbelow: javac -d

directory javafilename

Howtorunjavapackageprogram

ToCompile:javac-d.Simple.java

ToRun:javamypack.Simple

Usingfullyqualifiedname

Exampleofpackagebyimportfullyqualifiedname

```
//savebyA.java
package pack;
publicclassA{
  publicvoidmsg(){System.out.println("Hello");}}
//save by B.java
packagemypack;
class B{
  publicstaticvoid main(Stringargs[]){
   pack.Aobj=newpack.A();//usingfullyqualifiedname obj.msg();
  }
}
```

Output:Hello

UNIT-3

ExceptionHandling

The**exceptionhandlinginjava**isoneofthepowerful*mechanismto handle the runtime errors* so that normal flow of the application can bemaintained.

Whatisexception

Injava, exception isanevent that disruptsthe normalflowofthe program. It isanobject which is thrown at runtime.

AdvantageofExceptionHandling

The core advantage of exception handling is **to maintain the normal flow of the application**. Exception normally disrupts the normal flow of the application that is why we use exception handling.

TypesofException

Therearemainlytwotypesofexceptions:checked anduncheckedwhereerrorisconsideredas unchecked exception. The sun microsystem says there are three types of exceptions:

1. CheckedException
2. UncheckedException
3. Error

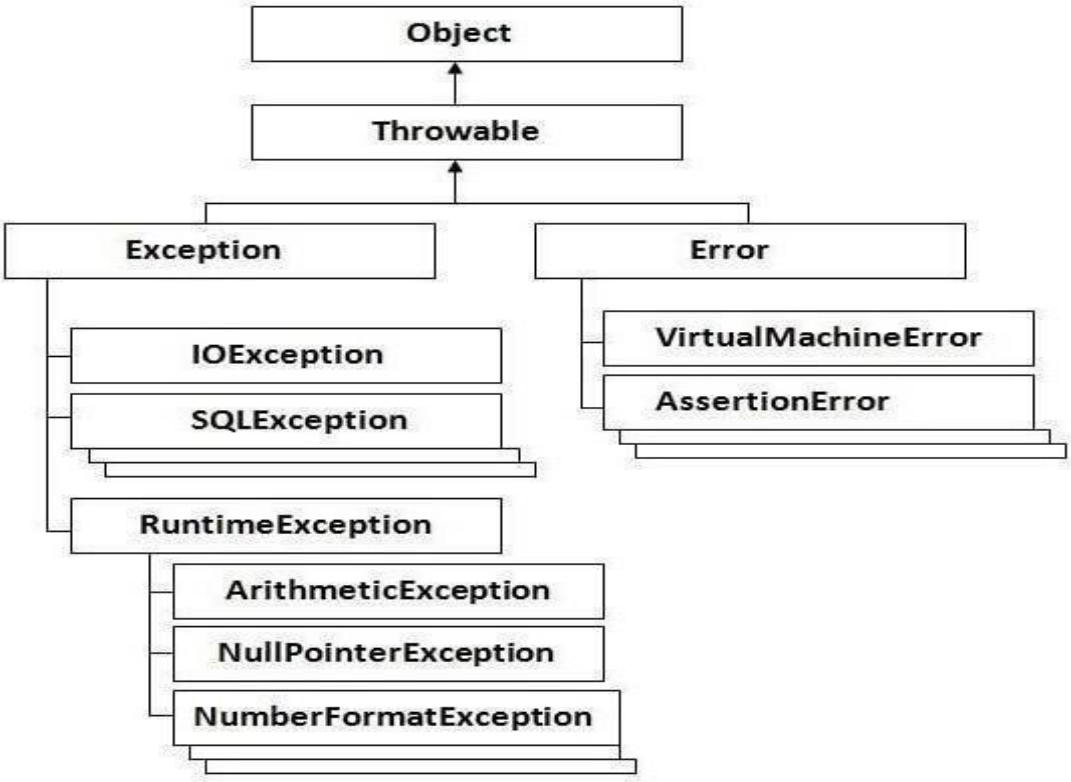
Differencebetweenchecked**anduncheckedexceptions**

1) Checked Exception: The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g.IOException, SQLException etc. Checked exceptions are checked atcompile-time.

2) Unchecked Exception: The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked atruntime.

3) Error:Errorisirrecoverablee.g.OutOfMemoryError,VirtualMachineError,AssertionErroretc.

Hierarchyof JavaExceptionclasses



CheckedandUnCheckedExceptions

Checked Exceptions	Unchecked Exceptions
• Exception which are checked at Compile time called Checked Exception • If a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword	• Exceptions whose handling is NOT verified during Compile time. • These exceptions are handled at run-time i.e., by JVM after they occurred by using the try and catch block
• Examples: ○ IOException ○ SQLException ○ DataAccessException ○ ClassNotFoundException ○ InvocationTargetException ○ MalformedURLException	• Examples ○ NullPointerException ○ ArrayIndexOutOfBound ○ IllegalArgumentException ○ IllegalStateException

Javatryblock

Javatryblock isusedtoenclosethecodethatmightthrowanexception. Itmust beusedwithin the method.

Javatryblockmustbefollowedbyeithercatchorfinallyblock.

Syntaxofjavatry-catch

```
1. try{
2. //codethatmaythrowexception
3. }catch(Exception_class_Name
```

ref){}Syntaxoftry-finallyblock

```
1. try{
2. //codethatmaythrowexception
3. }finally{}
```

Javacatchblock

Javacatchblock isusedtohandletheException. Itmust beusedafterthetryblockonly. You can

use multiple catch block with a single try.

Problemwithoutexceptionhandling

Let'strytounderstandthe problemifwedon'tusetry-catchblock.

```
publicclassTesttrycatch1{
  public static void main(String args[]){
    int data=50/0;//may throw exception
    System.out.println("restofthecode...");
}}
```

Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:/byzero
```

Asdisplayedintheaboveexample, restofthecodeisnot executed(insuchcase, restofthe code... statement is not printed).

Therecanbe100 linesofcodeafterexception. Soallthecodeafterexceptionwillnotbe executed.

Solutionbyexceptionhandling

Let'sseethesolutionofabove problembyjavatry-catchblock.

```
publicclassTesttrycatch2{
```

```
 publicstaticvoidmain(Stringargs[]){
  try{
    intdata=50/0;
  }catch(ArithmeticExceptione){System.out.println(e);}
  System.out.println("rest of the code...");
}}
```

1. Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:/byzero rest
of the code...
```

Now,asdisplayed intheaboveexample, restofthecodeisexecuted i.e. restofthecode... statement is printed.

JavaMulticatchblock

Ifyouhavetoperformdifferent tasksattheoccurrenceofdifferent Exceptions,use java multi catch block.

Let'sseeasimpleexampleofjavamulti-catchblock.

```
1.  publicclassTestMultipleCatchBlock{
2.    publicstaticvoidmain(Stringargs[]){
3.     try{
4.      inta[]=newint[5];
5.      a[5]=30/0;
6.     }
7.   catch(ArithmeticExceptione){System.out.println("task1iscompleted");}
8.  catch(ArrayIndexOutOfBoundsExceptione){System.out.println("task2completed");
9.}
10.    catch(Exceptione){System.out.println("commontaskcompleted");
11.}
12.    System.out.println("restofthecode...");
13. }}
```

```
Output:task1completed
      rest of the code...
```

Javanestedtryexample

Let'sseeasimpleexampleofjavanestedtryblock.

```
classExcep6{
 publicstaticvoid main(Stringargs[]){
  try{t
    ry{
     System.out.println("goingtodivide");
     intb=39/0;
    }catch(ArithmeticException e){System.out.println(e);}

    try{
```

```
    int a[]=new int[5];
    a[5]=4;
    }catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}
    System.out.println("other statement);
  }catch(Exception e){System.out.println("handeled");}
  System.out.println("normal flow..");
 }
1. }
```

Java finally block

Java finally block is a block that is used *to execute important code* such as closing connection, stream etc.

Java finally block is always executed whether exception is handled or not. Java

finally block follows try or catch block.

Usage of Java finally

Case1

Let's see the java finally example where **exception doesn't occur**.

```
class TestFinallyBlock{
  public static void main(String args[]){
  try{
   int data=25/5;
   System.out.println(data);
  }
  catch(NullPointerException e){System.out.println(e);}
  finally{System.out.println("finally block is always executed");}
  System.out.println("rest of the code...");
  }
}
```

```
Output:5
      finally block is always executed
      rest of the code...
```

Java throw keyword

The Java throw keyword is used to explicitly throw an exception.

We can throw either checked or unchecked exception in java by throw keyword. The throw keyword is mainly used to throw custom exception. We will see custom exceptions later.

The syntax of java throw keyword is given below.

```
1. throw exception;
```

Javathrowkeywordexample

Inthisexample, wehavecreatedthevalidatemethodthattakesinteger valueasaparameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message welcome to vote.

```
1. publicclassTestThrow1{
        staticvoidvalidate(intage){
        if(age<18)
        thrownewArithmeticException("notvalid");
        else
        System.out.println("welcometovote");
        }
        publicstaticvoidmain(Stringargs[]){
        validate(13);
        System.out.println("restofthecode...");
        }}
```

Output:

```
Exceptioninthreadmainjava.lang.ArithmeticException:notvalid
```

Javathrowskeyword

The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception so it is better for the programmer to provide the exception handling code so that normal flow can be maintained.

Exception Handling is mainly used to handle the checked exceptions. If there occurs any unchecked exception such as NullPointerException, it is programmers fault that he is not performing check up before the code being used.

Syntaxofjava throws

```
1. return_typemethod_name()throwsexception_class_name{
2. //method
code3. }
4.
```

Javathrowsexample

Let'sseetheexampleofjavathrowsclausewhichdescribesthat checkedexceptionscanbe propagated by throws keyword.

```
import java.io.IOException;
class Testthrows1{
void m()throwsIOException{
   thrownewIOException("deviceerror");//checkedexception
```

```
}
void n()throws IOException{
  m();
}
void p(){
 try{n();
 }catch(Exceptione){System.out.println("exceptionhandled");}
}
publicstaticvoidmain(Stringargs[]){
 Testthrows1 obj=new Testthrows1();
 obj.p();
 System.out.println("normalflow...");}} Output:
```

```
exceptionhandled
normal flow...
```

JavaCustomException

If you are creating your own Exception that is known as custom exception or user-defined exception.Javacustomexceptionsareusedto customizetheexceptionaccordingtouserneed.

Bythehelpofcustomexception,youcanhave yourownexceptionand message. Let's see

a simple example of java custom exception.

```
classInvalidAgeExceptionextendsException{
 InvalidAgeException(String s){
  super(s);
 }}
classTestCustomException1{
  staticvoidvalidate(intage)throwsInvalidAgeException{
    if(age<18)
     thrownewInvalidAgeException("notvalid");
    else
     System.out.println("welcometovote");
  }
  publicstaticvoid main(Stringargs[]){
     try{
     validate(13);
     }catch(Exceptionm){System.out.println("Exceptionoccured:"+m);}

     System.out.println("restofthecode...");
  }}
```

Output:Exceptionoccured:InvalidAgeException:not validrestofthecode...

Multithreading

Multithreadinginjavaisaprocessofexecuting multiplethreadssimultaneously.

Threadisbasicallyalightweight sub-process,asmallestunitofprocessing.Multiprocessingand multithreading, both are used to achieve multitasking.

But we use multithreading than multiprocessing because threads share a common memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

JavaMultithreadingismostlyusedingames, animationetc.

AdvantagesofJavaMultithreading

1) It**doesn'tblocktheuser**becausethreadsare independent and youcanperformmultiple operations at sametime.

2) You**canperformmanyoperationstogethersoit savestime**.

3) Threadsare**independent**soitdoesn'taffectotherthreads ifexceptionoccurinasinglethread.

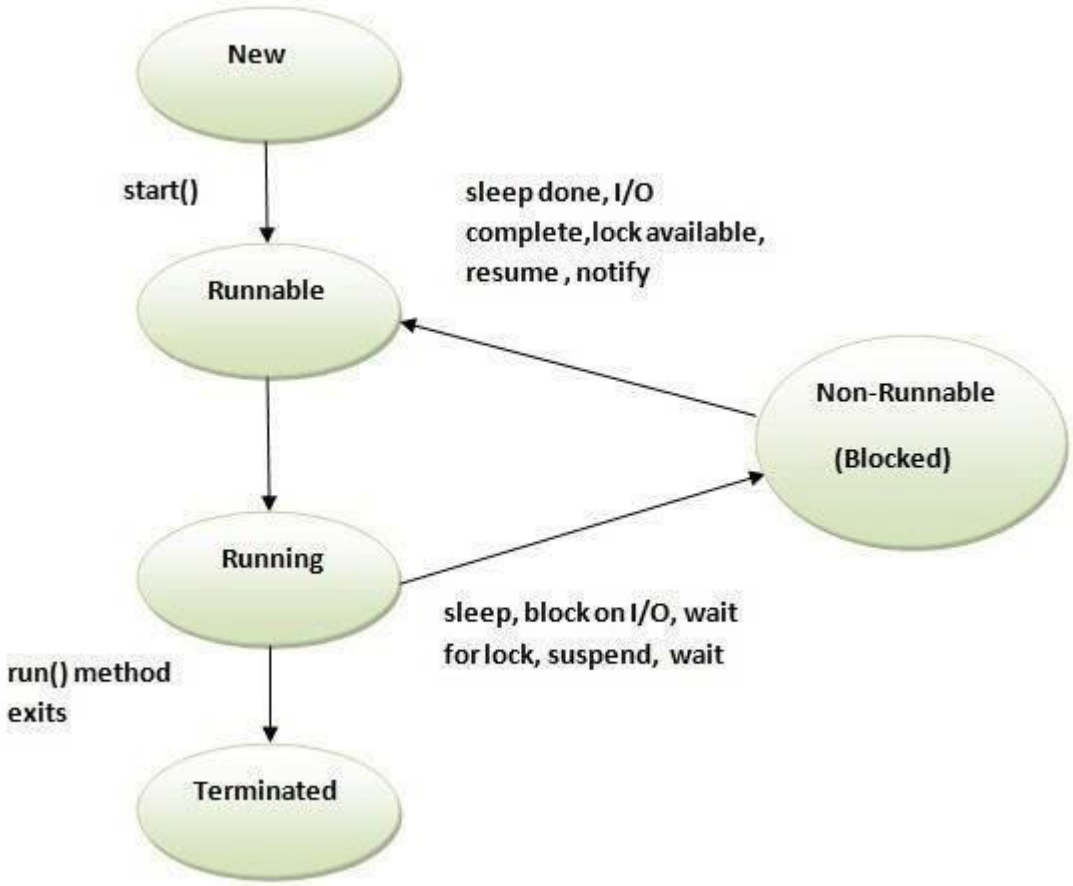
LifecycleofaThread(ThreadStates)

Athreadcanbe inoneofthefivestates.Accordingto sun,there isonly4statesin**threadlife cycle in java** new, runnable, non-runnable and terminated. There is no running state.

