GANAPATI INSTITUTE OF ENGINEERING & TECHNOLOGY (POLYTECHNIC)

Mathasahi , Jagatpur , Cuttack - 754200

LECTURE NOTES ON

PROGRAMMING THROUGH JAVA

PREPARED BY

BANINANDA ACHARYA

(LECTURER IN CSE DEPT IN G.I.E.T,JAGATPUR)

UNIT –I

INTRODUCTION TO JAVA

What is Java

Java is a high Level programming language and it is also called as a platform. Java is a secured and robust high level object-oriented programming language.

Platform: Any software or hardware environment in which a program runs is known as a platform. Java has its own runtime environment (JRE) and API so java is also called as platform.

Java follows the concept of **WriteOnce,Run Anywhere.**

Application of java

1. DesktopApplications
2. Web Applications
3. Mobile
4. Enterprise Applications
5. Smart Card
6. EmbeddedSystem
7. Games
8. Robotics etc

History of Java

James Gosling, Patrick Naughton and Mike Sheridan initiated the Java language project in 1991. Team of sun engineers designed for small, embeddedsystemsinelectronicappliancesliketset-topboxes.Initiallyitwas called "Greentalk" later it was called Oak .

Java is an open source software produced by Sunmicro system under the terms of the GNU General Public License (GPL) .

Features of Java:

- **Object Oriented** – Java implements basic concepts of Object oriented programming System (OOPS) ie Object, Class, Inheritance, Polymorphism, Abstraction, Encapsulation. In Java, everything is an Object. Java can be easily extended since it is based on the Objectmodel.
- **Platform Independent** − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine,rather into platform independent bytecode. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** – Java fallows the basic Syntax of C,C++. If you understand the basic concept of OOPS then it is easy to master in java.
- **Secure** − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Portable** − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- **Robust** − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtimechecking.
- **Multithreaded** − With Java's multithreaded feature In java we can write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- **Interpreted** − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- **High Performance** − With the use of Just-In-Time compilers, Java enables high performance.
- **Distributed** − Java is designed for the distributed environment of theinternet.
- **Dynamic** − Java is considered to be more dynamic than C or C++ sinceitisdesignedtoadapttoanevolvingenvironment.Java

Programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

Object Oriented Programming System(OOPS)

Object means a real word entity such as pen, chair, table etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

If any language fallows the OOPS concepts that language we call it as object oriented language

Procedure to write simple java Program

To write a java program First we have install the JDK.

To create a simple java program, you need to create a class that contains main method. Let's understand the requirement first.

- Install the JDK and install it.
- Set path of the jdk
- Create the java program
- Compile and run the java program

Setting Up the Path for Windows

Assuming you have installed Javain*c:\ProgramFiles\java\jdk*directory−

- Right-click on 'MyComputer' and select 'Properties'.
- Click the 'Environmentvariables' button under the'Advanced'tab.
- Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to'C:\WINDOWS\SYSTEM32', then change your path to read'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting Up The Path for Linux,UNIX,Solaris,FreeBSD

Environment variable PATH should be set to point to where the Javabinaries have been installed. Refer to your shell documentation, if you have trouble doing this. For Example if you use *bash* as your shell, then you would add the following line to the end of your'.bashrc:exportPATH=
/path/to/java:$PATH'

Popular Java Editors

- **Notepad** − On Windows machine, you can use any simple text editor like Notepad , TextPad.
- **Netbeans** − A Java IDE that is open-source and free which can be downloaded from **Eclipse**− A Java IDE developed by the eclipse open-source community and can be downloaded from

JVM(Java VirtualMachine)

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

What is JVM

It is:

1. **A specification** where working of Java Virtual Machine is specified. But implementation provider is independent to choose the algorithm. Its implementation has been provided by Sun and other companies.
2. **An implementation** Its implementation is known as JRE (Java Runtime Environment).
3. **Runtime Instance** Whenever you write java command on the command prompt to run the java class, an instance of JVM is created.

What it does

The JVM Performs following operation:

- Loads code
- Verifies code
- Executes code
- Provides runtime

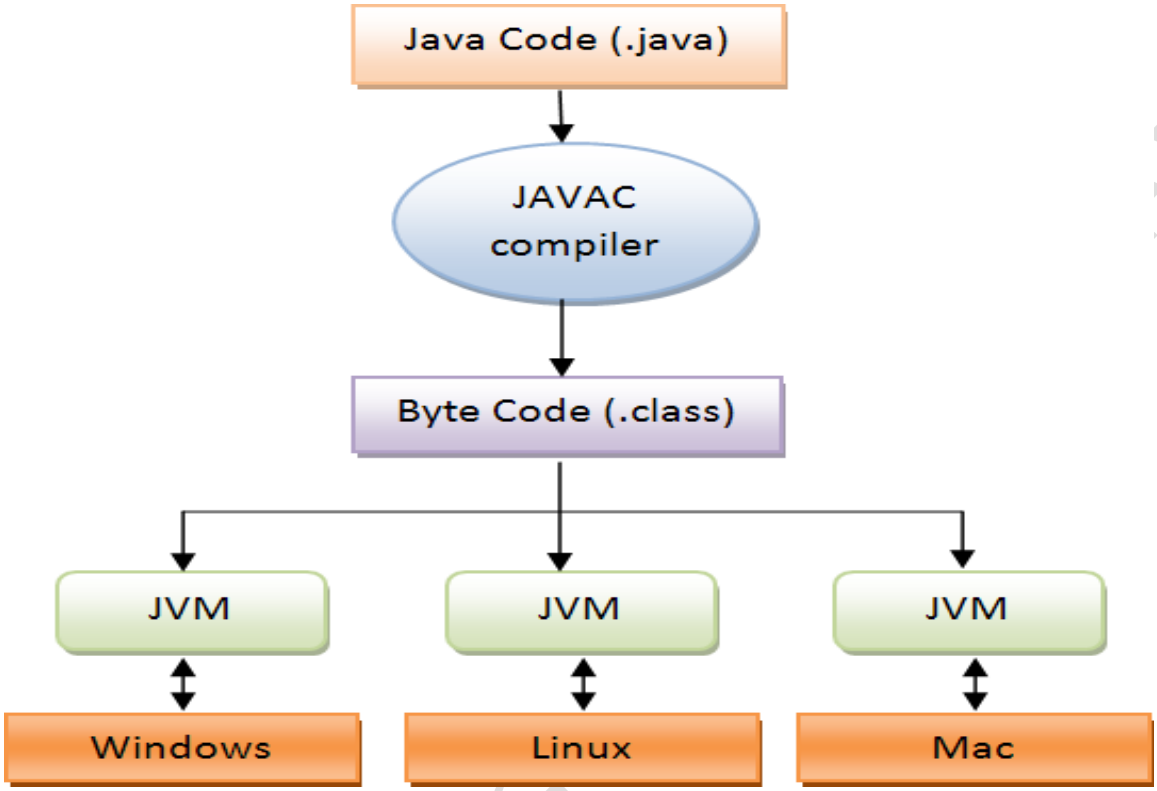
environment JVM provides

definitions for the:

- Memory area
- Class file format

- Register set
- Garbage-collectedheap
- Fatal error reporting etc.

JAVA VIRTUAL MACHINE



Java's Magic: The Bytecode output of a Java compiler is not executablecode. Rather, it is bytecode. Bytecode is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM).

Java complier translates the *java source* code into *byte code* or intermediate code ,not the executable file .JVMtake the byte code and convert into executable code corresponding to Operating system

Becauseof theabove featurejava is portable

UNIT –II

Introduction To ObjectOrientedProgrammingSystem(OOPS)

Object means a real word entity such as pen, chair, table etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

If any language fallows the OOPS concepts that language we call it as object oriented language

CLASS,OBJECT AND METHODS

Java program is a collection of objects that communicate via invoking each other's methods. We now briefly look intoclass, object, methods, and instance variables.

Class−A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

A class is declared by use of the class keyword. A simplified general form ofa class definition is shown here:

```
class classname

{

 Type instance-variable1; type instance-variable2;

 // ... type instance-

variableN; type method

name1(parameter-list)

{//body of method}

type method name2(parameter-list)

 {//body of method}

//... type method name N(parameter-list)

      {//body of method}}
```

The data, or variables, defined within a class are called instance variables. The code is contained within methods. Collectively, the methods and variables defined within a class are called members of the class. In most classes, the instance variables are acted upon and accessed by the methods defined for that class

Simple Class

Class Sample

{

int len,

fl

oat ht void

get()

{ //body

} }

Here a class Sample contains two variable len and ht

Object in Java

Object is the physical as well as logical entity where as class is the logical entity only.

An object has three characteristics:

- **state:**representsdata (value)ofan object.
- **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
- **identity:**Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But, it is used internally by the JVM to identify each object uniquely.

Object is an instance of a class. Classisatemplateorblueprintfrom which objects are created. So object is the instance(result) of a class.

Object Definitions:

- Objectis*arealworldentity*. Objectis*aruntimeentity*.
- Objectis *anentitywhichhasstateand behavior*.
- Objectis*an instance ofaclass*.

Samples=newSample() heres is an object for the class Sample

*new*operator isused to create an object

Methods − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

Let's create the Simple java program:

```
1. class Sample{
2.     publicstaticvoidmain(String args[]){
3.      System.out.println("Howareyou");
4.     }
5.}
```

save this file as Sample.java

To compile: javac Sample.java

To execute: Java Sample

Java Identifiers

All Java components require names. Names used for classes, variables, and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows −

- All identifiers should begin with a letter(AtoZ or atoz),currency character ($) or an underscore (_).
- After the first Character,identifiers canhave any combination of characters.
- A keyword cannot be used as an identifier.
- Most importantly, identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, 1_value.
- Examples of illegal identifiers: 123abc, -salary.

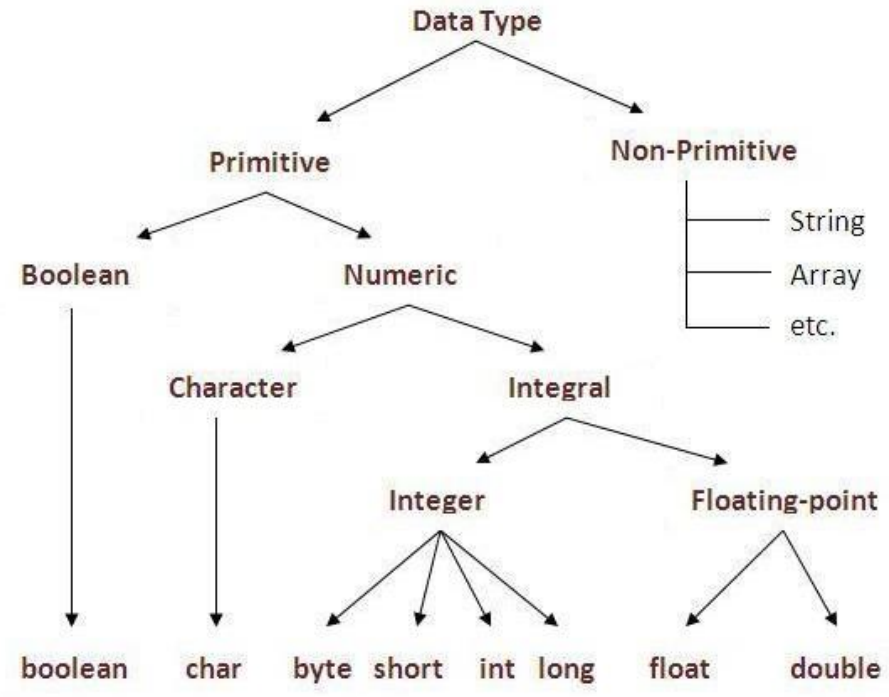
Java Modifiers: There are two categories of modifiers−

