SHREE GURU TUITION CLASSES

Java core

[Type the document subtitle]

Deepankar pal

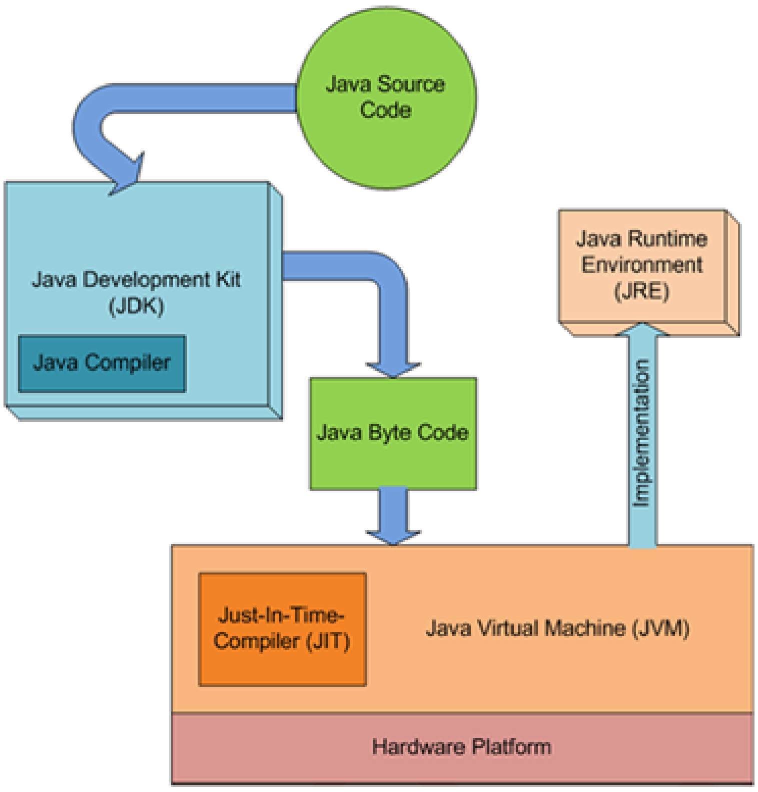
1/1/2020

[Type the abstract of the document here. The abstract is typically a short summary of the contents of the document. Type the abstract of the document here. The abstract is typically a short summary of the contents of the document.]

History

The principles for creating Java programming were "Simple, Robust, Portable, Platform-independent, Secured, High Performance, Multithreaded, Architecture Neutral, Object-Oriented, Interpreted, and Dynamic". Java was developed by James Gosling, who is known as the father of Java, in 1995.

Work of JDK, JVM, JRE


Java Source Code
Java Development Kit (JDK)
Java Compiler
Java Runtime Environment (JRE)
Java Byte Code
Implementation
Just-In-Time-Compiler (JIT)
Java Virtual Machine (JVM)
Hardware Platform

JDK

The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archive (jar), a documentation generator (Javadoc) and other tools needed in Java development.

JVM

It is a specification that provides runtime environment in which java byte code can be executed. ... A specification where working of Java Virtual Machine is specified. ... PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

JRE

The Java Runtime Environment (JRE) is a set of software tools for development of Java applications. It combines the Java Virtual Machine (JVM), platform core classes and supporting libraries. JRE is part of the Java Development Kit (JDK), but can be downloaded separately.

First program of java

```
public class HelloWorld
{
        public static void main(String[] args)
        {
                System.out.println("Hello world");
        }
}
```

Note:- The name of first latter of class always must be capital. Class name and file name must be same.
Out is object of class system and out object call println function on class system.

How can run this program in command prompt:-

First- javac <filename.java>   second- java <class file name>

Features of java

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Dynamic
8. Architecture neutral
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Identifier

In java an identifier is a name given to a variable, class or method.
Identifier start with a letter (A to Z), underscore or dollar sign ($).
It is case sensitive and has no maximum length.
E.g. Username, user_name, _sys_var1, $change.

Variable

A java variable is a piece of memory that can contain a data value.
A variable thus has a data type.