Butforbetterunderstandingthethreads,weareexplainingitinthe5states.

ThelifecycleofthethreadinjavaiscontrolledbyJVM. Thejavathreadstatesareasfollows:

1. New
2. Runnable
3. Running
4. Non-Runnable(Blocked)
5. Terminated



Howtocreatethread

Therearetwowaystocreate athread:

1. ByextendingThreadclass
2. ByimplementingRunnableinterface.

Threadclass:

Threadclassprovideconstructorsandmethodsto createandperformoperationsona thread.Thread class extends Object class and implements Runnable interface.

CommonlyusedConstructorsofThreadclass:

- Thread()
- Thread(Stringname)
- Thread(Runnable r)
- Thread(Runnabler,Stringname)

CommonlyusedmethodsofThreadclass:

1. **publicvoidrun():**isusedtoperformaction forathread.
2. **publicvoidstart():**startstheexecutionofthethread.JVMcallstherun()methodonthethread.
3. **publicvoidsleep(longmiliseconds):**Causesthecurrentlyexecutingthreadtosleep(temporarily cease execution) for the specified number ofmilliseconds.
4. **publicvoidjoin():** waitsforathreadtodie.
5. **publicvoidjoin(longmiliseconds):**waitsforathreadtodieforthespecifiedmiliseconds.
6. **publicintgetPriority():**returnsthepriorityofthethread.
7. **publicintsetPriority(intpriority):**changesthepriorityofthethread.
8. **publicString getName():**returnsthenameofthethread.
9. **publicvoidsetName(Stringname):**changesthenameofthethread.
10. **publicThreadcurrentThread():**returnsthereferenceofcurrentlyexecutingthread.
11. **publicintgetId():**returnstheidofthethread.
12. **publicThread.StategetState():**returnsthestateofthethread.
13. **publicboolean isAlive():**testsifthethreadisalive.
14. **publicvoidyield():**causesthecurrentlyexecutingthreadobjecttotemporarilypauseandallow other threads toexecute.
15. **publicvoidsuspend():**isusedtosuspendthethread(depricated).
16. **publicvoidresume():** isusedtoresumethesuspendedthread(depricated).
17. **publicvoidstop():**isusedtostopthethread(depricated).
18. **publicbooleanisDaemon():**testsifthethreadisa daemonthread.
19. **publicvoidsetDaemon(booleanb):**marksthethreadasdaemonoruserthread.
20. **publicvoidinterrupt():**interruptsthethread.
21. **publicbooleanisInterrupted():**testsifthethreadhasbeeninterrupted.
22. **publicstaticbooleaninterrupted():**testsifthecurrentthreadhasbeeninterrupted.

Runnableinterface:

TheRunnableinterfaceshouldbeimplementedbyanyclasswhoseinstancesareintendedtobe executed bya thread. Runnable interface have only one method named run().

1. **publicvoidrun():**isusedtoperformaction forathread.

Startingathread:

start()methodofThreadclassisusedtostartanewlycreatedthread.Itperforms following tasks:

- Anewthreadstarts(withnewcallstack).
- ThethreadmovesfromNewstatetotheRunnable state.
- Whenthethreadgetsachancetoexecute,itstargetrun()methodwillrun.

JavaThreadExamplebyextendingThreadclass

```
classMulti extends Thread{
public void run(){
System.out.println("thread isrunning...");
}
publicstaticvoidmain(Stringargs[]){
Multi t1=new Multi();
t1.start();
 }}
```

Output:threadisrunning...

JavaThreadExamplebyimplementingRunnableinterface

```
class Multi3 implements Runnable{
public void run(){
System.out.println("thread is running...");
}
publicstaticvoidmain(Stringargs[]){
Multi3 m1=new Multi3();
Threadt1=newThread(m1);
t1.start();
 }}
```

Output:threadisrunning...

PriorityofaThread(ThreadPriority):

Eachthreadhaveapriority.Prioritiesarerepresentedbya numberbetween1and10.Inmost cases,threadschedularschedulesthethreadsaccordingtotheirpriority(knownaspreemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses.

3constantsdefinedinThreadclass:

1. publicstaticintMIN_PRIORITY
2. publicstaticintNORM_PRIORITY
3. publicstaticintMAX_PRIORITY

Defaultpriorityofathread is5(NORM_PRIORITY).ThevalueofMIN_PRIORITYis1and the value of MAX_PRIORITY is 10.

ExampleofpriorityofaThread: class

```
TestMultiPriority1 extends
Thread{public void run(){
  System.out.println("running thread name is:"+Thread.currentThread().getName());
  System.out.println("running thread priority is:"+Thread.currentThread().getPriority());
  }
 publicstaticvoid main(Stringargs[]){
```

```
TestMultiPriority1m1=newTestMultiPriority1();
TestMultiPriority1m2=newTestMultiPriority1();
m1.setPriority(Thread.MIN_PRIORITY);
m2.setPriority(Thread.MAX_PRIORITY);
m1.start();
m2.start();
}}
```

Output:runningthreadname is:Thread-0
running thread priority is:10
runningthreadnameis:Thread-1
running thread priority is:1

Javasynchronizedmethod

Ifyoudeclareanymethodassynchronized, it isknownassynchronized method. Synchronized

method is used to lock an object for any shared resource.

Whenathreadinvokesasynchronized method,it automaticallyacquiresthe lockforthatobject and releases it when the thread completes its task.

Exampleofinterthreadcommunicationinjava

Let'sseethesimpleexampleofinterthread communication.

```
classCustomer{
intamount=10000;
synchronizedvoidwithdraw(intamount){
System.out.println("going to withdraw...");
if(this.amount<amount){
System.out.println("Lessbalance;waitingfordeposit...");
try{wait();}catch(Exception e){}
}
this.amount-=amount;
System.out.println("withdrawcompleted...");
}
synchronized void deposit(int amount){
System.out.println("going to deposit...");
this.amount+=amount;
System.out.println("depositcompleted...");
notify();
}
}
classTest{
publicstaticvoidmain(Stringargs[]){
final Customer c=newCustomer();
new Thread(){
publicvoidrun(){c.withdraw(15000);}
}.start();
newThread(){
```

```
publicvoidrun(){c.deposit(10000);}
}
start();
}}
```

```
Output:goingtowithdraw...
       Lessbalance;waitingfordeposit...
       going to deposit...
       depositcompleted...
       withdrawcompleted
```

ThreadGroupinJava

Javaprovidesaconvenient waytogroupmultiplethreads inasingleobject.Insuchway, wecan suspend, resume or interrupt group ofthreads by a single method call.

Note:Nowsuspend(),resume()andstop()methodsaredeprecated.

Javathreadgroupis implementedby*java.lang.ThreadGroup*

class.ConstructorsofThreadGroup class

TherearetwoconstructorsofThreadGroupclass.

ThreadGroup(String name)
ThreadGroup(ThreadGroup parent,String name)

Let'sseeacodetogroupmultiplethreads.

```
1. ThreadGrouptg1=newThreadGroup("GroupA");
2. Threadt1=newThread(tg1,newMyRunnable(),"one");
3. Threadt2=newThread(tg1,newMyRunnable(),"two");
4. Threadt3=newThread(tg1,newMyRunnable(),"three");
```

Nowall3threadsbelongtoonegroup.Here,tg1isthethreadgroupname, MyRunnable isthe class that implements Runnable interface and "one", "two" and "three" are thethread names.

Nowwecaninterruptallthreadsbyasingleline ofcodeonly.

```
1. Thread.currentThread().getThreadGroup().interrupt();
```

Producer-ConsumersolutionusingthreadsinJava

Incomputing,theproducer-consumerproblem(alsoknownasthebounded-bufferproblem)isa classic example of a multi-process synchronization problem. The problem describes two processes, the producer and the consumer, which share a common, fixed-size buffer used as a queue.

- Theproducer'sjobistogeneratedata,putitintothebuffer,andstartagain.
- At the same time, the consumer is consuming the data (i.e. removing itfrom the buffer), one piece at a time.

Problem

Tomakesurethattheproducerwon'ttry toadddataintothebufferifit'sfull andthattheconsumer won't try to remove data from an empty buffer.

Solution

The producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removesanitemfromthebuffer,itnotifiestheproducer,whostartstofillthebufferagain.Inthe same way, the consumer can go to sleep if it finds the buffer to be empty. The next time the producer putsdataintothebuffer,itwakesupthesleepingconsumer. An inadequate solution could resultin a deadlock where both processes are waiting to be awakened.

ImplementationofProducerConsumerClass

- A**LinkedListlist**–tostorelistofjobsinqueue.
- **AVariableCapacity**–tocheckforifthelistisfullornot
- Amechanism tocontroltheinsertionandextraction from thislistsothatwedonotinsertintolistifit is full or remove from it if it is empty.

```
//Javaprogramtoimplementsolutionofproducer
// consumer problem.

importjava.util.LinkedList;

publicclassThreadexample{
publicstaticvoidmain(String[]args)
        throws InterruptedException
{
        // Object ofa classthathasbothproduce()
        //andconsume()methods
        final PC pc = new PC();

        //Createproducerthread
        Threadt1=newThread(newRunnable(){
                @Override
                publicvoidrun()
                {
                        try{
                                pc.produce();
                        }
                        catch(InterruptedExceptione){
                                e.printStackTrace();
                        }
                }
        });
```

```
//Createconsumer thread
    Threadt2=newThread(newRunnable(){
            @Override
            publicvoidrun()
            {
                try{
                    pc.consume();
                }
                catch(InterruptedExceptione){
                    e.printStackTrace();
                }
            }
        });

        //Startboththreads t1.start();
        t2.start();

        //t1finishesbeforet2
        t1.join();
        t2.join();
}

//Thisclasshasalist,producer(addsitemsto list
//andconsumer(removesitems).
public static class PC {
        //Createalistsharedbyproducer and consumer
        //Sizeoflistis2.
        LinkedList<Integer>list=newLinkedList<>();
        int capacity = 2;

        // Functioncalled byproducer thread
        publicvoid produce()throws InterruptedException
        {
            int value=0;
            while(true) {
                synchronized(this)
                {
                    //producerthreadwaitswhilelist isfull
                    while (list.size() == capacity)
                        wait();
                    System.out.println("Producerproduced-"+value);

                    //toinsertthejobsinthelist
                    list.add(value++);
                    //notifiestheconsumerthreadthat nowit canstartconsuming notify();
                    //makestheworkingofprogrameasierto understand Thread.sleep(1000);
                }
            }
        }
```

```
        //Functioncalledbyconsumerthread
        publicvoidconsume() throwsInterruptedException
        {
                while(true) {
                        synchronized(this)
                        {
                                //consumerthreadwaitswhilelist
                                //is empty
                                while(list.size()==0)
                                        wait();

                                //toretrievethefirstjobinthe list int
                                val = list.removeFirst();

                                System.out.println("Consumerconsumed-"
                                                                +val);

                                //Wakeupproducerthread
                                notify();

                                // and sleep
                                Thread.sleep(1000);
                        }
                }
        }
}
}
```

UNIT-4

CollectionFrameworkinJava

Collections in javais a framework that provides an architecture to store and manipulate the groupofobjects.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque etc.) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet etc).

Whatisframeworkinjava

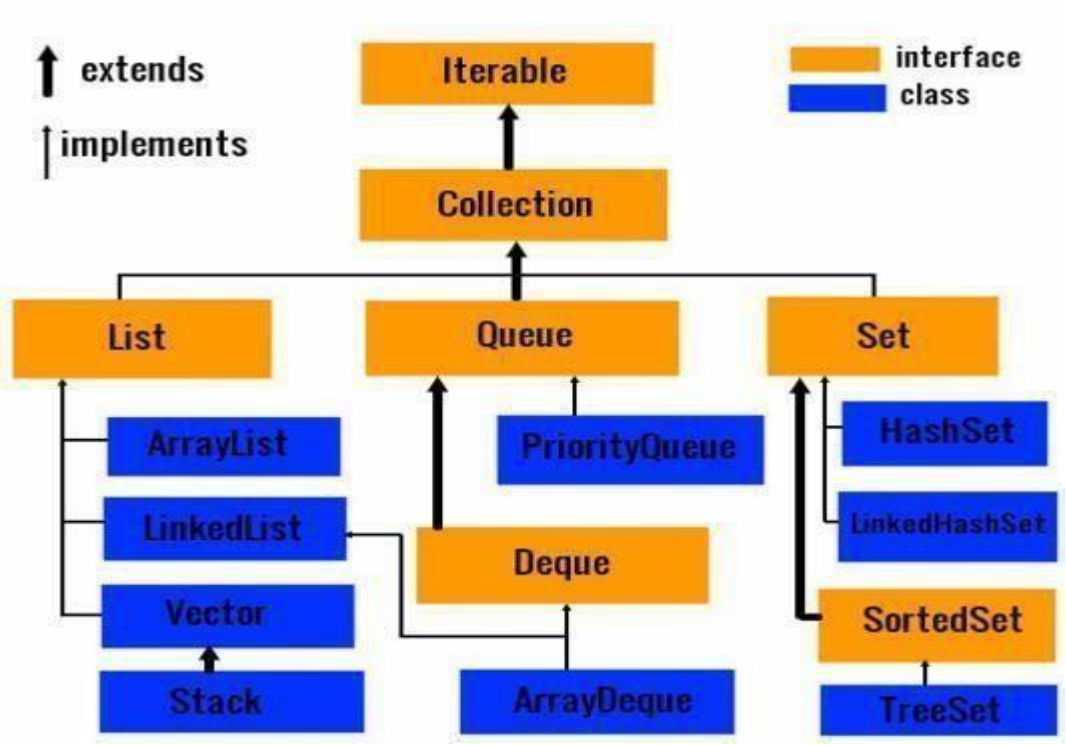
- providesreadymadearchitecture.
- representssetofclassesandinterface.
- isoptional.

WhatisCollectionframework

Collection framework represents a unified architecture for storing and manipulating group of objects. It has:

1. Interfacesanditsimplementationsi.e.classes
2. Algorithm

HierarchyofCollectionFramework



JavaArrayListclass

JavaArrayList classusesadynamicarrayforstoringtheelements.ItinheritsAbstractList class and implements List interface.