- **AccessModifiers**−default, public, protected, private
- **Non-accessModifiers**−final,abstract,strictfp

DATA TYPES

There are two datatypes available in Java−

- PrimitiveDataTypes
- NonPrimitiveTypes



Primitive Data Types

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a keyword. Let us now look into the eight primitive data types in detail.

byte

- Byte data type is an8-bitsignedtwo'scomplementinteger
- Minimum value is-128 (-2^7)
- Maximum value is127(inclusive)(2^7-1)
- Default value is 0
- Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer.
- Example: byte a = 100, byte b = -50

short

- Short data type is a16-bit signed two's complement integer
- Minimum value is-32,768(-2^15)
- Maximum value is32,767(inclusive)(2^15-1)
- Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer
- Default value is 0.
- Example: short s = 10000, short r = -20000

int

- Int data type isa32-bit signed two's complement integer.
- Minimum value is-2,147,483,648(-2^31)
- Maximum value is2,147,483,647(inclusive)(2^31 -1)
- Integer is generally used as the default datatype for integral values unless there is a concern about memory.
- The default value is 0
- Example: int a = 100000, int b = -200000

long

- Long datatype is a 64-bitsigned two's complement integer
- Minimum value is -9,223,372,036,854,775,808(-2^63)
- Maximum value is9,223,372,036,854,775,807(inclusive)(2^63-1)
- This type is used when a wide rrange than int is needed
- Default value is 0L
- Example: long a = 100000L, long b = -200000L

float

- Float data type is a single-precision32-bitIEEE754floatingpoint
- Float is mainly used to save memory inlarge arrays of floating point numbers
- Default value is 0.0f
- Float data type is never used for precise values such as currency
- Example:float f1 = 234.5f

double

- Double datatype is a double-precision64-bitIEEE754floating point
- This datatype is generally used as the default datatype for decimal values, generally the default choice
- Double datatype should never be used or recisevaluessuchascurrency
- Default value is 0.0d
- Example:double d1 = 123.4

boolean

- Boolean datatype represents one bit of information
- There are only two possible values: true and false
- This datatype is used for simple flags thattracktrue/false conditions
- Default value is false
- Example:booleanone=true

char

- Char data type is a single16-bit Unicode character

- Minimum valueis'\u0000'(or0)
- Maximum valueis'\uffff'(or65,535inclusive)
- Char datatype is used to store any character
- Example:char letterA='A'

Java Literals

A literal is a source code representation of a fixed value. Literals can be assigned to any primitive type variable.

Byte a = 68;char a = 'A'

byte, int, long, and short can be expressed in decimal(base 10), hexadecimal(base 16) or octal(base 8) number systems as well.

Prefix 0 is used to indicate octal, and prefix 0x indicates hexadecimal when using these number systems for literals. For example −

Int decimal=100;int octal=0144;int hexa =0x64;

String literals in Java are specified like they are in most other languages by enclosing a sequence of characters between a pair of double quotes. Examples of string literals are −

Example

"HelloWorld" "two\nlines" "\"Thisisinquotes\""

Stringandchartypesofliteralscancontainanyunicodecharacters.For example − char a = '\u0001'; String a = "\u0001";

Javalanguage supportsfew specialescape sequencesfor Stringand char

Notation	Character represented
\n	Newline (0x0a)
\r	Carriagereturn (0x0d)
\f	Formfeed(0x0c)
\b	Backspace (0x08)
\s	Space (0x20)
\t	Tab
\"	Double quote
\'	Single quote
\\	Backslash

\ddd Octal character (ddd)

\uxxxx HexadecimalUNICODEcharacter (xxxx)

Java Variable Example:AddTwoNumbers

```
1. class Sample{
2. publicstaticvoidmain(String[] args){
3. int i=50;
4. int j=60;
5. int k=a+b;
6. System.out.println(k);
7. }}
```

Java VariableExample :Widening

```
1. class Sample{
2. publicstaticvoidmain(String[] args){
3. int j=10;
4. float k=a;
5. System.out.println(i);
6. System.out.println(j);
7. }}
```

Unicode System

Unicode is a universal international standard character encoding that is capable of representing most of the world's written languages.

Before Unicode, there were many language standards:

- **ASCII** (American Standard Code for Information Interchange) for theUnited States.
- **ISO8859-1**forWesternEuropeanLanguage.
- **KOI-8**for Russian.
- **GB18030andBIG-5**forchinese,andsoon.

Java Tokens

Java Tokens are the smallest individual building block or smallest unit of a Java program, it is used by the Java compiler for constructing expressions and statements. Java program is collection different types of tokens, comments, and white spaces.

JavaSupportsFiveTypesofTokens:

- ReservedKeywords Identifiers Literals
- Operators Separators

JavaKeywordscannotbeusedasavariable name.

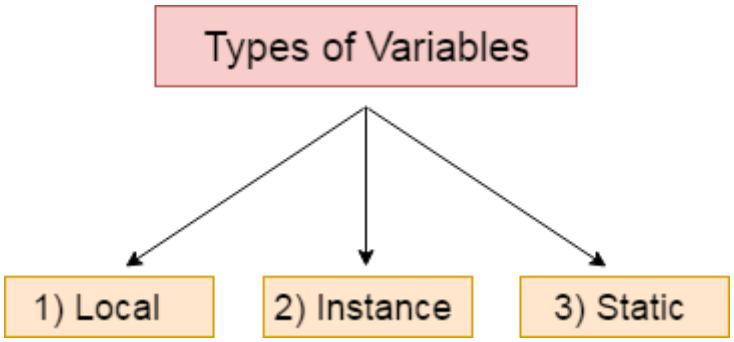
Abstract	Assert	boolean	break
Byte	Case	catch	char
Class	Const	continue	default
Do	Double	else	enum
extends	Final	finally	float
For	Goto	if	implements
Import	Instanceof	int	interface
Long	Native	new	package
private	Protected	public	return
Short	Static	strictfp	super
Switch	synchronized	this	throw
throws	Transient	try	void
volatile	While	true	false
Null			

Variable

Variable is name of *reserved area allocated inmemory*. In other words, it is a *name of memory location*. It is a combination of "vary + able" that means its value can be changed.

Thereare three typesof variables in java:

- local variable
- instance variable
- static variable



1) Local Variable

A variable which is declared inside the method is called local variable.

2) Instance Variable

A variable which is declared inside the class but outside the method,is called instance variable . It is not declared as static.

3) Static variable

A variable that is declared as static is called static variable.It cannot be local.

OPERATORS & IF,Switch,loopStatements

Operators in java

Operator in ava is a symbol that is used to perform operations.For example: +, -, *, / etc.

There are many types of operators in java which are givenbelow:

- Unary Operator,
- Arithmetic Operator,
- shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- TernaryOperatorand
- Assignment Operator.

Java If-elseStatement

The Java *ifstatement* is used to test the condition.It checks boolean condition: *true* or *false*. There are various types of if statement in java.

- if statement if-else statement
- if-else-if ladder nestedif statement

Java IF Statement

The Java if statement tests the condition. It executes the *if block* if condition is true.The following is the syntax

```
1. if(condition){
2. //codeto be executed
3. }
```

```
1. public class Example {
2. publicstaticvoid main(String[]args){
3.     intk=35;
4.     if(k>18){       System.out.print("Hello");
5.     }  }}
```

IF-else Statement

The if-else statement in java tests the condition.It executes the *if block* if condition is true otherwise *else block* is executed.

Syntax:

```
1. if(condition){
2. //code ifcondition is true
3. }else{
4. //code ifcondition is false
5. }
```

```
Public class Sample {
publicstaticvoidmain(String[]args){
    int n=23;
    if(number%2==0){System.
        out.println("even");
    }else{
        System.out.println("odd");
    }   }
}
```

IF-else-ifladder Statement

The if-else-if ladder statement executes one condition from multiple statements.

Syntax:

```
if(condition1){
//codetobeexecutedifcondition1istrue
}else if(condition2){
//codetobeexecutedifcondition2istrue
}
else if(condition3){
//codetobeexecutedifcondition3istrue
}
...
        else{
        //codetobeexecutedifalltheconditionsare false
        }
```

```
public class Simple {
publicstaticvoid main(String[]args){
    int marks=70;

    if(marks<40){
        System.out.println("FAIL");
    }
    elseif(marks>=40&&marks<50){
        System.out.println("Dgrade");
            }
            elseif(marks>=50&&marks<60){
                System.out.println("Cgrade");
            }
            elseif(marks>=60&&marks<70){
                System.out.println("Bgrade");
            }
            elseif(marks>=70&&marks<80){
                System.out.println("Agrade");
            }elseif(marks>=80&&marks<100){
                System.out.println("A+grade");
            }else{
                System.out.println("Invalid!");
            }
        }
        }
```

Switch Statement

The *switch statement* in java executes one statement from multipleconditions. It is like if-else-if ladder statement.

Syntax:

```
1. switch(expression){
2. case value1:
3.  //codeto be executed;
4.  break;//optional
5. case value2:
6.  //codeto be executed;
7.  break;//optional
8. ......
9.
10.         default:
11.          codetobeexecutedifallcasesarenotmatched;
12.         }
```

Example:

```
1. public class Sample {
2. publicstaticvoid main(String[]args){
3.     int k=20;
4.     switch(k){
5.     case 10: System.out.println("10");break;
6.     case 20: System.out.println("20");break;
7.     case 30: System.out.println("30");break;
8.     default:System.out.println("Notin 10,20 or 30");
9.     }  }
10.         }
```

Java ForLoop

The Java *for loop* is used to iterate a part of the program several times. If the number of iteration is fixed, it is recommended to use for loop.

Therearethree typesoffor loopin java.

- Simple For Loop
- For-eachorEnhancedForLoop
- LabeledForLoop

Java Simple For Loop

The simple for loop is same as C/C++.We can initialize variable,check condition and increment/decrement value.

Syntax:

```
1. for(initialization;condition;incr/decr){
2. //codeto be executed
3. }
```

Example:

```
1. public class Sample {
2. public static void main(String[]args){
3.     for(int i=1;i<=20;i++){
4.         System.out.println(i);
5.     }
6.}
7.}
```

Java While Loop

The Java *while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:

```
1. while(condition){
2. //codeto be executed
3. }
```

```
1. public class Sample {
2. publicstaticvoid main(String[]args){
3.     int j=1;
4.     while(j<=10){
5.         System.out.println(j);
6.     j++;
7.     }
8. }}
```

Java do-while Loop

The Java *do-while loop* is used to iterate a part of the program several times. If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.The Java *do-while loop*is executed at least once because condition is checked after loop body.

Syntax:do{/code to be executed

}while(condition);

```
Public class Example {
Public static void
    main(String[]args){ int j=1;
    do{
            System.out.prin
        tln(j);j++;
     }while(j<=10);
        }}
```

Java Break Statement

The Java *break* is used to break loop or switch statement. It breaks the current flow of the program at specified condition. In case of inner loop, it breaks only inner loop.

Example:

```
1. public class Simple {
2. publicstaticvoid main(String[]args){
3.     for(int i=1;i<=10;i++){
4.         if(i==5){
5.             break;
6.         }
7.         System.out.println(i);
8.     }
9.}
10.     }
```

Java Continue Statement

The Java *continue statement* is used to continue loop. It continues the current flow of the program and skips the remaining code at specified condition. In case of inner loop, it continues only inner loop.

Example:

```
1. public class Sample {
```

```
2.  public static void main(String[]args){
3.      for(int k=1;k<=10;k++){
4.          if(k==5){
5.              continue;
6.          }
7.          System.out.println(k);
8.      }
9.}
10.         }
```

ARRAYS & COMMENTS in JAVA

Array in java

Arrayisgroupofelements ofsimilartypesoccupyingcontiguousmemorylocations

Advantage of Java Array

- **CodeOptimization Random access**

Disadvantage ofJava Array is thatfixedsizeitcannotgrow

There are two types of array in java .