Input and output in java

Input – in java there are three types of classes

1. **Buffer reader class** it is old method and now it is not use.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class Test
{
public static void main(String[] args) throws IOException
{
BufferedReader reader =
new BufferedReader(new InputStreamReader(System.in));
String name = reader.readLine();
System.out.println(name);
}
}
```

2. **Scanner class**

```
import java.util.Scanner;
class GetInputFromUser
{
public static void main(String args[])
{
Scanner in = new Scanner(System.in);
String s = in.nextLine();
System.out.println("You entered string "+s);
int a = in.nextInt();
System.out.println("You entered integer "+a);
float b = in.nextFloat();
System.out.println("You entered float "+b);
}
}
```

3. Console class

```
public class Sample
{
public static void main(String[] args)
{
// Using Console to input data from user
String name = System.console().readLine();
System.out.println(name);
}
}
```

Output-

System .out.println(); or System.out.print(); ln is use for new line,

Keywords

Total 51 keywords in java. 49 are user and (const, goto) is in future.

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			

Operator

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

There are many types of operators in Java which are given below:

- Arithmetic Operator,

Operator	Description	Example
+ (Addition)	Adds values on either side of the operator.	A + B will give 30
- (Subtraction)	Subtracts right-hand operand from left-hand operand.	A - B will give -10
* (Multiplication)	Multiplies values on either side of the operator.	A * B will give 200
/ (Division)	Divides left-hand operand by right-hand operand.	B / A will give 2
% (Modulus)	Divides left-hand operand by right-hand operand and returns remainder.	B % A will give 0
++ (Increment)	Increases the value of operand by 1.	B++ gives 21
-- (Decrement)	Decreases the value of operand by 1.	B-- gives 19

- Relational Operator,

Operator	Description	Example
== (equal to)	Checks if the values of two operands are equal or not, if	(A == B) is

	yes then condition becomes true.	not true.
!= (not equal to)	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
> (greater than)	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
< (less than)	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>= (greater than or equal to)	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<= (less than or equal to)	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

- Bitwise Operator,

Operator	Description	Example
& (bitwise and)	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
| (bitwise or)	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^ (bitwise XOR)	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~ (bitwise compliment)	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a

		signed binary number.
<< (left shift)	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>> (right shift)	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 1111
>>> (zero fill right shift)	Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.	A >>>2 will give 15 which is 0000 1111

- Logical Operator,

Operator	Description	Example
&& (logical and)	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false
|| (logical or)	Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.	(A || B) is true
! (logical not)	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true

- Ternary Operator