TheimportantpointsaboutJavaArrayListclassare:

- JavaArrayListclasscancontainduplicateelements.
- JavaArrayListclassmaintainsinsertionorder.
- JavaArrayListclassisnonsynchronized.
- JavaArrayListallowsrandomaccessbecausearrayworksattheindexbasis.
- InJavaArrayList class,manipulationisslowbecausea lotofshifting needstobeoccurred if any element is removed from the arraylist.

ArrayListclassdeclaration

Let'sseethedeclarationforjava.util.ArrayListclass.

ConstructorsofJavaArrayList

Constructor	Description
ArrayList()	It is usedtobuildanemptyarraylist.
ArrayList(Collection c)	Itisusedtobuildanarraylist that isinitializedwiththe elements of the collection c.
ArrayList(int capacity)	Itisusedtobuildanarraylist that hasthespecified initial capacity.

JavaArrayListExample

```
import java.util.*;
classTestCollection1{
 publicstaticvoid main(Stringargs[]){
  ArrayList<String> list=new ArrayList<String>();//Creating arraylist
  list.add("Ravi");//Adding object in arraylist
  list.add("Vijay");
  list.add("Ravi");
  list.add("Ajay");
  //Traversing list through Iterator
  Iterator itr=list.iterator();
  while(itr.hasNext()){
  System.out.println(itr.next());}}}
```

```
Ravi
Vijay
Ravi
Ajay
```

vector

ArrayListandVectorbothimplementsListinterfaceandmaintainsinsertionorder.

ButtherearemanydifferencesbetweenArrayListandVectorclassesthataregivenbelow.

ArrayList	Vector
1)ArrayListisnotsynchronized.	Vectoris**synchronized**.
2)ArrayList**increments 50%** of current array size if number of element exceeds from its capacity.	Vector **increments 100%** means doubles the array size if total number of element exceeds than its capacity.
3)ArrayListis**notalegacy**class, itis introduced in JDK1.2.	Vectorisa**legacy** class.
4)ArrayListis**fast**becauseitis non-synchronized.	Vector is **slow** because it is synchronized i.e. in multithreading environment, it will hold the other threads in runnable or non-runnable stateuntilcurrent thread releases the lock of object.
5)ArrayListusesIteratorinterface to traverse the elements.	VectorusesEnumerationinterfacetotraversethe elements. But it can use Iterator also.

ExampleofJavaVector

Let'sseeasimpleexampleofjavaVectorclassthat usesEnumerationinterface.

```
1. importjava.util.*;
2. classTestVector1{
3.  publicstaticvoidmain(Stringargs[]){
4.   Vector<String>v=newVector<String>();//creatingvector
5.   v.add("umesh");//methodofCollection
6.   v.addElement("irfan");//methodofVector
7.   v.addElement("kumar");
8.   //traversingelementsusingEnumeration
```

```
9.    Enumeratione=v.elements();
10.   while(e.hasMoreElements()){
11.    System.out.println(e.nextElement());
12. }}}
```

Output:

```
umesh
irfan
kumar
```

JavaHashtableclass

JavaHashtableclassimplementsahashtable,whichmapskeystovalues.ItinheritsDictionary class and implements the Map interface.

TheimportantpointsaboutJavaHashtableclassare:

- A Hashtable is an array of list. Each list is known as a bucket. The position of bucket is identified by calling the hashcode() method. A Hashtable contains values based on thekey.
- Itcontainsonlyuniqueelements.
- Itmayhavenothaveanynullkeyorvalue.
- Itissynchronized.

Hashtableclassdeclaration

Let'sseethedeclarationforjava.util.Hashtable class.

```
1. publicclassHashtable<K,V>extendsDictionary<K,V>implementsMap<K,V>,Cloneable,Ser
   ializable
```

HashtableclassParameters

Let'sseetheParametersforjava.util.Hashtableclass.

- **K**:Itisthetypeofkeysmaintainedbythismap.
- **V**:Itisthetypeofmappedvalues.

ConstructorsofJavaHashtableclass

Constructor	Description
Hashtable()	Itisthedefault constructorofhashtableit instantiatesthe Hashtableclass.
Hashtable(intsize)	Itisusedtoaccept anintegerparameterandcreatesahashtable that has an initial size specified by integer valuesize.
Hashtable(int size, float fillRatio)	Itisusedtocreateahashtablethat hasaninitialsizespecified by size and a fill ratio specified by fillRatio.

JavaHashtableExample

```
importjava.util.*;
classTestCollection16{
 public static void main(String args[]){Hashtable<Integer,String>
  hm=new Hashtable<Integer,String>(); hm.put(100,"Amit");
  hm.put(102,"Ravi");
  hm.put(101,"Vijay");
  hm.put(103,"Rahul");
  for(Map.Entry m:hm.entrySet()){
   System.out.println(m.getKey()+""+m.getValue());
  }}}
```

Output:

```
103 Rahul
102Ravi
101 Vijay
100 Amit
```

Stack

StackisasubclassofVectorthat implementsastandardlast-in,first-outstack.

Stackonlydefinesthedefault constructor,whichcreatesanemptystack. Stackincludesallthe methods defined by Vector, and adds several of its own.

```
Stack()
```

Example

Thefollowing programillustratesseveralofthemethodssupportedbythiscollection−

```
importjava.util.*;
publicclassStackDemo{
   static void showpush(Stack st, int a) {
      st.push(new Integer(a));
      System.out.println("push("+a+")");
      System.out.println("stack: "+ st);}
   static void showpop(Stack st) {
      System.out.print("pop ->");
      Integer a = (Integer) st.pop();
      System.out.println(a);
      System.out.println("stack:"+st);}
   publicstaticvoidmain(Stringargs[]){
      Stack st = new Stack();
      System.out.println("stack: "+ st);
      showpush(st,42);
      showpush(st,66);
      showpush(st,99);
      showpop(st);
      showpop(st);
      showpop(st);try{
         showpop(st);
      } catch (EmptyStackException e) {
         System.out.println("emptystack");
```

```
}}}
```

Thiswillproducethefollowingresult−

Output

```
stack: [ ]
push(42)
stack: [42]
push(66)
stack:[42,66]
push(99)
stack:[42,66,99]
pop->99
stack:[42,66]
pop->66
stack:[42]
pop->42
stack: [ ]
pop->emptystack
```

Enumeration

TheEnumerationInterface

TheEnumerationinterfacedefinesthe methodsbywhichyoucanenumerate(obtainoneat a time) the elements in a collection of objects.
ThemethodsdeclaredbyEnumerationaresummarizedinthefollowingtable−

Sr.No.	Method&Description
1	**booleanhasMoreElements()** Whenimplemented,it must returntruewhiletherearestillmoreelementstoextract,and false when all the elements have been enumerated.
2	**ObjectnextElement()** ThisreturnsthenextobjectintheenumerationasagenericObjectreference.

Example

Followingisanexampleshowingusageof Enumeration.

```
importjava.util.Vector;
importjava.util.Enumeration;
publicclassEnumerationTester{
  publicstaticvoidmain(Stringargs[]){
    Enumeration days;
    VectordayNames=newVector();
    dayNames.add("Sunday");
    dayNames.add("Monday");
    dayNames.add("Tuesday");
    dayNames.add("Wednesday");
    dayNames.add("Thursday");
    dayNames.add("Friday");
    dayNames.add("Saturday");
    days= dayNames.elements();
    while (days.hasMoreElements()){
      System.out.println(days.nextElement());
    }}}
```

Thiswillproducethefollowingresult−

Output

```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

Iterator

It is a **universal** iterator as we can apply it to any Collection object. By using Iterator, we can performbothreadandremoveoperations. Itis improvedversionofEnumerationwithadditional functionality of remove-ability of a element.
Iterator must be used whenever we want to enumerate elements in all Collection framework implemented interfaces like Set, List, Queue, Deque and also in all implemented classes of Map interface. Iterator is the **only** cursor available for entire collection framework.
Iteratorobjectcanbecreatedbycalling*iterator()*methodpresentinCollectioninterface.

```
//Here"c"isanyCollectionobject.itr is of

//typeIteratorinterfaceandrefersto"c" Iterator

itr = c.iterator();
```

Iteratorinterfacedefines**three**methods:

```
//Returnstrueiftheiterationhasmore elements

publicbooleanhasNext();

//Returnsthenextelementintheiteration
//ItthrowsNoSuchElementExceptionifnomore
//elementpresent
publicObjectnext();

//Removethenextelementintheiteration
//Thismethodcanbecalledonlyoncepercall
//tonext()

publicvoidremove();
```

***remove()*methodcanthrowtwoexceptions**

- *UnsupportedOperationException:* Iftheremoveoperation isnotsupportedbythis iterator
- *IllegalStateException:*Ifthenext methodhasnotyet beencalled, ortheremove method has already been called after the last call to the next method

LimitationsofIterator:

- Onlyforwarddirectioniteratingispossible.
- ReplacementandadditionofnewelementisnotsupportedbyIterator.

StringTokenizerinJava

The**java.util.StringTokenizer**classallows you to break astring intotokens. It issimple wayto break string.

Itdoesn'tprovidethefacilitytodifferentiatenumbers,quotedstrings,identifiersetc.

ConstructorsofStringTokenizerclass

Thereare3constructorsdefinedintheStringTokenizerclass.

Constructor	Description
StringTokenizer(Stringstr)	createsStringTokenizerwithspecifiedstring.
StringTokenizer(String str, String delim)	creates StringTokenizer with specified string and delimeter.
StringTokenizer(String str, String delim, boolean returnValue)	createsStringTokenizerwithspecifiedstring,delimeter andreturnValue. Ifreturnvalueistrue, delimiter charactersareconsidered tobetokens.Ifitisfalse, delimitercharactersservetoseparatetokens.

MethodsofStringTokenizerclass

The6usefulmethodsofStringTokenizerclassareasfollows:

Publicmethod	Description
booleanhasMoreTokens()	checksifthereismoretokensavailable.
StringnextToken()	returnsthenexttokenfromtheStringTokenizerobject.
StringnextToken(String delim)	returnsthe nexttokenbasedonthe delimeter.
booleanhasMoreElements()	sameashasMoreTokens()method.
ObjectnextElement()	sameasnextToken()butitsreturntypeisObject.
intcountTokens()	returnsthetotalnumberoftokens.

SimpleexampleofStringTokenizerclass

Let'sseethesimpleexampleofStringTokenizer classthattokenizesastring"myname iskhan" on the basis of whitespace.

```
importjava.util.StringTokenizer;
public class Simple{
 publicstaticvoid main(Stringargs[]){
StringTokenizerst=newStringTokenizer("mynameiskhan","");
    while(st.hasMoreTokens()) {
      System.out.println(st.nextToken());
    }}}
```

Output:my
name
is
khan

ExampleofnextToken(Stringdelim)methodofStringTokenizer class

```
importjava.util.*;
publicclassTest{
  publicstaticvoidmain(String[]args) {
    StringTokenizerst=newStringTokenizer("my,name,is,khan");
    //printing next token
    System.out.println("Nexttokenis:"+st.nextToken(","));
  }    }
```

Output:Nexttokenis:my

java.util.Random

- For using this class to generate random numbers, we have to first create an instance ofthis class and then invoke methods such as nextInt(), nextDouble(), nextLong() etc using that instance.
- We cangenerate random numbers oftypes integers, float, double, long, booleans using this class.
- We can pass arguments to the methods for placing an upper bound on the range of the numbers to be generated. For example, nextInt(6)willgenerate numbers in the range 0 to 5 bothinclusive.

```
//AJavaprogramtodemonstraterandomnumbergeneration
// using java.util.Random;
import java.util.Random;
publicclassgenerateRandom{

   publicstaticvoidmain(Stringargs[])
   {
      //createinstanceofRandomclass
      Random rand = new Random();
      //Generaterandomintegers inrange0to999 int
      rand_int1 =rand.nextInt(1000);
      intrand_int2=rand.nextInt(1000);
```

```
        // Print random integers
        System.out.println("Random Integers: "+rand_int1);
        System.out.println("RandomIntegers:"+rand_int2);

        //GenerateRandomdoubles
        doublerand_dub1=rand.nextDouble();
        doublerand_dub2=rand.nextDouble();

        //Printrandomdoubles
        System.out.println("RandomDoubles:"+rand_dub1);
        System.out.println("RandomDoubles:"+rand_dub2);
    }}
```

Output:

```
RandomIntegers:547
RandomIntegers:126
RandomDoubles:0.8369779739988428
RandomDoubles:0.5497554388209912
```

JavaScannerclass

There arevariouswaysto readinput fromthe keyboard,the java.util.Scanner class isoneofthem. The **Java Scanner** class breaks the input into tokens using a delimiter that is whitespace bydefault. It provides many methods to read and parse various primitive values.

JavaScannerclassiswidelyusedtoparsetextforstringandprimitivetypesusingregular expression.

JavaScannerclassextendsObjectclassand implementsIteratorandCloseableinterfaces.

CommonlyusedmethodsofScannerclass

ThereisalistofcommonlyusedScannerclassmethods:

Method	Description
publicStringnext()	itreturnsthenexttokenfromthe scanner.
publicStringnextLine()	itmovesthescannerpositiontothenext lineandreturnsthevalue as a string.
publicbytenextByte()	itscansthenexttokenasa byte.

publicshortnextShort()	itscansthenexttokenasashortvalue.
publicintnextInt()	itscansthenexttokenasanintvalue.
publiclongnextLong()	itscansthenext tokenasalongvalue.
publicfloatnextFloat()	itscansthenext tokenasafloatvalue.
public double nextDouble()	itscansthenext tokenasadouble value.

JavaScannerExampletogetinputfromconsole

Let'sseethesimpleexampleoftheJavaScanner classwhichreadsthe int,stringanddouble value as aninput:

```
importjava.util.Scanner;
classScannerTest{
public static void main(String args[]){
 Scanner sc=new Scanner(System.in);
 System.out.println("Enter your rollno");
 int rollno=sc.nextInt();
 System.out.println("Enteryourname");
 String name=sc.next();
 System.out.println("Enter your fee");
 double fee=sc.nextDouble();
 System.out.println("Rollno:"+rollno+" name:"+name+" fee:"+fee);
 sc.close();
}}
```

Output:

```
Enteryourrollno
111
Enteryourname
Ratan
Enter
450000
Rollno:111name:Ratanfee:450000
```

JavaCalendarClass

Java Calendar class is an abstract class that provides methods for converting date between a specific instant in time and a set of calendar fields such as MONTH, YEAR, HOUR, etc. It inherits Object class and implements the Comparable interface.

JavaCalendarclassdeclaration

Let'sseethedeclarationofjava.util.Calendarclass.

1. **publicabstractclass**Calendar**extends**Object
2. **implements**Serializable,Cloneable,Comparable<Calendar>

JavaCalendarClassExample