- Single Dimensional Array
- Multidimensional Array

Single Dimensional Array in java Syntax to Declare an Array in java

1. datatype[] arrayname (or)
2. datatype []arrayname;(or)
3. datatype arrayname[];

Array initialization in java

1. arrayname=newdatatype[size];

Example

Let's see the simple example of java array,where we are going to declare, instantiate, initialize and traverse an array.

```
1. class Testarray{
2. public static void main(String args[]){
3. int a[]=newint[5];//declarationandinstantiation
4. a[0]=10;      a[1]=20;    a[2]=70;    a[3]=40;a[4]=50;
5.   for(inti=0;i<a.length;i++)//length isthe propertyof array
6. System.out.println(a[i]);
7.  }}
```

Declaration,Instantiation and Initialization of Java Array

We can declare,instantiate and initialize the java array together by:

```
1. Int
```

a[]={33,3,4,5};//declaration,instantiationandinitialization Let's

see the simple example to print this array.

```
1. class Testarray1{
2. publicstatic
voidmain(Stringargs[]){3.
4.int
a[]={33,3,4,5};//declaration,instantiationandinitialization5.
6. //printing array
7. for(int i=0;i<a.length;i++)//length is the property of array
8. System.out.println(a[i]);
9.
10.         }}
```

Multi dimensional array in java

In such case, data is stored in row and column based index (also known as matrix form).

Syntax to Declare Multidimensional Array in java

```
1. dataType[][]arrayRefVar;(or)dataType[][]arrayRefVar;(or)
2. dataTypearrayRefVar[][];(or)      dataType[]arrayRefVar[];
```

Example to instantiate Multidimensional Array in java

```
1. int[][]arr=new int[3][3];//3rowand3column
```

Example to initializeMultidimensional Array in java

```
1. arr[0][0]=1;
2. arr[0][1]=2;
3. arr[0][2]=3;
4. arr[1][0]=4;
5. arr[1][1]=5;
6. arr[1][2]=6;
```

```
7. arr[2][0]=7;
8. arr[2][1]=8;
9. arr[2][2]=9;
```

Example of Multidimensional java array

Let's see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

```
1. class Ex{
2. publicstaticvoidmain(Stringargs
[]){3.
4. //declaringand initializing2D array
5. int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
6.
7. //printing 2D array
8. for(int i=0;i<3;i++){
9.  for(int j=0;j<3;j++){
10.          System.out.print(arr[i][j]+"");
11.        }
12.        System.out.println();
13.       }
14.        }  }
```

Java Comments

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

Types of Java Comments

There are3 types ofcomments in java.

1. Single Line Comment
2. Multi Line Comment
3. DocumentationComment

1) Java SingleLine Comment

The single line comment is used to comment only one line.

Syntax:

```
1. //This is single line comment
```

Example:

```
1. public class Sample{
2. public static void main(String[]args){
3.     int j=10;//Here,j is a variable
4.     System.out.println(j);
5.}
6.}
```

2) Java MultiLine Comment

The multi line comment is used to comment multiplelines of code.

Syntax:

```
1. /*
2. This
3. is
4. multiline
5. comment
6. */
```

Example:

```
1. Public class CommentExample2{
2. Public static void main(String[]args){
3. /* Let's declare and
4.  Print variable in java.*/
5.     int j=10;
6.     System.out.println(j);
7.}
8.}
```

3) Java Documentation Comment

The documentation comment is used to create documentation API. To create documentation API, you need to use **javadoc tool**.

Syntax:

```
1. /**
2. This
3. is
4. documentation
5. comment
```

6. */

CONSTRUCTORS

Constructor is special member function ,it has the same name as class name. It is called when an instance of object is created and memory is allocated for the object.

It is a special type of method which is used to initialize the object

Rules for creating java constructor

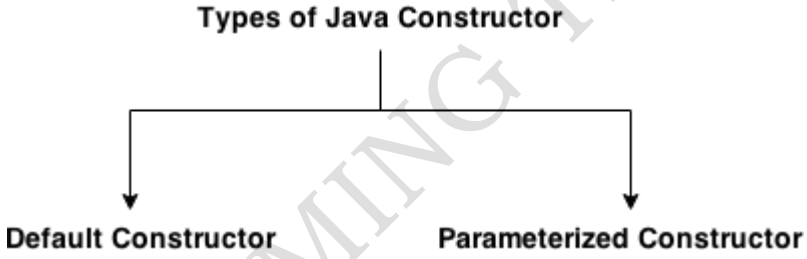
There are basically two rules defined for the constructor.

1. Constructor name must be same as its class name
2. Constructormusthavenoexplicitreturntype

Types of java constructors

There are two types of constructors in java:

1. Default constructor(no-arg constructor)
2. Parameterized constructor



A constructor is called "Default Constructor" when it doesn't have anyparameter.

```
Class
Sample{Sa
mple()
 {
  System.out.println("Sample is created");
 }
 Public static void main(String args[])
  {
    Sample b=new Sample();
  }  }
```

A constructor which has a specific number of parameters is calledparameterized constructor.

```
1. class Student4{
2.     int id;
3.     String name;
4.     Student4(int i,String n){
5.     id= i;
6.     name = n;
7.     }
8.     Void
Display(){System.out.println(id+""+name);}9.
10.            publicstaticvoidmain(String args[]){
11.            Student4s1=new Student4(111,"Karan");
12.            Student4s2=new Student4(222,"Aryan");
13.            s1.display();
14.            s2.display();
15.          }
16.       }
```

Difference between constructor and method in java

There are many differences between constructors and methods. They are given below.

Java Constructor	**Java Method**
Constructor is used to initialize the state of anobject.	Methodisusedtoexpose behaviour of an object.
Constructormustnothavereturntype.	Method must have return type.
Constructorisinvoked implicitly.	Methodisinvokedexplicitly.
The java compiler provides a default constructor if you don't have anyconstructor.	Methodisnotprovidedby compiler in any case.
Constructornamemustbesameasthe class name.	Methodnamemayormaynot be same as class name.

Java static keyword

The **static keyword** in java is used for memory management mainly.We can apply java static keyword with variables, methods, blocks and nested class. The static keyword belongs to the class than instance of the class.

The static canbe:

1. variable(also known as class variable)
2. method(alsoknown as class method)

3. block
4. nested class

1) Java static variable

If you declar eany variable as static,it is known static variable.

- The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees,college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Advantage of static variable:It makes your program memory efficient (i.e it saves memory).

```
class
 Stud{
 int
 rollno;
 Stringn
 ame;
 staticStringcollege="ITS";
 Stud(int r,String n){
 rolln
 o=r;
 nam
 e= n;
 }
Void display(){System.out.println(rollno+""+name+""+college);}

 Public static void main(String
args[]){ Stud s1 = new
Stud(11,"Krishna"); Stud s2 =
new Stud(22,"Rama");s1.display();
s2.display();
}    }
```

2) Java static method

If you apply static keyword with any method,it is known as static method.

- A static method belongs to the class rather than object of a class.
- A static method can be invoked without then eed for creating an instance of a class.
- Static method can access static datamember and can change the value of it.

Example of static method

```
1.  //Programofchanging thecommon propertyof allobjects(static field).

2.
3.  class Stud{
4.       int rollno;
5.       String name;
6.       StaticStringcollege="BE
C";7.
8.       Static void change(){
9.       college = "JBIEIT";
10.              }
11.              Stud(intr, String n){
12.              rollno=r;
13.              name = n;
14.              }
15.
16.              Void display(){System.out.println(rollno+""+name+""+college
    );}
17.               Public staticvoid main(String args[]){
18.             Stud.change();
19.               Studs1=newStud(11,"Kiran");
20.             Studs2=newStud(22,"Arjun");
21.             Studs3=newStud(33,"srinu");
22.               s1.display();
23.             s2.display();
24.             s3.display();
25.             }
26.         }
```

This keyword in java

There can be a lot of usage of java this keyword. In java, this is a **reference variable** that refers to the current object.

Usage of java this keyword

Here is given the6 usage of javathis keyword.

1. This can be used to refer current class instance variable.
2. This can be used to invoke current class method(implicitly)
3. this()can be used to invoke current class constructor.
4. This can be passed as an argument in the methodcall.
5. This can be passed as argument in the constructor call.
6. This can be used to return the current class instance from the method.

1) this:torefercurrentclassinstancevariable

The this keyword can be used to refer current class instance variable. If there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

```
Class
Student{
int rollno;
String
name;
float fee;
Student(introllno,Stringname,floatfee){this.r
ollno=rollno;
this.name=name;
this.fee=fee;
}
Void display(){System.out.println(rollno+""+name+""+fee);}
}
class TestThis2{
public static void main(String args[]){
Student s1=new
Student(111,"ankit",5000f);
Students2=newStudent(112,"sumit",60
00f);s1.display();
s2.display();   }}
```

2) this:toinvokecurrentclassmethod

You may invoke the method of the current class by using the this keyword.If you don't use the this keyword, compiler automatically adds this keyword while invoking the method

```
1. class B{
2. voidm(){System.out.println("hellom");}
3. voidn(){
4. System.out.println("hellon");
5. //m();//same as this.m()
6. this.m();
7. }
8. }
9. class TestThis4{
10.     publicstaticvoidmain(String args[]){
11.     B b=new B();
12.     b.n();
13.     }}
```

3) this():toinvokecurrentclassconstructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

Callingdefaultconstructorfromparameterizedconstructor:

```
1. class B{
2. B(){System.out.println("hello");}
3. B(int x){
4. this();
5. System.out.println(x);
6. }
7. }
8. class Sample{
9. publicstaticvoidmain(String args[]){
10.         Ba=newB(10);
11.         }}
```

```
1. class Student{
2. int rollno;
3. String name,course;
4. float fee;
5. Student(int rollno,String name,String course){
6. this.rollno=rollno;
7. this.name=name;
8. this.course=course;
9. }
10.         Student(int rollno,String name,String course,float fee){
11.         this(rollno,name,course);//reusingconstructor
12.         this.fee=fee;
13.         }
14.         voiddisplay(){System.out.println(rollno+""+name+""+course+""
    +fee);}
15.         }
16.         class SamplTest{
17.         publicstaticvoidmain(String args[]){
18.         Student s1=newStudent(111,"ankit","java");
19.         Student s2=newStudent(112,"sumit","java",6000f);
20.         s1.display();
21.         s2.display();
22.         }}
```

4) this:topass asanargument inthe method

The this keyword can also be passedas an argument in the method.Itis mainly used in the event handling. Let's see the example:

```
1. class S2{
2.   voidm(S2obj){
3.   System.out.println("methodisinvoked");
4.   }
5.   voidp(){
6.   m(this);
```

```
7.  }
8.  publicstaticvoidmain(String args[]){
9.  S2s1 =newS2();
10.         s1.p();
11.         }
12.       }
```

5) this:to passas argumentin theconstructor call

We can pass the this keyword in the constructor also. It is useful if we have to use one object in multiple classes. Let's see the example:

```
1. class B{
2.   A4obj;
3.   B(A4obj){
4.     this.obj=obj;
5.   }
6.   voiddisplay(){
7.     System.out.println(obj.data);//usingdatamemberofA4 class
8.   }
9.}
10.
11.       class A4{
12.         int data=10;
13.         A4(){
14.          Bb=new B(this);
15.          b.display();
16.         }
17.         publicstaticvoidmain(String args[]){
18.          A4a=new A4();
19.         }
20.       }
```

6) This keyword can be used to return currentclass instance

We can return this keyword as an statement from the method. In such case, return type of the method must be the class type (non-primitive). Let's see the example:

Syntax ofthisthatcanbereturnedasastatement

```
1. return_type method_name(){
2. returnthis;
3. }
```