```
variable x = (expression) ? value if true : value if false
```

- Assignment Operator.

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand.	C = A + B will assign value of A + B into C
+=	Add AND assignment operator. It adds right operand to the left operand and assign the result to left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts right operand from the left operand and assign the result to left operand.	C -= A is equivalent to C = C – A
*=	Multiply AND assignment operator. It multiplies right operand with the left operand and assign the result to left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides left operand with the right operand and assign the result to left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assign the result to left operand.	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator.	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator.	C >>= 2 is same as C = C

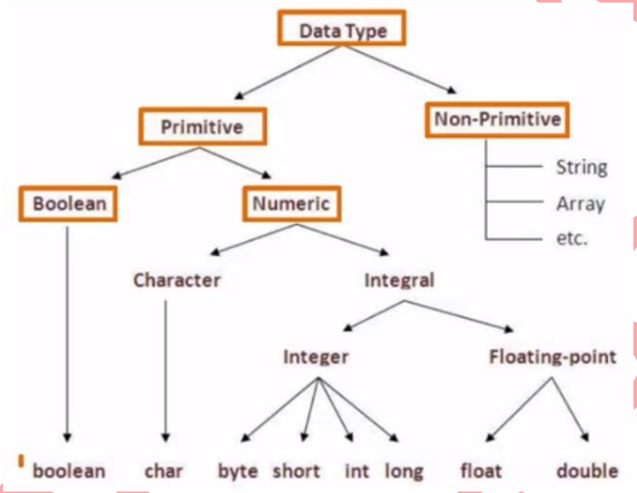
		>> 2
&=	Bitwise AND assignment operator.	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator.	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator.	C |= 2 is same as C = C | 2

Java Operator Precedence

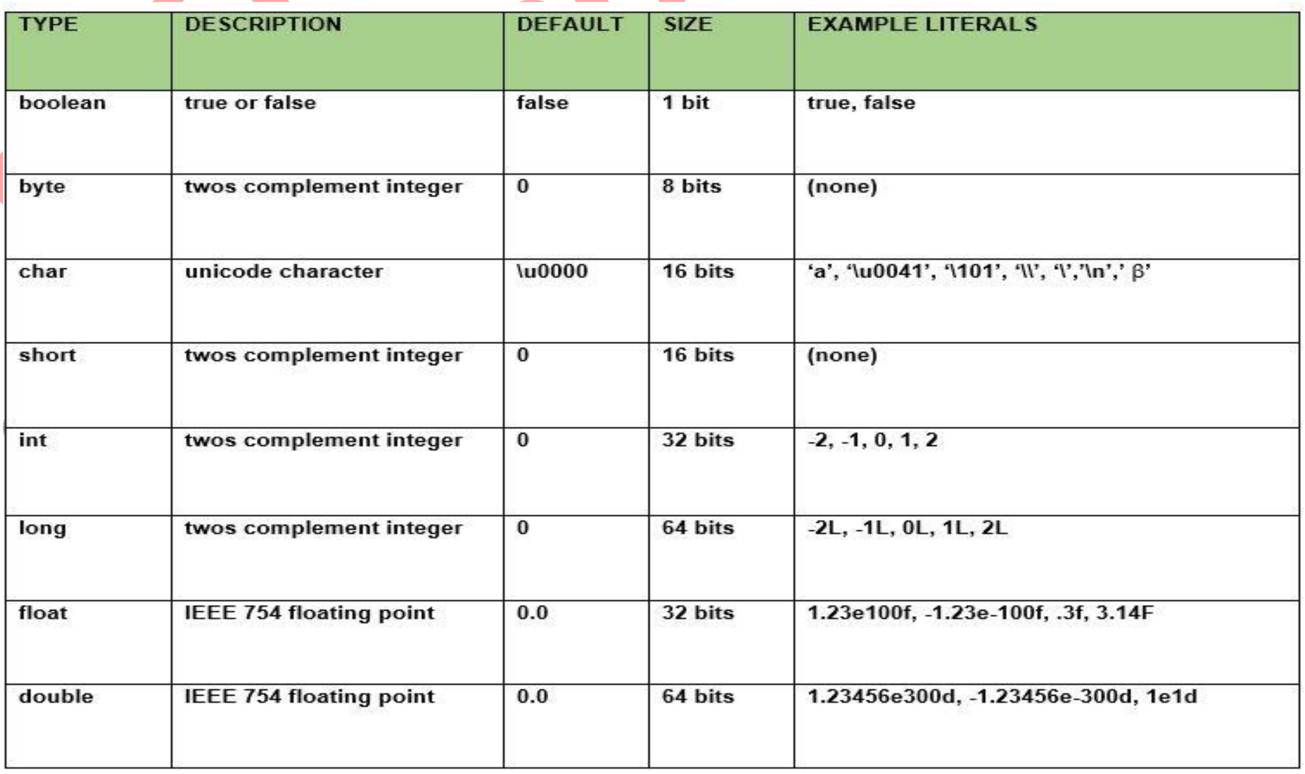
Operator Type	Category	Precedence
Unary	postfix	`expr++ expr--`
	Prefix	`++expr --expr +expr -expr ~ !`
Arithmetic	multiplicative	`* / %`
	additive	`+ -`
Shift	shift	`<< >> >>>`
Relational	comparison	`< > <= >= instanceof`
	equality	`== !=`
Bitwise	bitwise AND	`&`
	bitwise exclusive OR	`^`
	bitwise inclusive OR	`|`

Logical	logical AND	`&&`
	logical OR	`||`
Ternary	ternary	`? :`
Assignment	assignment	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Data types in java



1. Primitive type- there is 8 types.



TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	unicode character	\u0000	16 bits	‘a’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’,’\n’,’ β’
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

2. User-defined types

Type conversion

We want to convert a data type to another data type.

Widening conversion- if in java we want to int value put into float value than no error occurs because int upgrade into float. And in this conversion have no data lost. Widening conversion is valid in java.
Ex. Int y=3;

Float x=y; // widening conversion no error

Boolean -> byte ->char ->short ->int->long ->float -> double

Narrowing conversion- in this conversion have fully data lost and java lets it invalid because java not convert high value to low value. Narrowing conversion is not valid in java.

Ex. Float x=3.4f; f is suffices which is told that this value is float if we not put f then java lets this value to double type.

Int y=x; // Narrowing conversion, error

Int y = (int) x; // no error because here we use forcefully type casting but data may be lost.

Float k = 3.5 //narrowing conversion, error because 3.5 is double vale.

Float k=3.5f // no error

Double -> float -> long ->int ->short ->char ->byte->Boolean

Array in java

Array is a collection of variable of the same data type. First element of the array is stored at 0 indexes.

Advantage of array

1. Code optimization.
2. Random access.

Disadvantage of array

1. Size limit.

Types of array and define

1. One d array
   Syntax:- datatype[size] arrya_name; int[]roll;
   Datatype array_name[size]; int roll[];

2. Multi d array
   Syntax:- datatype [row size][column size] var_name;
   Datatype var_name[][];
   Datatype[]var_name[];

Crating array in java

Var_name=new data_types[size];	int roll_name=new int[5];	one d array
Var_name=new data_types[][];	int roll_name=new int[5][6]	multi d array