```
importjava.util.Calendar;
publicclassCalendarExample1{
  public static void main(String[] args) {
  Calendar calendar =Calendar.getInstance();
  System.out.println("Thecurrentdateis:"+calendar.getTime());
  calendar.add(Calendar.DATE, -15);
  System.out.println("15daysago:"+calendar.getTime());
  calendar.add(Calendar.MONTH, 4);
  System.out.println("4monthslater:"+calendar.getTime());
  calendar.add(Calendar.YEAR, 2);
  System.out.println("2years later:"+calendar.getTime());
  }}
```

Output:

```
Thecurrent dateis:ThuJan1918:47:02IST2017 15
days ago: Wed Jan 04 18:47:02 IST 2017
4monthslater: ThuMay0418:47:02IST2017
2yearslater:SatMay0418:47:02IST2019
```

Java-FilesandI/O

The java.io package containsnearlyeveryclass you might ever need to performinput and output (I/O) in Java. All these streams representan inputsource and an output destination. The stream in the java.io package supports manydata such as primitives, object, localized characters,etc.

Stream

Astreamcanbe defined asasequenceofdata.TherearetwokindsofStreams −

- **InPutStream**−TheInputStreamisusedtoreaddatafromasource.
- **OutPutStream**−TheOutputStreamisusedforwritingdatatoadestination.

Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionalityrelated to streams and I/O. We will see the most commonly used examples one by one −

ByteStreams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classesrelatedtobytestreamsbutthemostfrequentlyusedclasses are, **FileInputStream** and **FileOutputStream**. Following is an example which makes use of these two classes to copyan input file into an output file−

Example

```
importjava.io.*;
publicclassCopyFile{
  publicstaticvoidmain(Stringargs[])throwsIOException{ FileInputStream in
    = null;
    FileOutputStreamout=null;
    try{
      in= new FileInputStream("input.txt");
      out= new FileOutputStream("output.txt");
      int c;
      while((c=in.read())!=-1){
```

```
        out.write(c);
      }
    }finally{
      if(in!=null){
        in.close();
      }
      if(out!=null){
        out.close();
      }}}}
```

Nowlet'shaveafile**input.txt**withthefollowingcontent −

```
Thisistestforcopy file.
```

As a next step, compile the above programand execute it, whichwillresult increating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following−

```
$javacCopyFile.java
$javaCopyFile
```

CharacterStreams

Java **Byte** streamsareusedtoperforminputandoutputof8-bitbytes,whereas Java **Character** streams are used to perform input and output for 16-bit unicode. Though there aremanyclassesrelatedtocharacterstreamsbutthemostfrequentlyusedclasses are, **FileReader** and **FileWriter**. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference isthat FileReader reads two bytes at atime and FileWriter writes two bytes at atime.

Wecanre-writetheaboveexample, which makestheuseofthesetwo classestocopyaninput file (having unicode characters) into an output file −

Example

```
importjava.io.*;
publicclassCopyFile{
  publicstaticvoidmain(Stringargs[])throwsIOException{
    FileReader  in  =
    null;    FileWriter
    out = null; try {
      in= new
      FileReader("input.txt");out=
      newFileWriter("output.txt");
      int c;
      while((c=in.read())!=-1){
        out.write(c);}
    }finally{
        if(in!=
        null){
```

Nowlet'shaveafile**input.txt** withthefollowingcontent−

```
Thisistestforcopyfile.
```

As a next step, compile the above programand execute it, whichwillresult increating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following −

```
$javacCopyFile.java
$javaCopyFile
```

StandardStreams

All the programming languages provide support for standard I/O where the user's program can takeinput froma keyboard andthenproduceanoutputonthe computer screen. Java providesthe following three standard streams −

- **StandardInput**−Thisisusedtofeedthedatatouser'sprogramandusuallyakeyboard is used as standard input stream and represented as**System.in**.
- **Standard Output** − This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as**System.out**.
- **Standard Error** − This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as**System.err**.

Followingisasimpleprogram,whichcreates**InputStreamReader**toreadstandardinput stream until the user types a "

Example

```
importjava.io.*;

publicclassReadConsole{

  publicstaticvoidmain(Stringargs[])throwsIOException{ InputStreamReader

    cin = null;

    try{

      cin= new InputStreamReader(System.in);

      System.out.println("Entercharacters,'q'toquit."); char

      c;

      do{

        c=(char)cin.read();

        System.out.print(c);

      }while(c !='q');

    }finally{

if(cin!=null){

        cin.close();

      }}}}
```

Thisprogramcontinuestoreadandoutputthesamecharacteruntilwepress'q'−

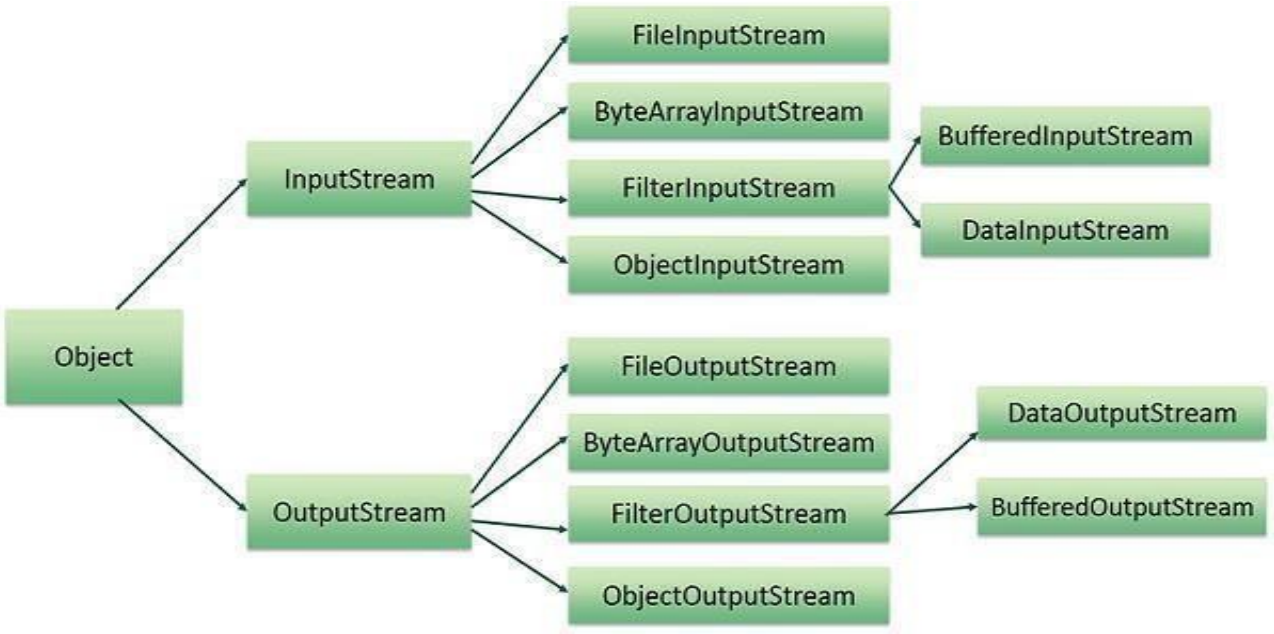
```
$javacReadConsole.java
$javaReadConsole
```

```
Entercharacters,'q'toquit. 1
1
e
e
q
q
```

ReadingandWriting Files

Asdescribedearlier, astreamcanbedefinedasasequenceofdata.The**InputStream** isusedto read data froma source and the **OutputStream** is used for writing data to a destination.

Hereisahierarchyofclasses to dealwithInputandOutputstreams.



Thetwoimportant streamsare**FileInputStream**and**FileOutputStream**

FileInputStream

Thisstreamisusedforreadingdata fromthefiles.Objectscanbecreatedusingthekeyword **new** and there areseveraltypesofconstructorsavailable

Followingconstructortakesafilenameasastringtocreateaninputstreamobjecttoreadthefile−

```
InputStreamf= newFileInputStream("C:/java/hello");
```

Followingconstructortakesafileobjecttocreateaninput streamobjecttoreadthefile. Firstwe create a file object using File() method as follows −

```
File f = new File("C:/java/hello");
InputStreamf=new FileInputStream(f);
```

Onceyouhave*InputStream*object inhand,thenthereisalist ofhelper methodswhichcanbe used to read to streamor to do other operations on the stream.

- ByteArrayInputStream
- DataInputStream

FileOutputStream

FileOutputStreamisusedtocreateafileandwritedataintoit.Thestreamwouldcreateafile, if it doesn't already exist, before opening it foroutput.

Herearetwoconstructorswhichcanbeusedtocreatea FileOutputStreamobject.

Followingconstructortakesafilenameasastringtocreateaninput streamobjecttowritethe file −

```
OutputStreamf=newFileOutputStream("C:/java/hello")
```

Followingconstructortakesafileobjecttocreateanoutputstreamobjectto writethefile. First, we create a file object using File() method as follows −

```
File f = new File("C:/java/hello");
OutputStreamf=newFileOutputStream(f);
```

Onceyouhave*OutputStream*object inhand,thenthereisa listofhelper methods, whichcanbe used to write to stream or to do other operations on the stream.

- ByteArrayOutputStream
- DataOutputStream

Example

FollowingistheexampletodemonstrateInputStreamandOutputStream−

```
importjava.io.*;
publicclassfileStreamTest{
  publicstaticvoidmain(Stringargs[]){
    try{
```

```
        byte bWrite []={11,21,3,40,5};
        OutputStream os=new FileOutputStream("test.txt");
        for(int x = 0; x <bWrite.length ; x++) {
          os.write(bWrite[x]);//writes the bytes} os.close();
        InputStream is=new FileInputStream("test.txt"); int
        size = is.available();
        for(int i = 0; i <size; i++) {
          System.out.print((char)is.read() + "");}
        is.close();
      } catch (IOException e) {
        System.out.print("Exception");


      }      }}
```

Java.io.RandomAccessFileClass

The**Java.io.RandomAccessFile**class filebehaves likea largearrayofbytesstoredinthe file system.Instances ofthis class support both reading and writing to a randomaccess file.

Class declaration

Followingisthedeclarationfor**Java.io.RandomAccessFile**class−

```
publicclassRandomAccessFile
  extends Object
    implementsDataOutput,DataInput,Closeable
```

Classconstructors

S.N.	Constructor&Description
1	**RandomAccessFile(Filefile,Stringmode)** Thiscreatesarandomaccess filestreamto readfrom, and optionallytowriteto,thefile specified by the File argument.

2	**RandomAccessFile(Filefile,Stringmode)** Thiscreatesarandomaccess filestreamtoreadfrom, andoptionallytowriteto, afilewith the specifiedname.

Methodsinherited

Thisclassinheritsmethodsfromthefollowingclasses−

- Java.io.Object

Java.io.FileClassinJava

The File class is Java's representation ofa file or directorypath name. Because file and directory names have different formats on different platforms, a simple string is not adequate to namethem. The File class contains several methods for working with the path name, deleting and renaming files, creating new directories,listing the contents of a directory, and determining several common attributes of files anddirectories.

- Itisanabstractrepresentationoffileanddirectorypathnames.
- A pathname, whether abstract or in string form can be either absolute or relative. The parent of an abstract pathname may be obtained by invoking the getParent() method of thisclass.
- First of all, we should create the File class object by passing the filename or directoryname to it. A file system may implement restrictions tocertain operations on the actual file-system object, such as reading, writing, and executing. These restrictions are collectively known as accesspermissions.
- Instances of the File class are immutable; that is, once created, the abstract pathname represented by a File object will neverchange.

HowtocreateaFileObject?

AFile object is created bypassing in a String that represents the name ofa file, or a String or another File object. For example,

```
Filea=newFile("/usr/local/bin/geeks");
```

defines an abstract file name for the geeks file in directory /usr/local/bin. This is an absolute abstract file name.

Programtocheckifa fileordirectory physically existornot.

```
//Inthisprogram,weacceptsafileordirectorynamefrom
//commandlinearguments.Thentheprogramwillcheckif
//thatfileordirectoryphysicallyexistor notand
//itdisplaysthepropertyofthatfileordirectory.
*import java.io.File;

//Displayingfileproperty class
fileProperty
{
   publicstaticvoidmain(String[] args){
```

```
        //acceptfilenameordirectorynamethroughcommand lineargs String
        fname =args[0];

        //passthefilenameordirectorynametoFileobject File f
        = new File(fname);

        //apply File class methods on File object
        System.out.println("File name :"+f.getName());
        System.out.println("Path: "+f.getPath());
        System.out.println("Absolute path:" +f.getAbsolutePath());
        System.out.println("Parent:"+f.getParent());
        System.out.println("Exists :"+f.exists());
        if(f.exists())
        {
           System.out.println("Is writeable:"+f.canWrite());
           System.out.println("Is readable"+f.canRead());
           System.out.println("Is a directory:"+f.isDirectory());
           System.out.println("File Size in bytes "+f.length());
        }
     }
}
```

Output:

```
Filename:file.txt Path:

file.txt

Absolutepath:C:\Users\akki\IdeaProjects\codewriting\src\file.txt

Parent:null

Exists:true

Iswriteable:true

Is readabletrue

Is a directory:false

FileSizeinbytes20
```

ConncetingtoDB

WhatisJDBCDriver?

JDBCdriversimplementthedefinedinterfacesintheJDBCAPI,forinteractingwithyour database server.

Forexample,usingJDBCdriversenable youtoopendatabaseconnectionsandtointeract withit by sending SQL or database commands then receiving results with Java.

The *Java.sql* packagethat ships withJDK, contains variousclasses withtheir behaviours defined and their actual implementaions are done in third-party drivers. Third party vendors implements the *java.sql.Driver* interface in their database driver.

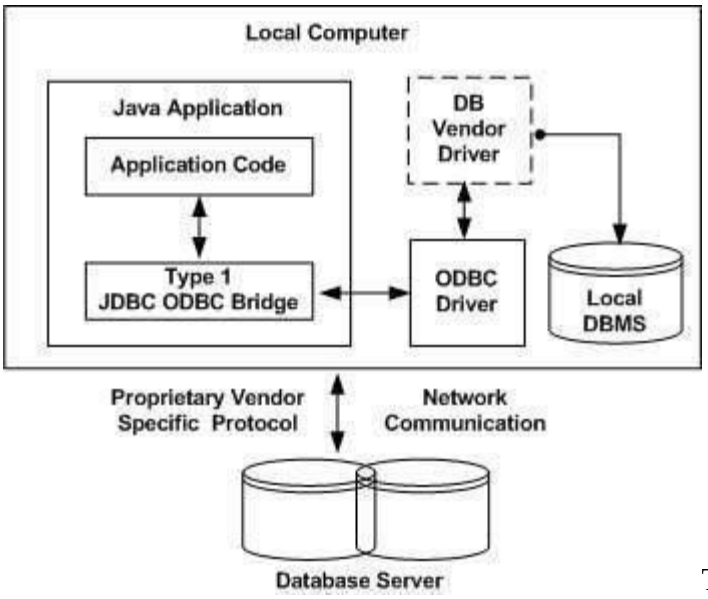
JDBCDriversTypes

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below −

Type1:JDBC-ODBCBridgeDriver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

WhenJava first came out,thiswas a usefuldriver because most databasesonlysupportedODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

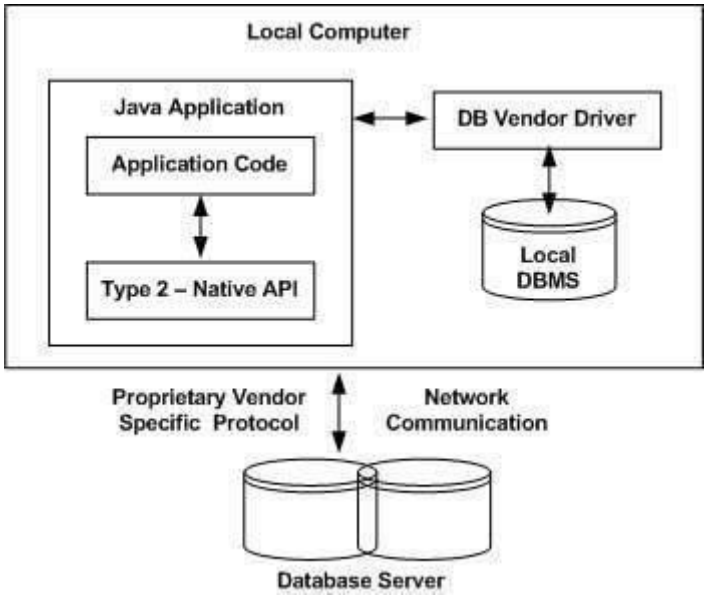


TheJDBC-ODBCBridgethatcomeswith JDK 1.2 is a good example of this kind ofdriver.

Type2:JDBC-NativeAPI

Ina Type 2 driver, JDBC API calls are convertedinto native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on each client machine.

If we change the Database, we have to change the native API, as it is specific to a database and they are mostly obsolete now, but you may realize some speed increase with a Type 2 driver, because it eliminates ODBC's overhead.

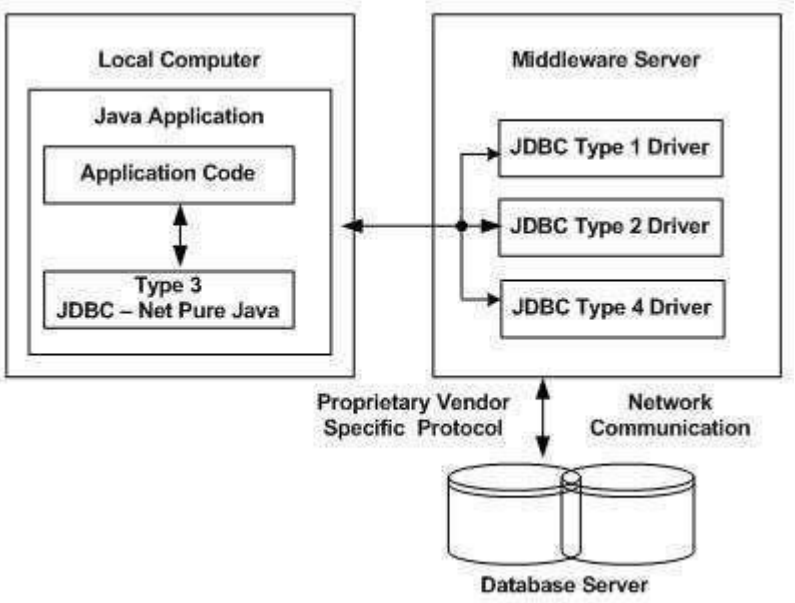


TheOracleCallInterface(OCI)driverisanexampleofaType2 driver.

Type3:JDBC-NetpureJava

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.

This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.



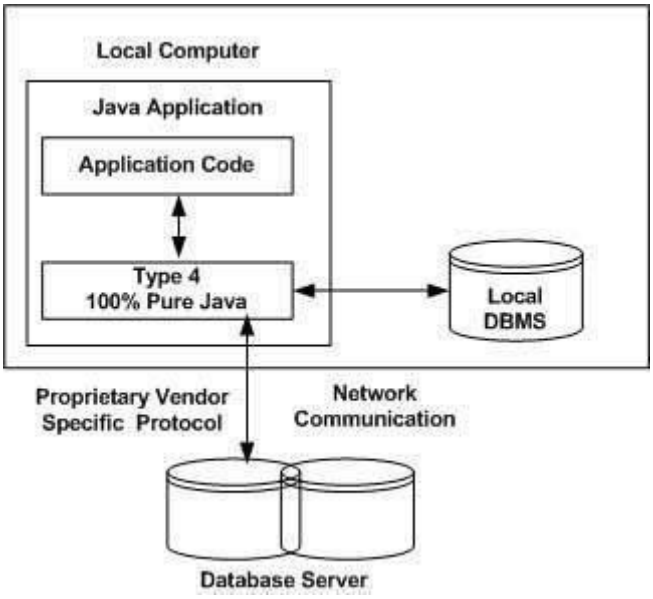
You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. Asaresult, you need someknowledgeoftheapplicationserver'sconfiguration in order to effectively use this driver type.

Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.

Type4:100%PureJava

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.

This kind of driver is extremely flexible,you don't need to installspecialsoftware on the client or server. Further, these drivers can be downloadeddynamically.



MySQL'sConnector/JdriverisaType4driver.Becauseoftheproprietarynatureoftheir network protocols, database vendors usually supply type 4 drivers.

WhichDrivershouldbeUsed?

Ifyouareaccessingonetypeofdatabase,suchasOracle,Sybase,orIBM,thepreferreddriver type is 4.

IfyourJavaapplicationisaccessingmultipletypesofdatabasesatthesametime,type3isthe preferred driver.

Type 2drivers are usefulinsituations,where a type3ortype4driverisnotavailableyetfor yourdatabase.

Thetype1driver isnotconsideredadeployment-leveldriver,and istypicallyused for development and testing purposes only.

Exampletoconnecttothemysqldatabaseinjava

Forconnecting javaapplicationwiththe mysqldatabase, youneedto follow5stepstoperform database connectivity.

InthisexamplewareusingMySqlasthedatabase.Soweneedtoknowfollowing informations for the mysql database:

1. **Driverclass:**Thedriverclassforthemysqldatabaseis**com.mysql.jdbc.Driver**.
2. **ConnectionURL:**TheconnectionURLforthemysqldatabase is **jdbc:mysql://localhost:3306/sonoo** where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we mayalso use IP address, 3306 is the port number and sonoo is the database name. We may use any database,in such case, you need to replace the sonoo with your databasename.
3. **Username:**Thedefaultusernameforthemysqldatabaseis**root**.
4. **Password:**Password isgivenbythe user at thetime ofinstalling the mysqldatabase. In this example, we are going to use root as thepassword.

Let'sfirstcreateatable inthe mysqldatabase,butbeforecreatingtable,weneedto create database first.

1. createdatabasesonoo;
2. usesonoo;
3. create table emp(id **int**(10),name varchar(40),age **int**(3));**ExampletoConnectJavaApplicationwith mysqldatabase**

Inthisexample, sonoo isthedatabasename,rootistheusernameandpassword.

```
importjava.sql.*;
classMysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connectioncon=DriverManager.getConnection("jdbc:mysql://llocalhost:3306/sonoo","root","root");
```

```
//heresonooisdatabasename,rootisusernameand password
Statement stmt=con.createStatement();
ResultSetrs=stmt.executeQuery("select*fromemp");
while(rs.next())
System.out.println(rs.getInt(1)+""+rs.getString(2)+""+rs.getString(3));
con.close();
}catch(Exceptione){System.out.println(e);}
}}
```

Theaboveexamplewill fetchalltherecordsofemptable.

Toconnectjavaapplicationwiththemysqldatabasemysqlconnector.jarfile isrequiredtobe loaded.

Twowaystoloadthejarfile:

1. pastethemysqlconnector.jar fileinjre/lib/extfolder
2. setclasspath

1) pastethemysqlconnector.jarfileinJRE/lib/extfolder:

Downloadthemysqlconnector.jarfile.Gotojre/lib/ext folderandpastethejarfilehere.

2) setclasspath:

Therearetwowaystosettheclasspath:

1. temporary2.permanent

Howtosetthetemporaryclasspath

opencommandprompt andwrite:

1. C:>setclasspath=c:\folder\mysql-connector-java-5.0.8-bin.jar;.;

Howtosetthepermanentclasspath

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to the mysqlconnector.jar file by appending mysqlconnector.jar;.; as C:\folder\mysql-connector-java-5.0.8-bin.jar;

JDBC-ResultSets

The SQL statements that read data from a database query, return the data in a result set. The SELECT statement is the standard way to select rows from a database and view them in a result set. The *java.sql.ResultSet* interface represents the result set of a database query.

AResultSetobject maintainsacursorthat pointstothecurrentrowintheresult set.Theterm "result set" refers to the row and column data contained in a ResultSet object.

The methodsoftheResultSetinterfacecanbebrokendownintothreecategories−

- **Navigationalmethods:**Usedtomovethecursoraround.
- **Getmethods:** Usedtoviewthedatainthecolumnsofthecurrentrowbeingpointedby thecursor.
- **Updatemethods:**Usedtoupdatethedatainthecolumnsofthecurrentrow.Theupdates can then be updated in the underlying database aswell.

Thecursoris movablebasedonthepropertiesoftheResultSet.Thesepropertiesaredesignated when the corresponding Statement that generates the ResultSet is created.

JDBCprovidesthefollowingconnectionmethodstocreatestatementswithdesiredResultSet−

- **createStatement(intRSType,intRSConcurrency);**
- **prepareStatement(String SQL,intRSType,intRSConcurrency);**
- **prepareCall(Stringsql,intRSType,intRSConcurrency);**

Thefirst argument indicatesthetypeofaResultSetobject andthesecondargument isoneoftwo ResultSet constants for specifying whether a result set is read-onlyor updatable.

TypeofResultSet

ThepossibleRSTypearegivenbelow.IfyoudonotspecifyanyResultSettype, youwill automatically get one that is TYPE_FORWARD_ONLY.

Type	Description
ResultSet.TYPE_FORWARD_ONLY	Thecursorcanonlymoveforwardintheresult set.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can scroll forward and backward, and the result set is not sensitive to changes made by otherstothedatabasethatoccuraftertheresultset was created.

ResultSet.TYPE_SCROLL_SENSITIVE.	The cursor can scroll forward and backward, and the result set is sensitive to changes made by otherstothedatabasethatoccuraftertheresultset was created.

ConcurrencyofResultSet

ThepossibleRSConcurrencyaregivenbelow.Ifyoudo notspecifyanyConcurrencytype, you will automatically get one that is CONCUR_READ_ONLY.

Concurrency	Description
ResultSet.CONCUR_READ_ONLY	Createsaread-onlyresultset.Thisisthedefault
ResultSet.CONCUR_UPDATABLE	Createsanupdateableresultset.

ViewingaResultSet

The ResultSet interface contains dozens of methods for getting the data of the current row.

Thereisaget methodforeachofthepossibledatatypes,andeachgetmethodhastwoversions

- Onethattakesinacolumnname.
- Onethattakesinacolumnindex.

Forexample, ifthecolumnyouare interested inviewingcontainsanint, youneedtouseoneof the getInt() methods of ResultSet −

S.N.	Methods&Description
1	**publicintgetInt(StringcolumnName)throwsSQLException** ReturnstheintinthecurrentrowinthecolumnnamedcolumnName.
2	**publicint getInt(intcolumnIndex)throwsSQLException** Returnstheint inthecurrent rowinthespecifiedcolumnindex.Thecolumnindexstarts at 1, meaning the first columnofa rowis 1,the second columnofa rowis 2, and so on.

Similarly, there are get methods in the ResultSet interface for each of the eight Java primitive types, as wellas common types such as java.lang.String, java.lang.Object, and java.net.URL.

There are also methods for getting SQL data types java.sql.Date, java.sql.Time, java.sql.TimeStamp, java.sql.Clob, and java.sql.Blob. Check the documentation for more information about using these SQL data types.

For abetterunderstanding, let usstudyViewing -ExampleCode.

Updatinga Result Set

TheResultSetinterfacecontainsacollectionofupdatemethodsforupdatingthedataofaresult set.

Aswiththegetmethods,therearetwo updatemethodsforeachdatatype −

- Onethattakesinacolumnname.
- Onethattakesinacolumnindex.

Forexample, to updateaStringcolumnofthecurrent rowofaresult set, youwoulduse oneofthe following updateString() methods −

S.N.	Methods&Description
1	**publicvoid updateString(intcolumnIndex,String s) throwsSQLException** ChangestheStringinthespecifiedcolumntothevalueofs.
2	**public void updateString(String columnName, String s) throws SQLException** Similartothepreviousmethod, exceptthatthecolumnisspecifiedbyitsname insteadof its index.

Thereareupdatemethodsfortheeight primitive datatypes, aswellasString, Object, URL, and the SQL data types in the java.sql package.

Updating a row in the result set changes the columns of the current row in the ResultSet object, but not in the underlying database. To update your changes to the row in the database, you need to invoke one of the following methods.

S.N.	Methods&Description
1	**publicvoidupdateRow()** Updatesthecurrentrowbyupdatingthecorrespondingrowinthe database.
2	**publicvoiddeleteRow()** Deletesthe currentrowfromthe database
3	**publicvoidrefreshRow()** Refreshesthedataintheresultsettoreflectanyrecentchanges inthe database.
4	**publicvoidcancelRowUpdates()** Cancelsanyupdatesmade onthecurrentrow.
5	**publicvoidinsertRow()** Insertsarowintothedatabase. This methodcanonlybe invokedwhenthecursoris pointing to the insert row.

UNIT-5

GUIProgrammingwithjava

TheAWTClasshierarchy

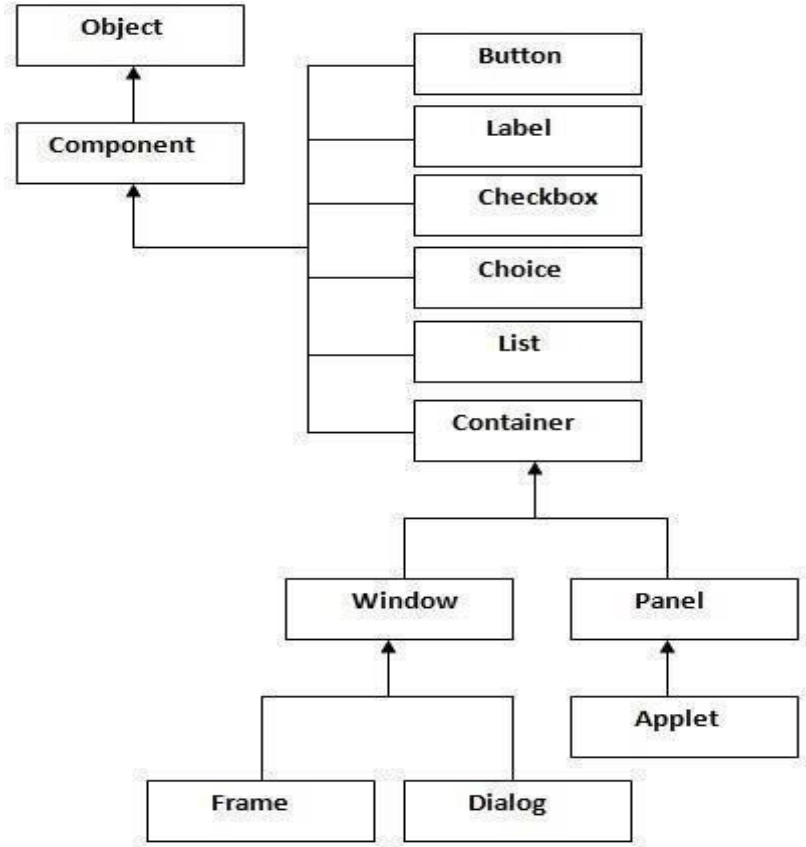
Java AWT(Abstract Window Toolkit) is*an APIto develop GUI or window-based applications* injava.

Java AWT components are platform-dependent i.e. components are displayed according to the viewofoperatingsystem.AWT isheavyweight i.e.itscomponentsareusingtheresourcesofOS.

Thejava.awtpackageprovidesclassesforAWTapisuchasTextField,Label,TextArea, RadioButton, CheckBox, Choice, List etc.

JavaAWTHierarchy

ThehierarchyofJavaAWTclassesaregivenbelow.



Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Window

Thewindowisthecontainerthathavenobordersandmenubars.Youmust useframe,dialogor another window for creating a window.

Panel

ThePanelisthecontainerthatdoesn't containtitlebarand menubars. Itcanhaveother components like button, textfield etc.

Frame

TheFrameisthecontainerthat containtitlebarandcanhave menubars. Itcanhaveother components like button, textfield etc.

UsefulMethodsofComponentclass

Method	Description
publicvoidadd(Componentc)	insertsacomponentonthiscomponent.
publicvoidsetSize(intwidth,int height)	setsthesize(widthandheight)ofthe component.
publicvoidsetLayout(LayoutManager m)	definesthe layoutmanagerforthecomponent.
publicvoidsetVisible(booleanstatus)	changesthevisibilityofthecomponent,bydefault false.

JavaAWTExample

Tocreatesimpleawtexample,youneedaframe. Therearetwowaystocreateaframe inAWT.

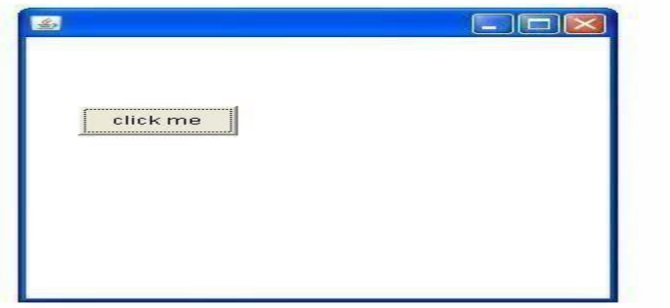
- ByextendingFrameclass(inheritance)
- BycreatingtheobjectofFrameclass(association)

AWTExamplebyInheritance

Let'sseeasimpleexampleofAWTwhereweareinheriting Frameclass.Here, weareshowing Button component on the Frame.