Exampleofthiskeywordthatyoureturnasastatement from the method

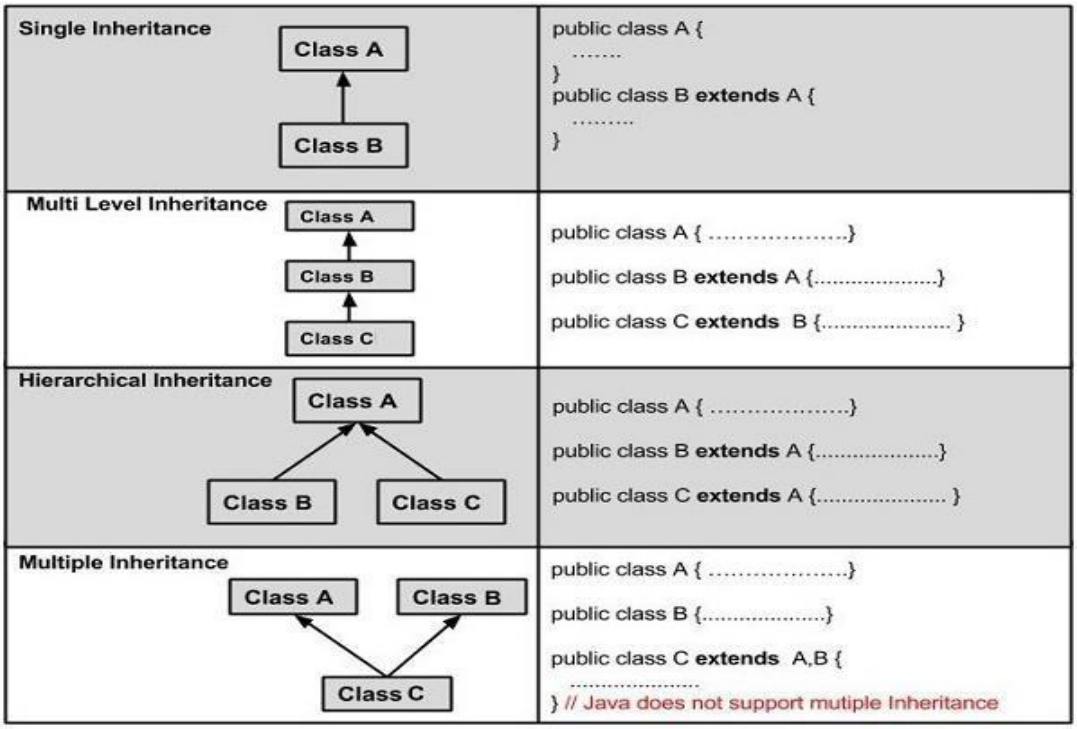
```
1. class A{
2. A getA(){
3. returnthis;
4. }
5. voidmsg(){System.out.println("Hellojava");}
6. }
7. class Test1{
8. publicstaticvoidmain(String args[]){
9. newA().getA().msg();
10.         }
11.         }
```

UNIT-III

Inheritance

Inheritance can be defined as the procedure or mechanism of acquiring all the properties and behavior of one class to another, i.e. acquiring the properties and behavior of child class from the parent class. This concept was built in order to achieve the advantage of creating a new class that gets built upon an already existingclass(es).It is mainly used for code reusability within a Java program. The class that gets inherited taking the properties of another class is the subclass or derived class or child class. Again, the class whose properties get inherited is the superclass or base class or parent class. The keyword **extends** is used to inherit the properties of the base class to derived class. The structure of using this keyword looks something like this:

```
class base
{
   .....
   .....
}
class derive extends base
 {
   .....
   .....
}
```




```
Class
 Person{
 void
 teach(){
  System.out.println("Teachingsubjects");
 }}

Class
 PersonextendsTeacher{
 void listen() {
  System.out.println("Listeningtoteacher");
 }}

Class Check ForInheritance {
 Public static void
main(Stringargs[]){ Person s1 = new
Students();
  s1.teach();
  s1.listen();
 }
}
```

In this type of inheritance, a derived class gets created from another derived class and can have any number of levels.

```
Class
 Teacher{
 void teach()
 {
  System.out.println("Teachingsubject");
 }
}
```

```
Class Student extends
 Teacher{ void listen() {
```

```
  System.out.println("Listening");
 }
}
class homeTution extends Student
 { void explains() {
 System.out.println("Does
 homework");
 }
}
Class Check For Inheritance {
 Public static void main(String
  argu[]){ homeTution h = new
  homeTution();h.explains();
  d.teach();
  d.listen();
 }
}
```

In this type of inheritance, there are more than 1 derived classes which get created from one single base class.

```
Class
 Teacher{
 void teach()
 {
  System.out.println("Teachingsubject");
 }
}
class Student extends
 Teacher { void listen()
 {System.out.println("Listenin
 g");
 }
}
class Principal extends
 Teacher { void evaluate()
 {System.out.println("Evaluati
 ng");
 }
}
Class Check For Inheritance {
 Public static void main(String
  argu[]){ Principal p = new
  Principal();p.evaluate();
  p.teach();
  //p.listen(); willproduce an error
 }
}
```

Let us imagine a situation where there are three classes: A, B and C. The C class inherit s A and B classes. In case, class A and class B have a method with same name and type and as a programmer,you have to

call that

method from child class's (C) object, there-there will be ambiguity as which method will be called either of A or of B class.

So Java reduces this hectic situation by the use of interfaces which implements this concept and reduce this problem; as compile-time errors are tolerable than runtime faults in the program.

Polymorphism

The word polymorphism means having multiple forms. The term Polymorphism gets derived from the Greek word where *poly* + *morphos*where *poly* means many and *morphos* means forms.

Polymorphism is another special feature of object-oriented programming(OOPs). The approach which lies beneath this concept is "single interface with multiple implementations." This offers a single interface for controlling access to a general class of actions.

Polymorphismcanbeachievedintwoofthefollowingways:

- **MethodOverloading**(CompiletimePolymorphism)
- **MethodOverriding**(RuntimePolymorphism)

- Static Polymorphismisinother wordstermedascompile-timebinding or early binding.
- Static binding occurs at compile time. Method overloading is a case of static binding and in this case binding of method call to its definition happens at the time of compilation.

- To call an overloaded method in Java, it is must use the type and/or the number of arguments to determine which version of theoverloaded method to actually call.
- The overloaded methods may have varied return types and the return type single-handedly is insufficient to make out two versions of amethod.
- As and when Java compiler encounters a call to an overloaded method, it simply executes the version of the method whose parameters match the arguments used in the call.
- It permits the user to obtain compile time polymorphism with name method name.
- An overloaded method is able to throw different kinds of exceptions.
- A method which is overloaded can contain different access modifiers.

Overloading method's argument lists might differ in:

- Number of parameters passed
- Datatypeofactual parameters
- Sequenceof datatype of actual parameters

```
class Mltply {
 void mul(int a, int b) {
  System.out.println("Sumoftwo="+(a*b)
  );
 }

 void mul(int a, int b, int c) {
  System.out.println("Sumofthree="+(a*b*c));
 }
}
class Polymorphism {
 public static void
  main(Stringargs[]){ Mltply m = new
  Mltply();
  m.mul(6,10);
  m.mul(10,6,5);
 }}
```

Rules to method overriding

- **Argument list**: The argument list at the time of overriding method need to be same as that of the method of the parent class. The data types of the arguments along with their sequence must have to be preserved as it is in the overriding method.
- **Access Modifier**: The Access Modifier present in the overriding method (method of subclass) cannot be more restrictive than that of an overridden method of the parent class.
- The private, static and final methods can't be overridden as they are local to the class.
- Any method which is overriding is able to throw any unchecked exceptions, in spite of whether the overridden method usually method of parent class might throw an exception or not.

```
//methodoverriding
class parent {
 Public void work(){
  System.out.println("Parentisunderretirementfromwork.");
 }
}
Class child extends parent{
 public void work() {
  System.out.println("Child has a job");
  System.out.println("Heisdoingitwell");
 }
 Public static void
  main(Stringargu[]){ child c1 = new
  child();
  c1.work();
 }}
```

Advantage of method overriding

One major advantage of method overriding is that a class can give its own specific execution to an inherited method without having the modification in the parent class (base class).

Super keyword in java

The **super** keyword in java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

UsageofjavasuperKeyword

1. supercanbe usedto referimmediateparent classinstance variable.
2. supercanbe usedtoinvoke immediateparentclass method.
3. super()canbe usedtoinvoke immediateparentclass constructor.

1) Super is used to refer immediate parent class instance variable.

We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

```
1. class Animal{
2. String color="white";
3. }
4. class Dog extends Animal{
5. String color="black";
6. Void printColor(){
7. System.out.println(color);//printscolorofDog class
8. System.out.println(super.color);//printscolor of Animalclass
9. }
10.        }
11.        class TestSuper1{
12.        public static void main(String args[]){
13.        Dog d=newDog();
14.        d.printColor();
15.        }}
```

2) supercanbeusedtoinvokeparentclassmethod

The super keyword can also be used to invoke parent class method. It should be used if subclass contains the same method as parent class. In other words, it is used if method is overridden.

```
1. class Animal{
2. void eat(){System.out.println("eating...");}
3. }
4. class Dog extends Animal{
5. void eat(){System.out.println("eating bread...");}
6. void bark(){System.out.println("barking...");}
7. void work(){
8. super.eat();
9. bark();
10.         }
11.         }
12.         class TestSuper2{
13.         public static voidmain(String args[]){
14.         Dog d=newDog();
15.         d.work();
16.         }}
```

3) Super is used to invoke parent class constructor.

The super keyword can also be used to invoke the parent class constructor. Let's see a simple example:

```
1. class Animal{
2. Animal(){System.out.println("animalis created");}
3. }
4. class Dog extends Animal{
5. Dog(){
6. super();
7. System.out.println("dog is created");
8. }
9. }
10.         class TestSuper3{
11.         public static void main(String args[]){
12.         Dog d=newDog();
13.         }}
```

```
1. class Person{
2. int id;
3. String name;
4. Person(int id,String name){
5. this.id=id;
```

```
6.  this.name=name;
7.  }
8.  }
9.  Class Empextends Person{
10.         float salary;
11.         Emp(int id,String name,float salary){
12.         super(id,name);//reusingparentconstructor
13.         this.salary=salary;
14.         }
15.         Void display(){System.out.println(id+""+name+""+salary);}
16.         }
17.         class TestSuper5{
18.         public static void main(String[] args){
19.         Emp e1=new Emp(1,"ankit",45000f);
20.         e1.display();
21.         }}
```

Method Overloading inJava

If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

Advantageofmethodoverloading

Method overloading *increases theread ability of the program.*

Different ways to overload the method

There are twoways to overload the method in java

1. By changing number of arguments
2. By changing the data type

1) Method Overloading:changing no.of arguments

In this example, we have created two methods, first add() method performs addition of two numbers and second add method performs addition of three numbers.

In this example, we are creating static methods so that we don't need to create instance for calling methods.

```
1.  class Adder{
2.  static int add(int a,int b){returna+b;}
3.  static int add(int a,int b,int c){returna+b+c;}
4.  }
```

```
5. class TestOverloading1{
6. publicstaticvoidmain(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(11,11,11));
9. }}
```

2) MethodOverloading:changingdatatypeofarguments

In this example, we have created two methods that differs in data type. The first add method receives two integer arguments and second add method receives two double arguments.

```
1. class Adder{
2. static int add(inta,intb){returna+b;}
3. static double add(doublea,doubleb){returna+b;}
4. }
5. class TestOverloading2{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));
8. System.out.println(Adder.add(12.3,12.6));
9. }}
```

Method OverridinginJava

If subclass (child class) has the same method as declared in the parentclass, it is known as **method overriding in java**.