```
public class HelloWorld{
    public static void main(String []args){
      int arr[];
      arr=new int[3];
      arr[0]=20;
      arr[1]=36;
      arr[2]=669;
      System.out.println(arr[2]);
    }}
```

```
public class HelloWorld{
    public static void main(String []args){
      int arr[]=new int[3];
      arr[0]=20;
      arr[1]=36;
      arr[2]=669;
      System.out.println(arr[2]);
    }
}
```

program

```
public class HelloWorld{

    public static void main(String []args){
      int arr[],i;
      arr=new int[3];
      arr[0]=45;
      arr[1]=85;
      arr[2]=78;

      for(i=0; i<arr.length;i++)
      {
         System.out.println(arr[i]);
      }
    }
}
```

Comments

There are three way of make comment in java

1. Multiline comment   /*   */

2. Single line comment   //

3. Document comment   /** */ if we want to a code of document then we use this type of comment.

Class and object

Object is instant of class and entity. Real world entity

An object has three characteristics.

1. State- represents data (value) of an object.
2. Behavior-represent the behavior of an object.
3. Identity- it is show unique identity objects.

For e.g.- pen is object then state(white, Reynolds), behavior(writting),

Object creates by new keyword. It is allocate to memory at runtime.

```
Class sum
{
        Int a=10, b=4, ans;
        void add(){
        ans=a+b;
        System.out.println(ans);
        }

        Public static void main(string args[])
        {
                Sum s=new sum();
                s.add();
        }
}
```

Class is blue print of object. Collection of object.

Syntax of class:-

```
Class <class_name>
{
        Data member;
        Method;
}
```

Access Modifier

The access modifier in java specifies accessibility (scope) of a data member, method, and constructer or class. There are three types of java access modifiers. Default access is public in java.

1. Public- it is access inside and outside of the class.
2. Private- it is use inside of class but not access outside class.
3. Protected- it is public for one classes and remaining all classes private.

Constructer and Destructor

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

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

Every time an object is created using the new () keyword, at least one constructor is called.

A constructer is a special method that is used to initialize an object.
A constructer does not have any return type then not write any return type.
A constructer has same name as the class name.
Constructer in java cannot be abstract, static, final or synchronized. And not be virtual.
Java constructer is invoked at the time of object creation.
There are two types of constructer:

1. **Default constructer(no argument constructer)**

   A constructer that has no parameter is known as default constructer.
   Default constructer provides the default values to the object like 0, null etc. depending on the type.

```
public class Con
{
   Con()
   {
      System.out.println("it is default constructer");
   }
   public static void main(String args[])
   {
      Con s=new Con();
   }
}
```

2. **Parameterized constructer(argument constructer)**
   A constructer that has parameter is known as parameterized (argument) constructer.

```
public class Main
{
   int id;
   String name;
   Main(int i, String j)     //parameteried constructer with argument
   {
      id=i;
      name=j;
   }

   void des(){
      System.out.println(id+" "+name);
   }
   public static void main(String args[])
   {
      Main s1=new Main(1,"ram");
      Main s2=new Main(2,"karan");
```

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

Constructer overloading=Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in a parameter list.

The compiler differentiates these constructer by taking into account the number of parameters in the list and their type.

```
Class Over
{
        Int id;
        String name;
        Int age;
        Over()   //default constructer
        {
                System.out.println("hiiiii");
        }
        Over(int i, String n){
                Id=i;
                Name=n;
        }
        Over(int i, String n, int a){
                Id=i;
                Name=n;
                Age=a;
        }
        Void display(){
                System.out.println(id+" "+name+" "+age);
        }
        Public static void main(String args[])
        {
                Over s=new over();
                Over s1=new over(1,"ram");
                Over s2=new over(1,"karan",45);
                s1.display();
                s1.display();
        }
}
Output:-
Hiii
1 ram
1 karan 45
```

Static member in java

In java static keyword is use to crate static variable, static member function, static nested class (static inner class) but we cannot create static variable in method.