```
importjava.awt.*;
classFirstextendsFrame{
First(){
Button b=new Button("click me");
b.setBounds(30,100,80,30);// setting button position
add(b);//adding button into frame
setSize(300,300);//framesize300widthand300height
setLayout(null);//no layout manager
setVisible(true);//nowframewillbevisible,bydefaultnotvisible
}
publicstaticvoidmain(Stringargs[]){
First f=new First();
}}
```

ThesetBounds(intxaxis, intyaxis,int width,intheight)methodisusedintheaboveexamplethat sets the position of the awt button.



JavaSwing

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top ofAWT (Abstract Windowing Toolkit) API and entirely written in java.

UnlikeAWT,JavaSwingprovidesplatform-independentandlightweightcomponents.

Thejavax.swingpackageprovidesclassesforjavaswingAPIsuchasJButton,JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference betweenAWT andSwing.

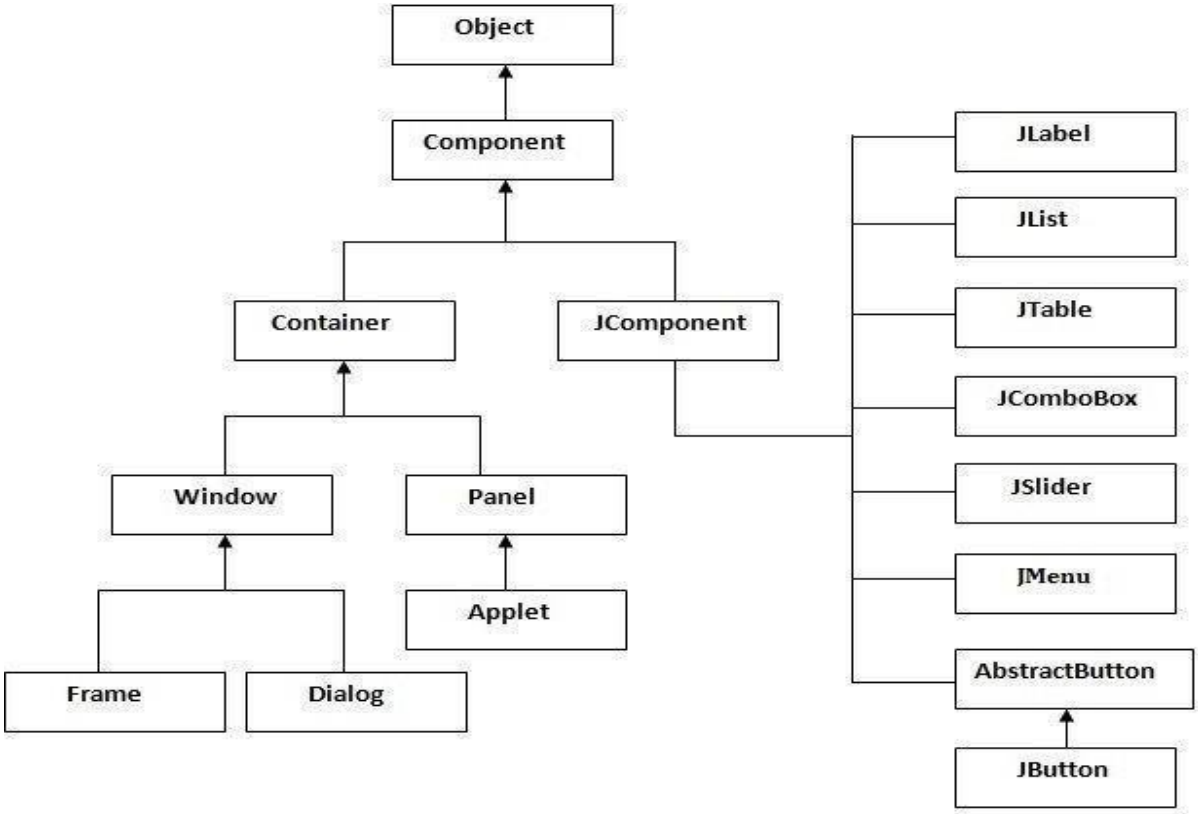
No.	JavaAWT	Java Swing
1)	AWT components are**platform-dependent**.	Javaswingcomponentsare**platform-independent**.
2)	AWTcomponentsare**heavyweight**.	Swingcomponentsare**lightweight**.
3)	AWT**doesn'tsupportpluggablelook and feel**.	Swing**supportspluggablelookand feel**.
4)	AWT provides**less components**than Swing.	Swing provides **more powerful components**such as tables, lists, scrollpanes, colorchooser, tabbedpaneetc.
5)	AWT **doesn't follows MVC**(Model View Controller) where model represents data, view represents presentation andcontroller acts as an interface between model andview.	Swing**followsMVC**.

CommonlyusedMethodsofComponentclass

Method	Description
publicvoidadd(Componentc)	addacomponent onanothercomponent.
publicvoidsetSize(intwidth,int height)	setssize ofthe component.
publicvoidsetLayout(LayoutManager m)	setsthelayoutmanagerforthecomponent.
publicvoidsetVisible(booleanb)	setsthevisibilityofthecomponent.Itisbydefault false.

HierarchyofJavaSwingclasses

Thehierarchyofjavaswing APIisgivenbelow.



JavaSwingExamples

Therearetwowaystocreateaframe:

- BycreatingtheobjectofFrameclass(association)
- ByextendingFrameclass(inheritance)

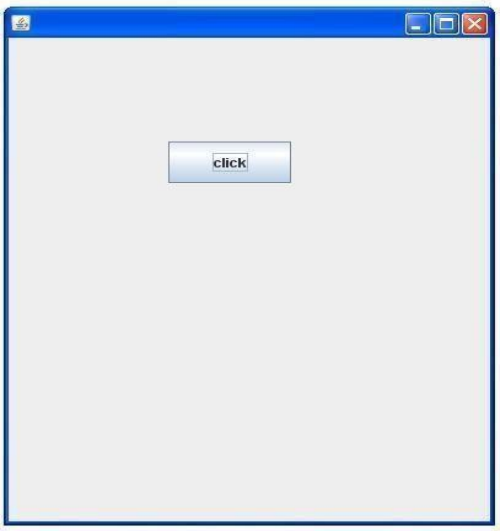
Wecanwritethecodeofswinginside themain(),constructororanyothermethod.

SimpleJavaSwingExample

Let'sseeasimpleswingexamplewherewearecreatingonebuttonandadding it ontheJFrame object inside the main() method.

File:FirstSwingExample.java

```
importjavax.swing.*;
public classFirstSwingExample{
publicstaticvoidmain(String[]args) {
JFramef=newJFrame();//creatinginstanceofJFrame
JButtonb=newJButton("click");//creatinginstanceofJButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height
f.add(b);//adding button in JFrame
f.setSize(400,500);//400widthand500height
f.setLayout(null);//usingno  layout  managers
f.setVisible(true);//making the frame visible
}}
```



Containers

JavaJFrame

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are addedto create aGUI.

UnlikeFrame, JFrame hastheoptiontohideorclosethewindowwiththehelpof setDefaultCloseOperation(int) method.

JFrameExample

```
importjava.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.Jpanel;
publicclassJFrameExample{
publicstaticvoidmain(Strings[]){
JFrameframe=newJFrame("JFrameExample");
JPanel panel = new JPanel();
panel.setLayout(newFlowLayout());
JLabellabel=newJLabel("JFrameByExample");
JButton button = new JButton();
button.setText("Button");
panel.add(label);
```

```
panel.add(button);
frame.add(panel);
frame.setSize(200, 300);
frame.setLocationRelativeTo(null);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
}}
```

JApplet

Aswepreferto AWT. NowwecanuseJApplet that canhaveallthecontrolsofswing. The JApplet class extends the Appletclass.

ExampleofEventHandlinginJApplet:

```
importjava.applet.*;
import javax.swing.*;
importjava.awt.event.*;
publicclassEventJAppletextendsJAppletimplementsActionListener{
JButton b;
JTextField tf;public
void init(){
tf=newJTextField();
tf.setBounds(30,40,150,20);
b=new JButton("Click");
b.setBounds(80,150,70,40);
add(b);add(tf);
b.addActionListener(this);
setLayout(null);
}
publicvoidactionPerformed(ActionEvente){
tf.setText("Welcome");
}}
```

Intheaboveexample,wehavecreatedallthecontrolsininit() methodbecauseit isinvoked onlyonce.

myapplet.html

```
1. <html>
2. <body>
3. <appletcode="EventJApplet.class"width="300"height="300">
```

</applet>
</body>
</html>

JDialog

TheJDialog controlrepresentsatoplevelwindowwithaborder and atitleusedtotakesome form of input from the user. It inherits the Dialogclass.

UnlikeJFrame,itdoesn'thavemaximizeandminimizebuttons.

JDialogclassdeclaration

Let'sseethedeclarationforjavax.swing.JDialogclass.

1. **public class** JDialog **extends** Dialog **implements** WindowConstants,Accessible,RootPaneConta iner

CommonlyusedConstructors:

Constructor	Description
JDialog()	Itisusedtocreateamodelessdialogwithoutatitleand without a specified Frame owner.
JDialog(Frameowner)	Itisusedtocreateamodelessdialogwithspecified Frame as its owner and an empty title.
JDialog(Frameowner,Stringtitle, boolean modal)	Itisusedtocreateadialogwiththespecifiedtitle, owner Frame andmodality.

JavaJDialogExample

```
importjavax.swing.*;
importjava.awt.*;
importjava.awt.event.*;
publicclassDialogExample{
private static JDialog d;
DialogExample(){
JFramef=newJFrame();
d=newJDialog(f,"DialogExample",true);
d.setLayout( new FlowLayout() );
JButton b = new JButton ("OK");
b.addActionListener ( newActionListener()
{
publicvoidactionPerformed(ActionEvente)
{
        DialogExample.d.setVisible(false);
}