In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

UsageofJavaMethod Overriding

- Methodoverriding is used to provide specific implementation ofa method that is already provided by its super class.
- Method overriding is used for runtime

polymorphism Rules for Java Method Overriding

1. Method must have same name as in the parent class
2. Method must have same parameteras in the parent class.
3. Must be IS-A relationship (inheritance).

```
1. class Vehicle{
2. void run(){System.out.println("Vehicle isrunning");}
3. }
4. Class Bike2 extends Vehicle{
5. Voidrun(){System.out.println("Bikeisrunningsafel
y");}6.
7. Public static void main(String args[]){
8. Bike2obj=new Bike2();
9. obj.run();
10.         }
```

ABSTRACTION

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for examples ending sms,youjusttype the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java

1. Abstractclass (0to 100%)
2. Interface(100%)

Abstractclass in Java

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

Exampleabstractclass

```
1. abstractclass A{}
```

abstract method

A method that is declared as abstract and does not have implementation is

Known as abstract method.

Example abstractmethod

```
1. abstract void printStatus();//nobodyandabstract
```

Exampleofabstractclassthathasabstractmethod

In this example,Bikethe abstractclass that contains only one abstract method run. It implementation is provided by the Honda class.

```
1. abstractclass Bike{
2.   abstract voidrun();
3.}
4. class Honda4extendsBike{
5. void run(){System.out.println("runningsafely..");}
6. public static void main(String args[]){
7.  Bike obj = newHonda4();
8.  obj.run();
9. }
10.        }
```

```
1. Abstract class Shape{
2. Abstract voiddraw();
3. }
4. //In real scenario,implementation is provided by others
    i.e.unknown by end user
5. class Rectangle extends Shape{
6. void draw(){System.out.println("drawing rectangle");}
7. }
8. Class Circle1extends Shape{
9. voiddraw(){System.out.println("drawing circle");}
10.        }
11.        //In real scenario,method is called by programmer or user
12.        class TestAbstraction1{
13.        public static void main(String args[]){
14.        Shape s=newCircle1();//In real scenario,object is provided
   thro ugh method e.g. getShape() method
15.        s.draw();
16.        }
17.        }
```

UNIT-IV

Interfacein Java

An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

Java Interface also **represents IS-**

Arelationship. It cannot be instantiated just

like abstract class.**Why use Java interface?**

There are mainly three reasons to use interface.They are given below.

- It is usedto achieveabstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling

Java InterfaceExample

In this example,Printable interface has onlyonemethod,it simplementation is provided in the A class.

```
1. interface printable{
2. void print();
3. }
4. classA6 implements printable{
5. publicvoidprint(){System.out.println("Hell
o");}.
7. publicstaticvoidmain(String args[]){
8. A6obj= new A6();
9. obj.print();
10.      }
11.     }
```

Differences between abstractclass and interface that are given below.

Abstract class	**Interface**
1) Abstract class can have abstract and non-abstractmethods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
2) Abstract class doesn't support multiple inheritance.	Interface supports multiple inheritance.
3)Abstract class can have final,	Interface has only static and final

non-final,staticandnon-static variables.	variables.
4) Abstract class can provide the implementation of interface.	Interfacecan't providetheimplementation of abstract class.
5)The abstract keyword is used to declare abstract class.	Theinterfacekeywordisusedtodeclareinterface.

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

```
1. interface A{
2. void a();void b();
3. void c();void d();
4. }
5.  abstract class B implements A{
6. public void c(){System.out.println("I amc");}
7. }
8. class M extends B{
9. public void a(){System.out.println("I ama");}
10.        public void b(){System.out.println("I amb");}
11.        public void d(){System.out.println("I amd");}
12.        }
13.        class Test5{
14.        public static void main(String args[]){
15.        A a=new M();
16.        a.a();
17.        a.b();
18.        a.c();
19.        a.d();
20.        }}
```

PACKAGE

A **javapackage** is a group of similar types of classes,interfaces andsub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

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

Here, we will have the detailed learning of creating and using user-definedpackages.

Advantage of Java Package

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provide saccess protection.

3) Java packagere moves naming collision.

Simple example of java package

The **package keyword** is used to create a package injava.

```
1. //save as Simple.java
2. Package mypack;
3. public class Simple{
4.  public static void main(String args[]){
5.     System.out.println("Welcome to package");
6.    }
7.}
```

How to compile java package

If you are not usinganyIDE,you needtofo llowthe **syntax**givenbelow:

```
1. javac-d directory
```

javafilename For **example**

```
1. javac -d . Simple.java
```

The -d switch specifies the destination where to put the generated class file. You can use any directory name like /home (in case of Linux), d:/abc (in case of windows) etc. If you want to keep the package within the same directory, you can use . (dot).

How to access package from anotherpackage?

There are threeways to accessthe package from outside the package.

```
1. import package.*;
2. import package.classname;
3. fullyqualifiedname.
```

1) Using packagename.*

If you use package.* then all the classes and interfaces of this package will be accessible but not subpackages.

The import keyword is used to make the classes and interface of another package accessible to the current package.

Example of package that import the packagename.*

```
1. //saveby A.java
2. packagepack;
3. public class A{
4.   public void msg(){System.out.println("Hello");}
5.}
```

```
1. //saveby B.java
2. Package mypack;
3. Import pack.*;
4.
5. class B{
6.   public static void main(String args[]){
7.    A obj= newA();
8.    obj.msg();
9.   }
10.        }
```

2) Using packagename.classname

If you import package.class name then only declared class of this package will be accessible.

Exampleofpackagebyimport package.classname

```
1. //savebyA.
java2.
3. Package pack;
4. public class A{
5.   public void msg(){System.out.println("Hello");}
6.}
```

```
1. //saveby B.java
2. Package mypack;
3. importpack.A;
4.
5. class B{
6.   public static void main(String args[]){
7.    A obj= newA();
8.    obj.msg();
9.   }
10.        }
```

3) Using ully qualified name

If you use fully qualified name then only declared class of this package will be accessible. Now there is no need to import. But you need to use fully qualified name every time when you are accessing the class or interface.

It is generally used when two packages have same class name e.g. java.util and java.sql packages contain Date class.

Exampleofpackagebyimportfullyqualifiedname

```
1. //saveby A.java
2. packagepack;
3. public class A{
4.   public void msg(){System.out.println("Hello");}
5.}
6.
```

```
1. //saveby B.java
2. Package mypack;
3. class B{
4.   public static void main(String args[]){
5.    pack.Aobj=newpack.A();//usingfullyqualifiedname
6.    obj.msg();
7.   }
8.}
```

Access Modifiers injava

There are two types of modifiers in java: **access modifiers** and **non-accessmodifiers**.

The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.

Thereare4typesofjavaaccess modifiers:

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

Understandingalljavaaccessmodifiers

Access Modifier	within class	within package	outside only	package bysubclass	outside package
Private	Y	N	N		N
Default	Y	Y	N		N
Protected	Y	Y	Y		N
Public	Y	Y	Y		Y

UNIT-V

STRING HANDLING in JAVA

Astring is a sequence o character inJava, widely used as an object.

In java, string is basically an object that represents sequence of char values. An array of characters works same as java string. For example:

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

ch); is same as:

1.String s="javatpoint";

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.

What is String in java

Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.

How to create String object?

There are two ways to create String object:

1. By string literal
2. By new keyword

1) String Literal

Java String literal is created by using double quotes.For Example:

```
1. String s="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";//willnot createnewinstance
```

2) Bynew keyword

```
1. Strings=newString("Welcome");//createstwoobjectsandonerefere
   n ce variable
```

```
1. public class StringExample{
2. public static void main(String args[]){
3. Strings1="java";//creating stringby javastring literal
4. Char ch[]={'s','t','r','i','n','g','s'};
5. Strings2=new String(ch);//converting chararray to string
6. Strings3=newString("example");//creatingjavastringbynewkeywo
   rd
7. System.out.println(s1);
8. System.out.println(s2);
9. System.out.println(s3);
10.        }}
```

Java Stringclass methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1	char charAt(int index)	returnscharvalueforthe particular index
2	int length()	returns string length
3	staticStringformat(Stringformat,Object...args)	returnsformattedstring
4	static String format(Locale l, String format,Object... args)	returns formatted stringwith given locale

5	String substring(int beginIndex)	returnssubstringforgiven begin index
6	String substring(int beginIndex, intendIndex)	returnssubstringforgiven begin index and end index
7	booleancontains(CharSequences)	returns true or false after matching the sequence of char value
8	staticStringjoin(CharSequencedelimiter,CharSequence... elements)	returnsajoinedstring
9	static String join(CharSequence delimiter,Iterable<? extends CharSequence>elements)	returnsajoinedstring
10	booleanequals(Objectanother)	checks the equality of string with object
11	booleanisEmpty()	checks if string is empty
12	String concat(String str)	Concatin atesspecifiedstring
13	String replace(char old, char new)	Replaces alloccurrencesof specified char value
14	String replace(CharSequence old,CharSequence new)	Replaces alloccurrencesof specified CharSequence
15	static String equalsIgnoreCase(Stringanother)	comparesanotherstring.It doesn't check case.
16	String[] split(String regex)	returns splitted stringmatching regex
17	String[] split(String regex, int limit)	returns splitted stringmatching regex and limit
18	String intern()	returnsinternedstring
19	int indexOf(int ch)	returnsspecifiedcharvalueindex
20	int indexOf(int ch, int fromIndex)	returns specified char value index starting with givenindex

21	int indexOf(String substring)	returns specified substring index
22	int indexOf(String substring, intfromIndex)	returns specified substring index starting with givenindex
23	String toLowerCase()	returnsstring in lowercase.
24	String toLowerCase(Locale l)	returnsstringinlowercase using specified locale.
25	String toUpperCase()	returnsstring in uppercase.
26	String toUpperCase(Locale l)	returnsstringinuppercase using specified locale.
27	String trim()	removes beginning andending spaces of this string.
28	static String valueOf(int value)	converts given type intostring. It is overloaded.

```
Public class Sample {

   public static void main(String args[])
      { char[] nameArray = {'A', 'l', 'e',
      'x'};
      Stringname=newString(nameArra
      y);System.out.println(name);
   }}
```

concat() method can be used to attach strings.

```
public class Sample {

   public static void main(String args[])
      {
      Stringstr1="Hello",str2="World!";S
      ystem.out.println(str1.concat(str2)
      );
   }
```

```
}
```

+operator is more commonly used to

attachstrings."Hello," + " world" + "!"

Java ***toUpperCase()*** and***toLowerCase()***method is used to change stringcase.

```
Public class Sample {

    public static void main(String args[])
        { String str1 =
        "Hello";System.out.println(str1.to
        UpperCase());
        System.out.println(str1.toLowerCa
        se());
    }
}
```

Java***trim()***method is usedto eliminate whitespacesbefore and after a string.

```
Public class Sample {

    Public static void main(String
        args[]){ String str = " Hello
        ";System.out.println(str.trim());
    }
}
```

Java ***length()***method is used to

getthelengthofthestring. public class Sample {

```
    Public static void
        main(Stringargs[]){ String str =
        "Cloud";System.out.println(str.
        length());
    }
}
```

JavaStringcompare

We can comparestring injavaonthebasisofcontentand reference.

It is used in **authentication** (by equals() method), **sorting** (by compareTo()method), **reference matching** (by == operator) etc.

There are three ways tocompare string n java:

1. By equals() method
2. By = = operator
3. BycompareTo()method

1) String compare byequals()method

TheStringequals()methodcomparestheoriginalcontentofthestring.It compares values of string for equality. String class provides two methods:

- **Public Boolean equals(Objectanother)**compares this string to the specified object.
- **Public Boolean equalsIgnoreCase(Stringanother)**comparesthis String to another string, ignoring case.

```
1. class Teststringcomparison1{
2.  public static void main(String args[]){
3.    String s1="Sachin";
4.    String s2="Sachin";
5.    Strings3=new String("Sachin");
6.    String s4="Saurav";
7.    System.out.println(s1.equals(s2));//true
8.    System.out.println(s1.equals(s3));//true
9.    System.out.println(s1.equals(s4));//false
10.         }
11.        }
```

2) Stringcompareby==operator

The= =operatorcompares references not values.

```
1. class Teststringcomparison3{
2.  public static void main(String args[]){
3.    String s1="Sachin";
4.    String s2="Sachin";
5.    String s3=new String("Sachin");
6.    System.out.println(s1==s2);//true(becausebothrefertosameinst
   ance)
7.    System.out.println(s1==s3);//false(becauses3referstoinstancec
   re ated in nonpool)
8.  }
9.}
```

3) String compareby compareTo()method

The String compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Supposes1ands2 are two string variables. If:

- **s1==s2**:0
- **s1>s2** :positive value
- **s1<s2** :negative value

```
1. class Teststring comparison4{
2.  public staticvoidmain(String args[]){
3.    String s1="Sachin";
4.    String s2="Sachin";
5.    String s3="Ratan";
6.    System.out.println(s1.compareTo(s2));//0
7.    System.out.println(s1.compareTo(s3));//1(becauses1>s3)
8.    System.out.println(s3.compareTo(s1));//-1(becauses3<s1)
9.  }
10.       }
```

StringConcatenationinJava

In java, string concatenation forms a new string *that is* the combination of multiple strings. There are two ways to concat string in java:

1. By + (string concatenation) operator
2. By concat() method

1) StringConcatenationby+(stringconcatenation)operator

Java string concatenation operator(+)is used to add strings.For Example:

```
1. class TestStringConcatenation1{
2.  publicstaticvoidmain(String args[]){
3.    Strings="Sachin"+"Tendulkar";
4.    System.out.println(s);//SachinTendulkar
5.  }
6.}
```

2) StringConcatenationbyconcat()method

The String concat()method concatenates the specified string totheendof current string. Syntax:

1. public String concat(String another)

Let'ssee the example of String concat() method.

```
1. class TestString Concatenation3{
2.  public static void main(String args[]){
3.    Strings1="Sachin";
4.    String s2="Tendulkar";
5.    String s3=s1.concat(s2);
6.    System.out.println(s3);//SachinTendulkar
7.  }}
```

Substringin Java

A part of string is called **substring**. In other words, substring is a subset of another string. In case of substring startIndex is inclusive and endIndex is exclusive.Note: Index starts from 0.

You can get substring from the given string object by one of the twomethods:

1. **public String substring(int startIndex):** This method returns new String object containing the substring of the given string fromspecified startIndex (inclusive).
2. **public String substring(int startIndex, int endIndex):** This method returns new String object containing the substring of the given string from specified startIndex to endIndex.

In caseofstring:

- **startIndex:**inclusive **endIndex:**exclusive

Let's understand the startIndex and endIndex by the codegiven below.

```
1. String s="hello";
2. System.out.println(s.substring(0,2));//he
```

In the above substring, 0 points to h but 2 points to e (because end index is exclusive).

Example of java substring

```
1. public class TestSubstring{
2.  public static void main(String args[]){
```

```
3.    String s="SachinTendulkar";
4.    System.out.println(s.substring(6));//Tendulkar
5.    System.out.println(s.substring(0,6));//Sachin
6.  }
7.}
```

String Tokenizer in Java

The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string.

It doesn't provide the facility to differentiate numbers, quoted strings, identifiers etc. like StreamTokenizer class. We will discuss about the StreamTokenizer class in I/O chapter.

Constructors of StringTokenizerclass

There are 3 constructors defined in the StringTokenizer

class There are 3constructors defined in the

StringTokenizer class.

Constructor	**Description**
StringTokenizer(Stringstr)	Creates StringTokenizer with specified string.
StringTokenizer(String str, String delim)	Creates StringTokenizerwithspecifiedstring and delimeter.
StringTokenizer(String str, String delim, booleanreturnValue)	creates StringTokenizer with specified string, delimeter and returnValue. If return value is true, delimiter characters are considered to be tokens.Ifitisfalse,delimitercharactersserve toseparate tokens.

Methods of StringTokenizer class

The 6 useful methods of StringTokenizerclassareasfollows:

Publicmethod	**Description**
booleanhasMoreTokens()	checks if there is more tokens available.
String nextToken()	returns the next token from theStringTokenizer object.

String nextToken(String delim)	returnsthenexttokenbasedonthe delimeter.
booleanhasMoreElements()	sameashasMoreTokens() method.
Object nextElement()	sameasnextToken()butitsreturntypeisObject.
int countTokens()	returns the total number of tokens.

```
1. importjava.util.StringTokenizer;
2. public class Simple{
3.  public static void main(String args[]){
4.    StringTokenizerst=newStringTokenizer("mynameiskhan","");
5.     while(st.hasMoreTokens()){
6.         System.out.println(st.nextToken());
7.     }
8.   }
9.}
```

Example of nextToken(String delim) method ofStringTokeni

```
1. importjava.util.*;
2.
3. Public classTest{
4.    Public static void main(String[]args){
5.       StringTokenizerst=newStringTokenizer("my,name,is,kha
n");6.
7.       //printing nexttoken
8.       System.out.println("Nexttokenis:"+ st.nextToken(","));
9.    }
10.       }
```

JAVA I/O

Java I/O(InputandOutput)isused*toprocesstheinput* and*producetheoutput*.

Java uses the concept of stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

We can perform **filehandling in java** byJavaI/OAPI.

Stream

A stream is a sequenceofdata.InJavaastreamiscomposedofbytes.It's called a stream because it is like a stream of water that continues to flow.

In java,3 streams are created for us automatically.All these streams are attached with console.

1) System.out:standardoutput stream

2) System.in:standardinput stream

3) System.err:standarderror stream

	Byte Based		Character Based	
	Input	Output	Input	Output
Basic	InputStream	OutputStream	Reader InputStreamReader	Writer OutputS
Arrays	ByteArrayInputStream	ByteArrayOutputStream	CharArrayReader	CharArr
Files	FileInputStream RandomAccessFile	FileOutputStream RandomAccessFile	FileReader	FileWrite
Pipes	PipedInputStream	PipedOutputStream	PipedReader	PipedWr
Buffering	BufferedInputStream	BufferedOutputStream	BufferedReader	Buffered
Filtering	FilterInputStream	FilterOutputStream	FilterReader	FilterWri
Parsing	PushbackInputStream StreamTokenizer		PushbackReader LineNumberReader	
Strings			StringReader	StringW
Data	DataInputStream	DataOutputStream		
Data – Formatted		PrintStream		PrintWri
Objects	ObjectInputStream	ObjectOutputStream		
Utilities	SequenceInputStream			

Byte Streams Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream. Following is an example which makes use of these two classes to copy an input file into an output file:

```
import   java.io.*;

public        class

CopyFile {

  publicstatic voidmain(String args[])throws IOException

  {
```

```
      FileInputStream in =
      null;
      FileOutputStreamout=nu
      ll; try {
         in=newFileInputStream("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){o
            ut.close();
      }
   }    }
}
```

Character Streams Java Byte streams are used to perform input and output of 8-bit bytes, whereas Java Character streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReader and FileWriter. Though internally FileReader uses FileInputStream andFileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at atime.

```
import java.io.*;
public
classCopyFile{
```

```
public static void main(String args[])throws IOException
```

```
{
   FileReader in =
   null;
   FileWriterout=n
   ull; try {
     in = new
     FileReader("input.txt"); out =
     new FileWriter("output.txt");
     int c;
      while((c=in.read())!=-
         1){out.write(c);
     }
   }finally {
     if(in!=null) {
        in.close();
     }
     if(out != null) {
        out.close();     }    }   }  }


import java.io.*;
public class ReadConsole {
  public static void main(String args[])throws IOException
  {
    InputStreamReadercin=n
    ull; try {
      cin = new InputStreamReader(System.in);
```

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

```
        ch

        ar

        c;

        do

        {

           c = (char)
           cin.read();

           System.out.pri
           nt(c);

        }while(c != 'q');

     }finally{

        if(cin!=null){ci

           n.close();

        }

     }

  }
```

JavaBufferedInputStreamClass

JavaBufferedInputStreamclassisusedtoreadinformationfromstream.It internally uses buffer mechanism to make the performance fast.

```
1.  importjava.io.*;
2.  public class BufferedInputStreamExample{
3.   public static void main(String args[]){
4.    try{
5.      FileInputStreamfin=new FileInputStream("D:\\testout.txt");
6.      BufferedInputStreambin=new BufferedInputStream(fin);
7.      int i;
8.      while((i=bin.read())!=-1){
9.       System.out.print((char)i);
10.            }
```

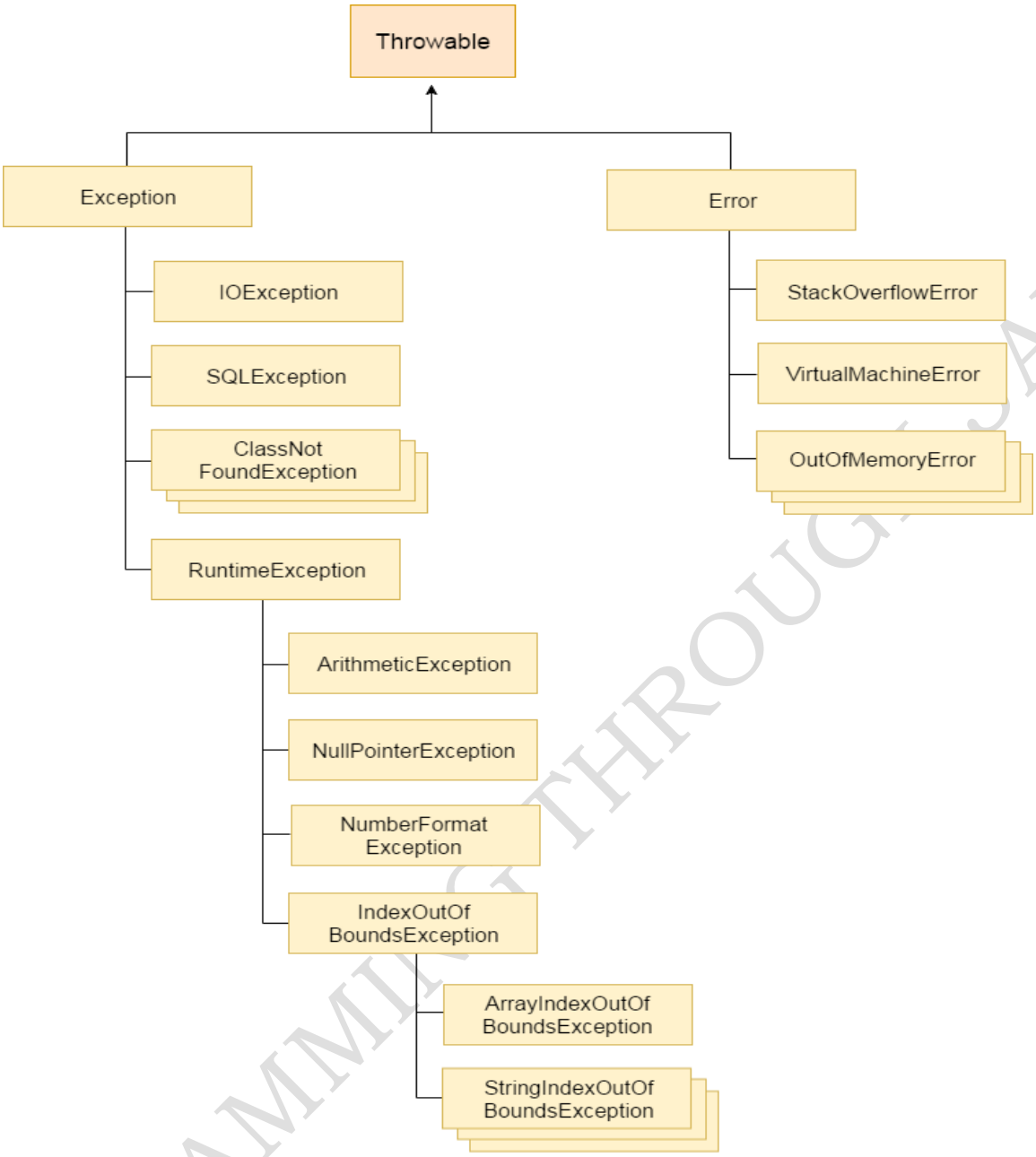
```
11.             bin.close();
12.             fin.close();
13.           }catch(Exceptione){System.out.println(e);}
14.         }
15.        }
```