});

d.add(newJLabel("Clickbuttontocontinue."));
d.add(b);
d.setSize(300,300);
d.setVisible(true);
}
publicstaticvoidmain(Stringargs[])
{
newDialogExample();
}}
```

Output:

JPanel

TheJPanelisasimplest container class. It providesspace inwhichanapplicationcanattachany other component. It inherits the JComponents class.

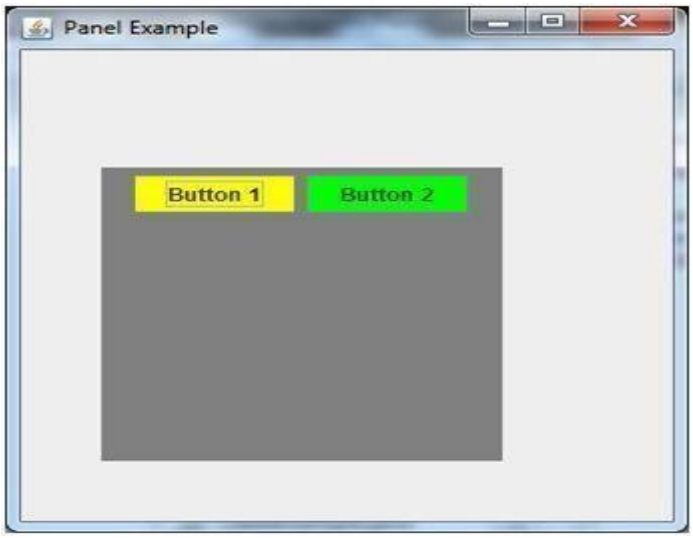
Itdoesn'thavetitlebar.

JPanelclassdeclaration

1.**public class** JPanel **extends** JComponent **implements** Accessible

JavaJPanelExample

```
import java.awt.*;
import javax.swing.*;public
class PanelExample{
    PanelExample()
      {
      JFramef=newJFrame("PanelExample");
      JPanel panel=new JPanel();
      panel.setBounds(40,80,200,200);
      panel.setBackground(Color.gray);
      JButtonb1=newJButton("Button1");
      b1.setBounds(50,100,80,30);
      b1.setBackground(Color.yellow);
      JButtonb2=newJButton("Button2");
      b2.setBounds(100,100,80,30);
      b2.setBackground(Color.green);
      panel.add(b1); panel.add(b2);
      f.add(panel);
            f.setSize(400,400);
            f.setLayout(null);
            f.setVisible(true);
      }
      publicstaticvoidmain(Stringargs[])
      {
      newPanelExample();
      }}
```



Overview of some Swing

ComponentsJavaJButton

TheJButtonclassisusedtocreatea labeledbuttonthat has platformindependentimplementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

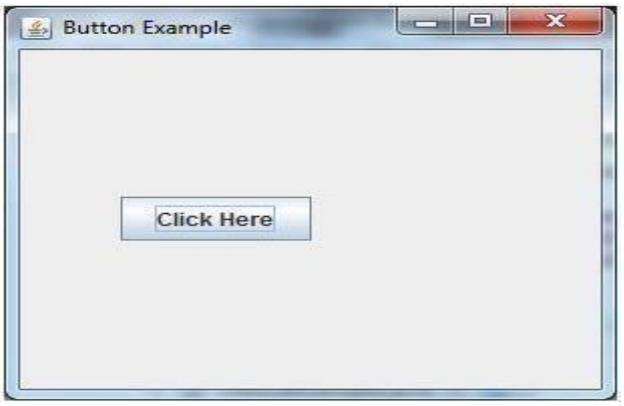
JButtonclassdeclaration

Let'sseethedeclarationforjavax.swing.JButtonclass.

1.**public class** JButton **extends** AbstractButton **implements** Accessible

Java JButtonExample

```
import javax.swing.*;
public class ButtonExample{
public static void main(String[] args) {
   JFramef=new JFrame("ButtonExample");
   JButton b=new JButton("Click Here");
   b.setBounds(50,100,95,30);
   f.add(b);
   f.setSize(400,400);
   f.setLayout(null);
   f.setVisible(true);}}
```



JavaJLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JLabelclassdeclaration

Let'sseethedeclarationforjavax.swing.JLabelclass.

1.**public class** JLabel **extends** JComponent **implements** SwingConstants,Accessible

CommonlyusedConstructors:

Constructor	Description
JLabel()	CreatesaJLabelinstancewithno imageandwithan empty string for the title.
JLabel(Strings)	CreatesaJLabelinstancewiththespecified text.
JLabel(Iconi)	CreatesaJLabelinstancewiththespecifiedimage.
JLabel(String s, Icon i, int horizontalAlignment)	CreatesaJLabelinstancewiththespecified text, image, andhorizontalalignment.

CommonlyusedMethods:

Methods	Description
StringgetText()	treturnsthetextstringthatalabeldisplays.
voidsetText(Stringtext)	Itdefinesthesinglelineoftextthiscomponentwill display.
void setHorizontalAlignment(int alignment)	Itsetsthealignmentofthelabel'scontentsalong the Xaxis.
IcongetIcon()	Itreturnsthegraphicimagethat thelabeldisplays.
int getHorizontalAlignment()	Itreturnsthealignmentofthelabel'scontentsalong the X axis.

JavaJLabelExample

```
importjavax.swing.*;
classLabelExample
{
publicstaticvoidmain(Stringargs[])
   {
   JFramef=newJFrame("LabelExample");
   JLabel l1,l2;
   l1=new JLabel("First Label.");
   l1.setBounds(50,50, 100,30);
   l2=newJLabel("SecondLabel.");
   l2.setBounds(50,100, 100,30);
   f.add(l1); f.add(l2);
   f.setSize(300,300);
   f.setLayout(null);
   f.setVisible(true);
   }
   }
```

JTextField

TheobjectofaJTextFieldclassisatext componentthat allowstheeditingofasingle linetext.It inherits JTextComponent class.

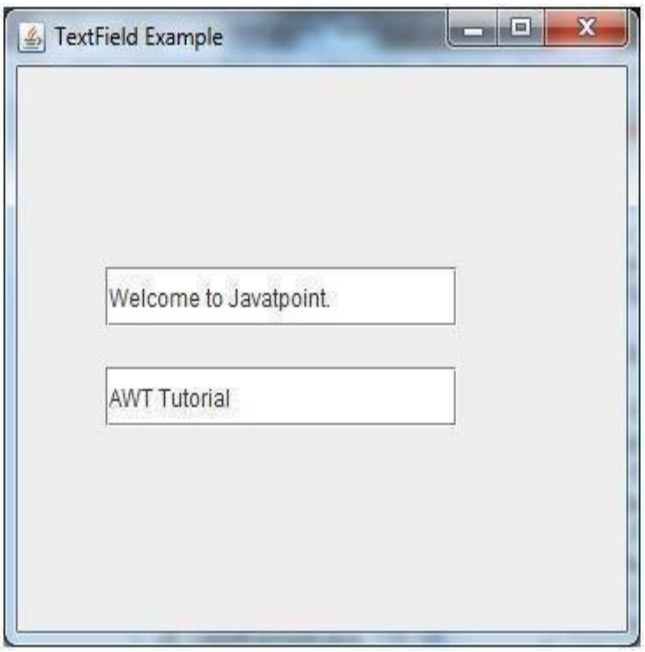
JTextFieldclassdeclaration

Let'sseethedeclarationforjavax.swing.JTextField class.

1.**public class**JTextField**extends**JTextComponent**implements**SwingConstants

JavaJTextFieldExample

```
importjavax.swing.*;
classTextFieldExample
{
publicstaticvoidmain(Stringargs[])
   {
   JFramef=newJFrame("TextFieldExample");
   JTextField t1,t2;
   t1=newJTextField("WelcometoJavatpoint.");
   t1.setBounds(50,100, 200,30);
   t2=new JTextField("AWT Tutorial");
   t2.setBounds(50,150, 200,30);
   f.add(t1); f.add(t2);
   f.setSize(400,400);
   f.setLayout(null);
   f.setVisible(true);
   }  }
```



JavaJTextArea

TheobjectofaJTextAreaclassisa multilineregionthat displaystext.It allowstheeditingof multiple line text. It inherits JTextComponent class

JTextAreaclassdeclaration

Let'sseethedeclarationforjavax.swing.JTextAreaclass.

1. **public class**JTextArea **extends**JTextComponent

Java JTextArea Example

```
importjavax.swing.*;
publicclassTextAreaExample
{
    TextAreaExample(){
      JFramef=newJFrame();
      JTextAreaarea=newJTextArea("Welcometojavatpoint");
      area.setBounds(10,30, 200,200);
      f.add(area);
      f.setSize(300,300);
      f.setLayout(null);
      f.setVisible(true);
    }
publicstaticvoidmain(Stringargs[])
   {
  newTextAreaExample();
   }}
```

SimpleJavaApplications

```
import javax.swing.JFrame;
importjavax.swing.SwingUtilities;

publicclassExampleextendsJFrame{

   public Example() {
     setTitle("Simpleexample");
     setSize(300, 200);setLocationRelativeTo(null);
     setDefaultCloseOperation(EXIT_ON_CLOSE);
   }

   publicstaticvoidmain(String[]args){
     Example ex = new Example();
     ex.setVisible(true);
   }}
```

LayoutManagement

JavaLayoutManagers

TheLayoutManagersareusedtoarrangecomponentsinaparticularmanner.LayoutManageris an interface that is implemented by all the classes of layoutmanagers.

BorderLayout

TheBorderLayoutprovidesfiveconstantsforeachregion:

1. **publicstaticfinalint NORTH**
2. **publicstaticfinalintSOUTH**
3. **publicstaticfinalintEAST**
4. **publicstaticfinalintWEST**
5. **publicstaticfinalintCENTER**

ConstructorsofBorderLayoutclass:

- **BorderLayout():**createsaborderlayoutbutwithnogapsbetweenthecomponents.
- **JBorderLayout(inthgap,intvgap):**createsaborderlayout withthegivenhorizontaland vertical gaps between thecomponents.

ExampleofBorderLayoutclass:

```
importjava.awt.*;
importjavax.swing.*;
public class Border
 {
JFramef;
Border()
{
   f=newJFrame();
   JButton b1=new JButton("NORTH");;
   JButton b2=new JButton("SOUTH");;
   JButton b3=new JButton("EAST");;
   JButton b4=new JButton("WEST");;
   JButtonb5=new JButton("CENTER");;
   f.add(b1,BorderLayout.NORTH);
   f.add(b2,BorderLayout.SOUTH);
   f.add(b3,BorderLayout.EAST);
   f.add(b4,BorderLayout.WEST);
   f.add(b5,BorderLayout.CENTER);
   f.setSize(300,300);
   f.setVisible(true);
}
publicstaticvoidmain(String[]args)
{
  newBorder();
}}
```

Output:

JavaGridLayout

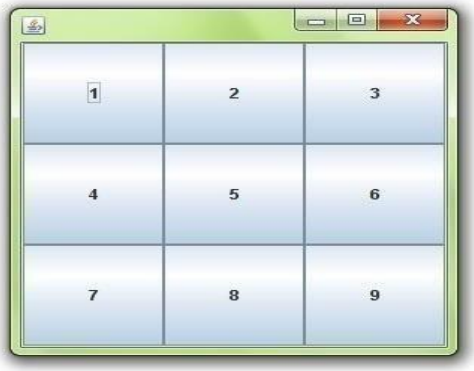
TheGridLayoutisusedtoarrangethecomponentsinrectangulargrid.Onecomponent is displayed in each rectangle.

ConstructorsofGridLayoutclass

1. **GridLayout():**createsagridlayoutwithonecolumnpercomponentinarow.
2. **GridLayout(introws,intcolumns):**createsagridlayout withthegivenrowsand columns but no gaps between the components.
3. **GridLayout(introws,intcolumns,inthgap,intvgap):**createsagrid layout withthe given rows and columns alongwith given horizontal and vertical gaps.

ExampleofGridLayoutclass

```
1. importjava.awt.*;
2. import
javax.swing.*;publicclass
MyGridLayout{ JFramef;
MyGridLayout(){
   f=newJFrame();
   JButton b1=new JButton("1");
   JButton b2=new JButton("2");
   JButton b3=new JButton("3");
   JButton b4=new JButton("4");
   JButton b5=new JButton("5");
   JButton b6=new JButton("6");
   JButton b7=new JButton("7");
   JButton b8=new JButton("8");
   JButton b9=new JButton("9");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
   f.add(b6);f.add(b7);f.add(b8);f.add(b9);
   f.setLayout(new GridLayout(3,3));
   //settinggridlayoutof3rowsand3columns
   f.setSize(300,300);
   f.setVisible(true);
}
publicstaticvoidmain(String[]args) {
 newMyGridLayout();}}
```



JavaFlowLayout

TheFlowLayout isusedtoarrangethecomponentsina line,oneafteranother(inaflow).Itisthe default layout of applet or panel.

FieldsofFlowLayoutclass

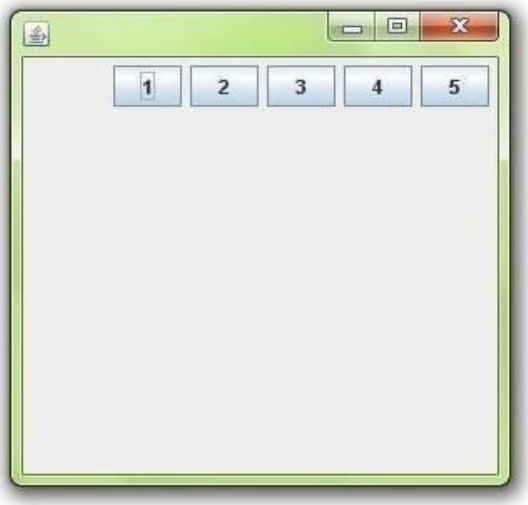
1. **publicstaticfinalintLEFT**
2. **publicstaticfinalintRIGHT**
3. **publicstaticfinalintCENTER**
4. **publicstaticfinalintLEADING**
5. **publicstaticfinalintTRAILING**

ConstructorsofFlowLayoutclass

1. **FlowLayout():**createsaflowlayoutwithcenteredalignment andadefault5unit horizontal and verticalgap.
2. **FlowLayout(intalign):**createsaflowlayout withthegivenalignment andadefault 5 unit horizontal and verticalgap.
3. **FlowLayout(intalign,inthgap,intvgap):**createsaflowlayout withthegiven alignment and the given horizontal and verticalgap.

Example ofFlowLayoutclass