UNIT VI

EXCEPTION HANDLING INJAVA

The **exception handling in java** is one of the powerful*mechanism to handle the runtime errors* so that normal flow of the application can be maintained. Exception is an abnormal condition. Exception Handling is a mechanism to handle runtime errors

HierarchyofJavaExceptionclasses



Types of Exception

There are mainly two types of exceptions: checked and unchecked where errorisconsideredasuncheckedexception.Thesunmicrosystemsaysthere are three types of exceptions:

1. CheckedException
2. UncheckedException
3. Error

Difference between checked and unchecked exceptions

1) CheckedException

The classes that extend Throwable classexceptRuntimeExceptionandError are known as checked exceptions e.g.IOException, SQLException etc. Checkedexceptionsarecheckedatcompile-time.

2) UncheckedException

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsExceptionetc.Uncheckedexceptionsarenot checked at compile-time rather they are checked at runtime.

3) Error

Errorisirrecoverablee.g.OutOfMemoryError,VirtualMachineErro r, AssertionError etc.

JavaExceptionHandlingKeywords

Thereare5keywordsusedinjavaexception handling.

1. Try catch finally throwthrows

Javatry-

catchJavat

ry block

Java tryblock is used to enclose the code that might throw an exception.It must be used within the method.

Java tryblock must be followed by either catch or finally block.

Syntax of java try-catch

```
1. try{
2. //codethat may throwexception
3. }catch(Exception_class_Nameref){}
```

```
1. public class Testtrycatch2{
2.   publicstaticvoidmain(String args[]){
3.    try{
4.       int data=50/0;
5.    }catch(ArithmeticExceptione){System.out.println(e);}
6.    System.out.println("rest of the code...");
7.}
8.}
```

Java Multicatch block

If you have to perform different tasks at the occurrence of different Exceptions, use java multi catch block.

Let's see a simple xample of javamulti-catch block.

```
1. public class TestMultipleCatchBlock{
2.   public static void main(String args[]){
3.    try{
4.     int a[]=new int[5];
5.     a[5]=30/0;
6.    }
7.    catch(ArithmeticExceptione){System.out.println("task1iscomple
   ted");}
8.    catch(ArrayIndexOutOfBoundsExceptione){System.out.println("task
   2 completed");}
9.    catch(Exceptione){System.out.println("commontaskcomplete
d");}10.
11.           System.out.println("rest of the code...");
12.         }
13.        }
```

Javafinallyblock

Javafinallyblockisablockthatisused*toexecuteimportantcode* suchas closing connection, stream etc.

Java finallyblockisalwaysexecutedwhetherexceptionishandledornot.

Java finally block follows try or catch block.

Whyusejavafinally

- Finallyblockinjavacanbeusedtoput"cleanup"codesuchas closing a file, closing connection etc.

Case1

Let'sseethejavafinallyexamplewhere **exceptiondoesn't occur**.

```
1. class TestFinallyBlock{
2.   public static void main(String args[]){
3.   try{
```

```
4.    int data=25/5;
5.    System.out.println(data);
6.   }
7.   catch(NullPointerExceptione){System.out.println(e);}
8.   finally{System.out.println("finallyblockisalwaysexecuted");}
9.   System.out.println("restof the code...");
10.          }
11.        }
```

Case3

Let'sseethejavafinallyexamplewhere **exceptionoccursandhandled**.

```
1.  public class TestFinallyBlock2{
2.    publicstaticvoidmain(String args[]){
3.    try{
4.     int data=25/0;
5.     System.out.println(data);
6.   }
7.   catch(ArithmeticExceptione){System.out.println(e);}
8.   finally{System.out.println("finallyblockisalwaysexecuted");}
9.   System.out.println("restof the code...");
10.          }
11.        }
```

Javathrowkeyword

TheJavathrowkeywordisusedtoexplicitlythrowanexception.

We can throw either checked or uncheked exception in java by throw keyword.Thethrowkeywordismainlyusedtothrowcustomexception.We will see custom exceptions later.

Thesyntaxofjavathrowkeywordisgiven below.

```
1.  throwexception;
```

```
1.  public class TestThrow1{
2.    static void validate(int age){
3.      if(age<18)
4.       throw newArithmeticException("notvalid");
5.      else
6.       System.out.println("welcome to vote");
7.   }
8.    Public static void main(String args[]){
9.      validate(13);
10.             System.out.println("restof the code...");
```

```
11.          }
12.        }
```

Javathrowskeyword

The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that theremayoccuranexceptionsoitis 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 programmersfaultthatheisnotperformingcheckupbeforethecodebeingus ed.

Syntaxofjava throws

```
1. return_type method_name()throws exception_class_name{
2. //methodcode
3. }
```

```
importjava.
io.*; class
M{
 void method()throws IOException{
  thrownewIOException("deviceerro
  r");
 }  }
publicclass Testthrows2{
   publicstaticvoidmain(Stringargs[
    ]){try{
     Mm=newM();m.
     method();
    }catch(Exceptione){System.out.println("exceptionhandled");}
    System.out.println("normal flow...");
  }  }
```

JavaCustomException

If you are creating you r ownExceptionthatisknownascustomexceptionor user-defined exception. Java custom exceptions are used to customize the exception according to user need.

Let's see asimple example ofjava custom exception.

```
1. classInvalidAgeExceptionextendsException{
2.  InvalidAgeException(Strings){
```

```
3.    super(s);
4.  }
5. }
```

```
1.  class TestCustomException1{
2.
3.     staticvoidvalidate(intage)throwsInvalidAgeException{
4.       if(age<18)
5.        thrownewInvalidAgeException("notvalid");
6.       else
7.        System.out.println("welcometovote");
8.     }
9.
10.            publicstaticvoidmain(String args[]){
11.               try{
12.               validate(13);
13.               }catch(Exceptionm){System.out.println("Exceptionoccu
    red:"+m);}
14.
15.               System.out.println("restof the code...");}}
```


UNIT-VII

MULTITHREADED PROGRAMMING IN JAVA

Multithreading in java is a process of executing multiple threads simultaneously. Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

Advantages of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at same time.

2) You canper form many operations together so it saves time.

3) Threads are independent so it doesn't affect ot therthreads if exception occur in a single thread.

What is Thread in java

A thread is a lightweight sub process, a smallest unit of processing. It is a separate path of execution.

Threads are independent, if there occurs exception in one thread, it doesn't affect other threads. It shares a common memory area.

LifecycleofaThread(ThreadStates)

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

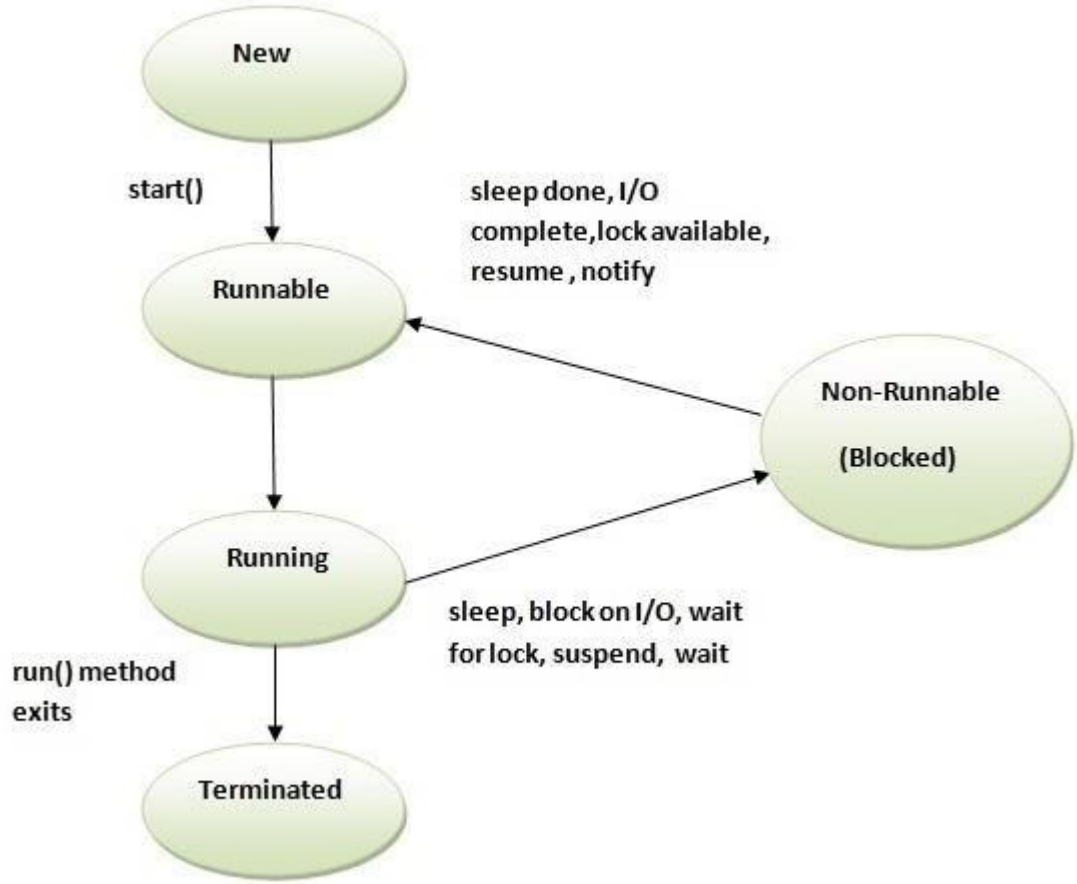
A thread canbe in one of the fivestates.According tosun,thereisonly4 states in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.

But for better understanding the threads,we are explaining it in the 5states.

The lifecycle of the thread injava is controlled byJVM.

The java thread states are as follows:

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



1) New

The thread is innew state if you create an instanceofThreadclassbut before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocationofstart()method,butthe thread scheduler has not selected it to be the running thread.

3) Running

The thread isinrunningstateifthethreadschedulerhasselectedit.

4) Non-Runnable(Blocked)

This is the state when the thread is still alive,butiscurrentlynoteligibletorun.

5) Terminated

A thread is interminated or dead statewhen its run()method exits.

Howtocreate thread

There are two ways to create a thread:

1. By extendingThread class
2. By implementing Runnable interface.

Threadclass:

Thread class provide constructors and methods to create and perform operationsonathread.ThreadclassextendsObjectclassandimplements Runnable interface.

CommonlyusedConstructorsofThreadclass:

- Thread()
- Thread(Stringname)
- Thread(Runnabler)
- Thread(Runnabler,Stringname)

Commonly used methods of Thread class:

publicvoidrun():isusedtoperformactionforathread.

publicvoidstart():startstheexecutionofthethread.JVMcallsthe run() method on the thread.

publicvoidsleep(longmiliseconds): Causesthecurrentlyexecutin g threadtosleep(temporarilyceaseexecution)forthespecifiednumber of milliseconds.

publicvoidjoin():waitsforathreadtodie.

publicvoidjoin(longmiliseconds):waitsforathreadtodieforthe specified miliseconds.

publicintgetPriority(): returnsthepriority ofthe thread.

publicintsetPriority(intpriority):changesthepriorityofthethread .

publicStringgetName():returnsthenameofthethread.

publicvoidsetName(Stringname):changesthenameofthe thread.

publicThreadcurrentThread():returnsthecurrentthread.

publicintgetId():returnsthe idofthe thread.

publicThread.StategetState():returnsthestateofthethread.

publicbooleanisAlive():testsifthethreadisalive.

publicvoidyield():causesthecurrentlyexecutingthreadobjectto temporarily pause and allow other threads to execute.

publicvoidsuspend():isusedtosuspendthe thread(depricated).

publicvoidresume():isusedtoresumethesuspendedthread (depricated).

publicvoidstop():isusedtostopthethread(depricated).

publicbooleanisDaemon():testsifthethreadisadaemon thread.

publicvoidsetDaemon(booleanb):marksthethreadasdaemonor user thread.

publicvoidinterrupt():interruptsthethread.

publicbooleanisInterrupted():testsifthethreadhasbeenin terrupted.

publicstaticbooleaninterrupted():testsifthecurrentthreadhas been interrupted.

Runnableinterface:

The Runnable interface should be implemented by any class whose instancesareintendedtobeexecutedbyathread.Runnableinterfacehave only one method named run().

1. **publicvoidrun():**isusedtoperformactionforathread.

Startingathread:

start()methodofThreadclassisusedtostartanewlycreatedthread.It performs following tasks:

- Anewthreadstarts(withnewcallstack).
- ThethreadmovesfromNew statetotheRunnable state.
- Whenthethreadgetsachancetoexecute,itstargetrun()methodwillru n.

1) JavaThreadExamplebyextendingThreadclass

```
1. class Multi extends Thread{
2. publicvoidrun(){
3. System.out.println("threadisrunning...");
```

```
4. }
5. publicstaticvoidmain(String args[]){
6. Multit1=new Multi();
7. t1.start();
8.  }
9. }
```

2) JavaThreadExamplebyimplementingRunnableinterface

```
1. class Sampleimplements Runnable{
2. publicvoidrun(){
3. System.out.println("Threadisrunning...");
4. }
5.
6. Public static void main(String args[]){
7. Samples=new Sample();
8. Threadt1=newThread(s);
9. t1.start();
10.         }
11.         }
```

Sleepmethod in java

Thesleep()methodofThreadclassisusedtosleepathreadforthespecified amount of time.

Syntaxofsleep()methodinjava

The Threadclassprovidestwomethodsforsleepingathread:

- publicstaticvoid sleep(longmiliseconds)throws InterruptedException
- publicstaticvoidsleep(longmiliseconds,intnanos)throw sInterruptedException

Exampleofsleepmethodinjava

```
1. class ExextendsThread{
2.  public voidrun(){
3.   for(int j=1;j<5;j++){
4.     try{Thread.sleep(500);}catch(InterruptedExceptione){System.o
   ut.println(e);}
5.     System.out.println(j);
```

```
6.   }
7.  }
8.  Public static void main(String args[]){
9.   Ext1=newE
10.x();       Ext2=new Ex();

112.          t1.start();
13.           t2.start();
14.
15.        }}
```

Thejoin() method

The join() method waits for a thread to die. In other words, it causes the currentlyrunningthreadstostopexecutinguntilthethreaditjoinswith completes its task.

Syntax:

Public void join()throwsInterruptedException

Public void join(longmilliseconds)throws InterruptedException

Exampleofjoin() method

```
1.  class Sample extends Thread{
2.   publicvoidrun(){
3.    for(int i=1;i<=5;i++){
4.     try{
5.      Thread.sleep(500);
6.     }catch(Exceptione){System.out.println(e);}
7.    System.out.println(i);
8.   }  }
9.  Public static void main(String args[]){
10.          Samplet1=newSample ();
11.          Samplet2=newSample ();
12.          Samplet3=newSample ();
13.          t1.start();
14.          try{
15.           t1.join();
16.          }catch(Exceptione){System.out.println(e);}
17.          t2.start();
18.          t3.start();
19.          }  }
```

getName(),setName(String)andgetId() method:

public String getName()

publicvoidsetName(Stringname)

public long getId()

```
1.  classSample1extends Thread{
2.    public voidrun(){
3.     System.out.println("running...");
4.    }
5.   Public static void main(String args[]){
6.   Sample1t1=newSample1();
7.    Sample1t2=newSample1();
8.    System.out.println("Nameof t1:"+t1.getName());
9.    System.out.println("Nameof t2:"+t2.getName());
10.          System.out.println("idoft1:"+t1.getId());
11.          t1.start();
12.          t2.start();
13.          t1.setName("SonooJaiswal");
14.          System.out.println("Afterchanging name of t1:"+t1.getName());

15.        }}
```

Priority of a Thread(ThreadPriority):

Each thread have a priority.Priorities are represented by an number between 1 and 10. In most cases, thread schedular schedules the threads according to their priority (known as preemptive scheduling). But it is not guaranteed because it depends on JVM specification that which scheduling it chooses.

3constantsdefinedinThread class:

```
1.  publicstaticint MIN_PRIORITY
2.  public static int NORM_PRIORITY
3.  public static int MAX_PRIORITY
```

Defaultpriorityofathreadis5(NORM_PRIORITY).Thevalueof MIN_PRIORITY is 1 and the value of MAX_PRIORITY is 10.

Example of priority of a Thread:

```
1. classTestMultiPriority1extendsThread{
2.  public voidrun(){
3.    System.out.println("runningthreadnameis:"+Thread.currentThr
   ead().getName());
4.    System.out.println("runningthreadpriorityis:"+Thread.currentT
   hread().getPriority());
5.
6.   }
7.  Public static void main(String args[]){
8.   TestMultiPriority1m1=newTestMultiPriority1();
9.   TestMultiPriority1m2=newTestMultiPriority1();
10.          m1.setPriority(Thread.MIN_PRIORITY);
11.          m2.setPriority(Thread.MAX_PRIORITY);
12.          m1.start();
13.          m2.start();
14.
15.        }
16.       }
```

ThreadGroupinJava

Java provides aconvenientwaytogroupmultiplethreadsinasingleobject. In such way, we can suspend, resume or interrupt group of threads by a single method call.

ConstructorsofThreadGroupclass

There are only two constructorsofThreadGroupclass.

No.	Constructor	Description
1)	ThreadGroup(Stringname)	createsathreadgroupwithgivenname.
2)	ThreadGroup(ThreadGroupparent, String name)	createsathreadgroupwithgiven parent group and name.

ThreadGroupExample

File: ThreadGroupDemo.java

```
1.  Public classThreadGroupDemo implements Runnable{
2.      Public void run()
3.     {
4.           System.out.println(Thread.currentThread().getName());
5.     }
6.     public static void main(String[] args)
7.      {
8.       ThreadGroupDemorunnable =newThreadGroupDemo();
9.           ThreadGrouptg1=newThreadGroup("ParentThreadGroup
");10.
11.                 Threadt1=newThread(tg1, runnable,"one");
12.                 t1.start();
13.                 Threadt2=newThread(tg1, runnable,"two");
14.                 t2.start();
15.                 Threadt3=newThread(tg1, runnable,"three");
16.                 t3.start();
17.
18.                 System.out.println("ThreadGroupName:"+tg1.getName()
    );
19.                tg1.list();
20.
21.            }
22.           }
```

```
1.  classTestMultitasking1extendsThread{
2.   publicvoidrun(){
3.     System.out.println("taskone");
4.   }
5.   Public static voidmain(String args[]){
6.    TestMultitasking1t1=newTestMultitasking1();
7.    TestMultitasking1t2=newTestMultitasking1();
8.    TestMultitasking1t3=newTestMultitaskin
g1();9.
10.          t1.start();
11.          t2.start();
12.          t3.start();
13.         }
14.        }
```

SynchronizationinJava

Synchronization in java is the capability *to control the accessof multiple threads to any shared resource.*

JavaSynchronizationisbetteroptionwherewewanttoallowonlyone thread to access the shared resource.

WhyuseSynchronization

Thesynchronizationismainlyusedto

1. To prevent thread interference.
2. To preventconsistencyproblem.

Types of Synchronization

There are twotypes of synchronization

1. Process Synchronization
2. ThreadSynchronization

ThreadSynchronization

Therearetwotypesofthreadsynchronizationmutualexclusiveandinter-thread communication.

1. Mutual Exclusive
    1. Synchronizedmethod.
    2. Synchronizedblock.
    3. static synchronization.
2. Cooperation(Inter-threadcommunicationinjava)

MutualExclusive

Mutual Exclusive helps keepthreadsfrominterferingwithoneanotherwhile sharing data. This can be done by three ways in java:

1. By synchronizedmethod
2. bysynchronizedblock
3. by static synchronization

ConceptofLockinJava

Synchronization is built around an internal entity known as the lock or monitor.Everyobjecthasanlockassociatedwithit.Byconvention,athread that needs consistent access to an object's fields has to acquire the object's lock before accessing them, and then release the lock when it's done withthem.

From Java5 thepackage java.util.concurrent.lockscontainsseverallockimplementations.

```
1. //exampleofjavasynchronizedmethod
2. class Table{
3.  synchronizedvoidprintTable(intn){//synchronizedmethod
4.    for(int i=1;i<=5;i++){
5.      System.out.println(n*i);
6.      try{
7.       Thread.sleep(400);
8.      }catch(Exceptione){System.out.println(e);}
9.    }
10.
11.         }
12.         }
13.
14.         classMyThread1extends Thread{
15.         Tablet;
16.         MyThread1(Tablet){
17.         this.t=t;
18.         }
19.         publicvoidrun(){
20.         t.printTable(5);
21.         }
22.
23.         }
24.         classMyThread2extends Thread{
25.         Tablet;
26.         MyThread2(Tablet){
27.         this.t=t;
28.         }
29.         publicvoidrun(){
30.         t.printTable(100);
31.         }
32.         }
33.
34.         public class TestSynchronization2{
35.         public static void main(String args[]){
36.         Tableobj= newTable();//onlyone object
37.         MyThread1t1=newMyThread1(obj);
38.         MyThread2t2=newMyThread2(obj);
39.         t1.start();
40.         t2.start();
41.         }
42.         }
```

Inter-threadcommunicationinJava

Inter-threadcommunicationor**Co-operation**isallaboutallowing synchronized threads to communicate with each other.

Cooperation (Inter-thread communication) is a mechanism in which a thread is paused running in its critical section and another thread is allowedtoenter(orlock)inthesamecriticalsectiontobeexecuted.Itis implemented by following methods of **Object class**:

- wait()
- notify()
- notifyAll()

1) wait() method

Causes current thread to release the lock and wait until either another threadinvokesthenotify()methodorthenotifyAll()methodforthisobject, or a specified amount of time has elapsed.

Thecurrentthreadmustownthisobject'smonitor,soitmustbecalledfrom the synchronized method only otherwise it will throw exception.

Method	Description
publicfinalvoidwait()throwsInterruptedException	waits until object is notified.
publicfinalvoidwait(longtimeout)throwsInterruptedException	waitsforthespecified amount of time.

2) notify() method

Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. Thechoiceisarbitraryandoccursatthediscretionoftheimplementation.Syntax:

publicfinalvoidnotify()

3) notifyAll() method

Wakesupallthreadsthatarewaitingonthisobject'smonitor.Syntax: public

final void notifyAll()