```
importjava.awt.*;
importjavax.swing.*;public
class MyFlowLayout{
JFramef;
MyFlowLayout(){
   f=newJFrame();
   JButtonb1=newJButton("1");
   JButtonb2=newJButton("2");
   JButtonb3=newJButton("3");
   JButtonb4=newJButton("4");
   JButtonb5=newJButton("5");
   f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
   f.setLayout(newFlowLayout(FlowLayout.RIGHT));
   //settingflowlayoutofrightalignment
  f.setSize(300,300);
   f.setVisible(true);
}
publicstaticvoidmain(String[]args) {
newMyFlowLayout();
}}
```



EventHandling

EventandListener(JavaEventHandling)

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc.Thejava.awt.event packageprovides manyevent classesand Listener interfaces for event handling.

TypesofEvent

Theeventscanbebroadlyclassifiedintotwocategories:

- **Foreground Events** - Those events which require the direct interaction of user.They are generated as consequences of a person interacting with the graphical components in GraphicalUser Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard,selecting an item from list, scrolling the pageetc.

- **BackgroundEvents**-Thoseeventsthatrequiretheinteractionofenduserareknownas

backgroundevents.Operatingsysteminterrupts,hardwareorsoftwarefailure,timer expires, an operation completion are the example of background events.

EventHandling

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events.Let's have a brief introduction to this model.

TheDelegationEventModelhasthefollowingkeyparticipantsnamely:

- **Source** - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provide as with classes for sourceobject.
- **Listener** - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns.

EventclassesandListenerinterfaces:

EventClasses	ListenerInterfaces
ActionEvent	ActionListener
MouseEvent	MouseListenerandMouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

StepstoperformEventHandling

Followingstepsarerequired toperformeventhandling:

1. ImplementtheListenerinterfaceandoverrides itsmethods
2. RegisterthecomponentwiththeListener

Forregisteringthecomponent withtheListener, manyclassesprovidetheregistrationmethods. For example:

- **Button**
  - publicvoidaddActionListener(ActionListenera){}
- **MenuItem**
  - publicvoidaddActionListener(ActionListenera){}
- **TextField**
  - publicvoidaddActionListener(ActionListenera){}
  - publicvoidaddTextListener(TextListenera){}
- **TextArea**
  - publicvoidaddTextListener(TextListenera){}
- **Checkbox**
  - publicvoidaddItemListener(ItemListenera){}
- **Choice**
  - publicvoidaddItemListener(ItemListenera){}
- **List**
  - publicvoidaddActionListener(ActionListenera){}
  - publicvoidaddItemListener(ItemListenera){}

EventHandlingCodes:

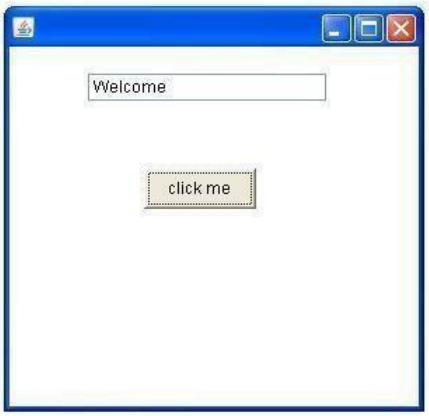
Wecanputtheeventhandlingcodeintooneofthefollowing places:

1. Sameclass
2. Otherclass
3. Annonymousclass

Exampleofeventhandlingwithinclass:

```
importjava.awt.*;
importjava.awt.event.*;
classAEvent extendsFrameimplementsActionListener{
TextField tf;
```

```
AEvent(){
tf=new TextField();
tf.setBounds(60,50,170,20);
Buttonb=newButton("clickme");
b.setBounds(100,120,80,30);
b.addActionListener(this);
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
publicvoidactionPerformed(ActionEvente){
tf.setText("Welcome");
}
publicstaticvoid main(Stringargs[]){
newAEvent();
}}
```

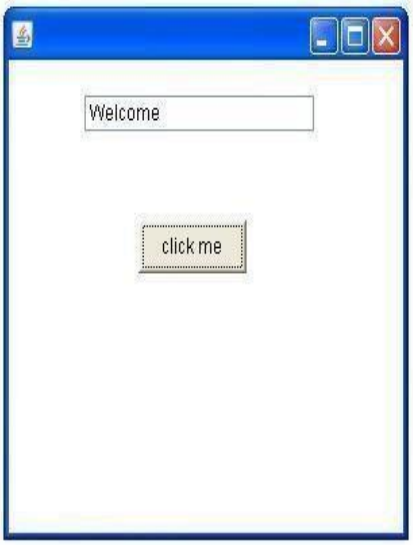


publicvoidsetBounds(intxaxis,intyaxis,intwidth,intheight);havebeenused inthe above example that sets the position ofthe component it may be button, textfield etc.

JavaeventhandlingbyimplementingActionListener

```
importjava.awt.*;
importjava.awt.event.*;
classAEvent extendsFrameimplementsActionListener{
TextField tf;
AEvent(){
//createcomponentstf=new
TextField();
tf.setBounds(60,50,170,20);
Buttonb=newButton("clickme");
b.setBounds(100,120,80,30);
//register listener
b.addActionListener(this);//passing current instance
//addcomponentsandsetsize,layoutandvisibility add(b);add(tf
setSize(300,300);
setLayout(null);
setVisible(true);
}
publicvoidactionPerformed(ActionEvente){
tf.setText("Welcome");
}
publicstaticvoid main(Stringargs[]){
newAEvent();

}}
```



JavaMouseListenerInterface

The Java MouseListener is notified whenever you change the stateofmouse. It isnotified against MouseEvent. The MouseListener interface is found in java.awt.event package. It hasfive methods.

MethodsofMouseListenerinterface

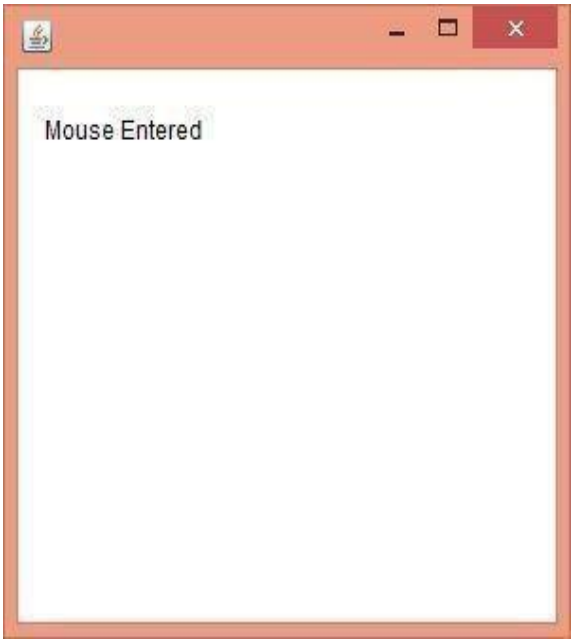
Thesignatureof5methodsfoundinMouseListener interfacearegivenbelow:

1. publicabstractvoidmouseClicked(MouseEvente);
2. publicabstractvoidmouseEntered(MouseEvente);
3. publicabstractvoidmouseExited(MouseEvente);
4. publicabstractvoidmousePressed(MouseEvent e);
5. publicabstractvoidmouseReleased(MouseEvente);

JavaMouseListenerExample

```
importjava.awt.*;import
java.awt.event.*;
publicclassMouseListenerExampleextendsFrameimplementsMouseListener{
   Label l;
   MouseListenerExample(){
      addMouseListener(this);
      l=new Label();
      l.setBounds(20,50,100,20);
      add(l);
      setSize(300,300);
      setLayout(null);
      setVisible(true);
   }
   publicvoidmouseClicked(MouseEvente){
      l.setText("MouseClicked");
   }
   publicvoidmouseEntered(MouseEvente){
      l.setText("MouseEntered");
   }
   publicvoidmouseExited(MouseEvente){
      l.setText("Mouse Exited");
   }
   publicvoidmousePressed(MouseEvente){
      l.setText("Mouse Pressed");
   }
   publicvoidmouseReleased(MouseEvente){
      l.setText("Mouse Released");
   }
publicstaticvoidmain(String[]args){
   new MouseListenerExample();


}}
```



JavaKeyListenerInterface

The Java KeyListener is notified whenever you change the state of key. It is notified against KeyEvent.TheKeyListenerinterfaceisfoundinjava.awt.event package.Ithasthreemethods.

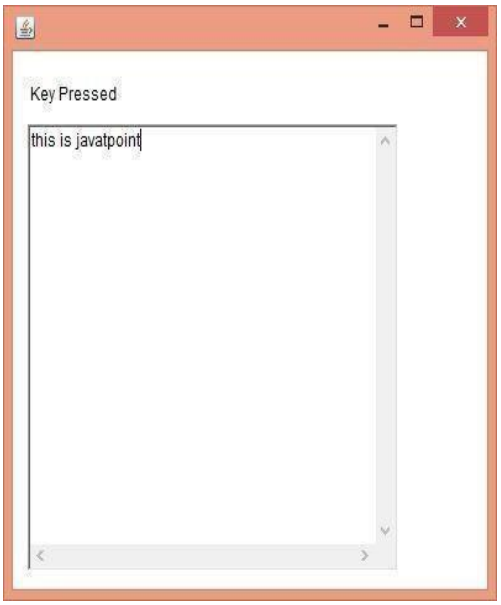
Methods ofKeyListenerinterface

Thesignatureof3methodsfoundinKeyListenerinterfacearegivenbelow:

1. publicabstractvoidkeyPressed(KeyEvente);
2. publicabstractvoidkeyReleased(KeyEvente);
3. publicabstractvoidkeyTyped(KeyEvente);

JavaKeyListenerExample

```
importjava.awt.*;import
java.awt.event.*;
publicclassKeyListenerExampleextendsFrameimplementsKeyListener{
   Label l;
   TextArea area;
   KeyListenerExample(){
      l=new Label();
      l.setBounds(20,50,100,20);
      area=new TextArea();
      area.setBounds(20,80,300,300);
      area.addKeyListener(this);
      add(l);add(area);
      setSize(400,400);
      setLayout(null);setVisible(true);
   }
   publicvoidkeyPressed(KeyEvente){
      l.setText("KeyPressed");
   }
   publicvoidkeyReleased(KeyEvente){
      l.setText("Key Released");
   }
   publicvoidkeyTyped(KeyEvente){
      l.setText("KeyTyped");
   }
   publicstaticvoidmain(String[]args){
      new KeyListenerExample(); }}
```



JavaAdapterClasses

Java adapter classes *provide the default implementation of listener interfaces*. If you inherit the adapter class, you will not be forced to provide the implementation of all the methods of listener interfaces. So it *saves code*.

java.awt.eventAdapterclasses

Adapterclass	Listenerinterface
WindowAdapter	WindowListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
FocusAdapter	FocusListener
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
HierarchyBoundsAdapter	HierarchyBoundsListener

JavaWindowAdapterExample


```
1. importjava.awt.*;
importjava.awt.event.*;public
classAdapterExample{
   Frame f;
   AdapterExample(){
      f=new Frame("Window Adapter");
      f.addWindowListener(newWindowAdapter(){
         publicvoidwindowClosing(WindowEvente){
            f.dispose();     }    });
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
   }
publicstaticvoidmain(String[]args) {
   newAdapterExample();
}}
```

Applets

Appletisaspecialtypeofprogramthat isembeddedinthewebpagetogeneratethedynamic content. It runs inside the browser and works at client side.

AdvantageofApplet

Thereare manyadvantagesofapplet.Theyare asfollows:

- Itworksatclientsidesolessresponsetime.
- Secured
- Itcanbeexecutedbybrowsersrunningunder manyplateforms, includingLinux, Windows, Mac Osetc.

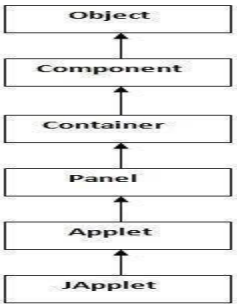
DrawbackofApplet

- Pluginisrequiredatclientbrowsertoexecuteapplet.

LifecycleofJavaApplet

1. Applet isinitialized.
2. Applet isstarted.
3. Applet ispainted.
4. Applet isstopped.
5. Applet isdestroyed.

HierarchyofApplet



LifecyclemethodsforApplet:

Thejava.applet.Appletclass4lifecyclemethodsandjava.awt.Componentclassprovides1life cycle methods for an applet.

java.applet.Appletclass

Forcreatinganyappletjava.applet.Applet classmust beinherited.Itprovides4lifecyclemethods ofapplet.

1. **publicvoidinit():**isusedtoinitializedtheApplet.Itis invokedonlyonce.
2. **publicvoidstart():**isinvokedaftertheinit()methodorbrowserismaximized.Itisused to start theApplet.
3. **publicvoidstop():**isusedtostoptheApplet.ItisinvokedwhenApplet isstopor browser isminimized.
4. **publicvoiddestroy():**isusedtodestroytheApplet.Itisinvokedonlyonce.

java.awt.Componentclass

TheComponentclassprovides1lifecyclemethodofapplet.

1. **publicvoidpaint(Graphicsg):**isusedtopainttheApplet.ItprovidesGraphicsclass object that can be used for drawing oval, rectangle, arc etc.

SimpleexampleofAppletbyhtmlfile:

Toexecutetheapplet byhtmlfile,createanappletandcompile it.Afterthatcreateanhtmlfile and place the applet code in html file. Now click the html file.

```
1.//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{
public void paint(Graphics g){
g.drawString("welcome",150,150);
}
}
```

SimpleexampleofAppletbyappletviewertool:

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. Afterthat run it by: appletviewer First.java. NowHtmlfile isnot required but it is for testing purpose only.

```
1.//First.java
import java.applet.Applet;
import java.awt.Graphics;
public class First extends Applet{
public void paint(Graphics g){
g.drawString("welcometoapplet",150,150);
}
}
/*
<appletcode="First.class"width="300"height="300">
</applet>
*/
```

Toexecutetheappletbyappletviewertool, write incommand prompt:

```
c:\>javacFirst.java
c:\>appletviewerFirst.java
```

DifferencebetweenAppletandApplicationprogramming

	Java Applet	Java Application
User graphics	Inherently graphical	Optional
Memory requirements	Java application requirements plus web browser requirements	Minimal java application requirements
Distribution	Linked via HTML and transported via HTTP	Loaded from the file system or by a custom class loading process
Environmental input	Browser client location and size; parameters embedded in the host HTML document	command-line parameters
Method expected by the virtual Machine	init- initialization method start-startup method stop pause/ deactive method destroy-termination method paint-drawing method	Main - startup method
Typical applications	public-access order-entry systems for the web, online multimedia persentations, web page animation	Network server, multimedia kiosks, developer tools, appliance and consumer electronics control and navigation.

ParameterinApplet

Wecanget anyinformationfromtheHTMLfileasaparameter.Forthispurpose,Applet class provides a method named getParameter(). Syntax:

1.**public**StringgetParameter(StringparameterName)

ExampleofusingparameterinApplet:

```
1.  importjava.applet.Applet;
2.  importjava.awt.Graphics;
3.  publicclassUseParamextends Applet
4.{
5. publicvoidpaint(Graphicsg)
6. {
7.  Stringstr=getParameter("msg");
8.  g.drawString(str,50,50);
9. }}
```

myapplet.html

```
1.  <html>
2.  <body>
3.  <appletcode="UseParam.class"width="300"height="300">
4.  <paramname="msg"value="Welcometoapplet">
5.  </applet>
6.  </body>
7.  </html>
```