```
Public class Example

{
        Int x;      //instance variable
        Static int y;       //static member variable
        Public void fun() {}        //instance member function
        Public static void fun(){}          //static member function

        Static class Test
        {         }
        Public static void main(String args[])
        {}
}
```

Static variable- static variable are declared in the class using static keyword. Static variable are by default initialized to its default value. Static variable has a single copy for the whole class and does not depend on the object.

```
Public class Example

{
        Int x;      //instance variable
        Static int y;                          //static member variable
        Public static void main (String args[])
        {
                Example exa1=new Example ();
                Example exa2=new Example ();
                Example.y=5; or y=5;               //static variable use

        }
}
```

Here we declare two objects so variable x is crate two time for each but variable y is not exist any object Because y is not a variable of object. It is only variable of class. We can use static variable in java to following syntax.

Example.y=5;

Static member function- static function defined inside the class is qualified with the keyword static. Static function can only access static member of the same class. Static function can be invoked using class name and dot operator.

```
Public class Example

{
        Int x;      //instance variable
        Static int y;       //static member variable
        Public void fun1() { }              //instance function
        Public static void fun2() { }       //static function
        Public static void main(String args[])
        {
```

```
                    Example exa1=new Example ();
                    Example exa2=new Example ();
                    Example.fun2();
          }
}
```

Public static void fun2 ()   {x=5 ;} //error

Here if we use x variable in static function then it is occur an error because fun2 () is static function and run without object but x is run with object. So static function only use static variable.

INHERITANCE IN JAVA

Inheritance is an important pillar of OOP (Object Oriented Programming). It is the mechanism in java by which one class is allows inheriting the features (fields and methods) of another class.

Important terminology:

- **Super Class:** The class whose features are inherited is known as super class (or a base class or a parent class).
- **Sub Class:** The class that inherits the other class is known as sub class (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
- **Reusability:** Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

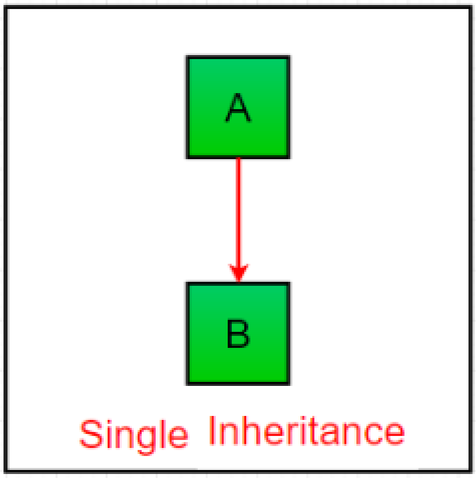
The keyword used for inheritance is **extends**.

Syntax:

```
class derived-class extends base-class

{

   //methods and fields

}
```

Types of Inheritance in Java

1. **Single Inheritance :** In single inheritance, subclasses inherit the features of one superclass. In image below, the class A serves as a base class for the derived class B.

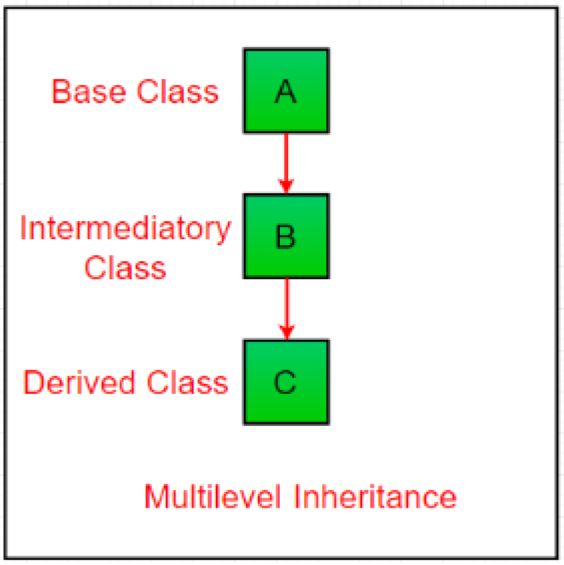


Program-

```
//Java program to illustrate the
// concept of single inheritance
import java.util.*;
import java.lang.*;
import java.io.*;

class one
{
        public void print_geek()
```

```
          {
                    System.out.println("Geeks");
          }
}

class two extends one
{
          public void print_for()
          {
                    System.out.println("for");
          }
}
// Driver class
public class Main
{
          public static void main(String[] args)
          {
                    two g = new two();
                    g.print_geek();
                    g.print_for();
                    g.print_geek();
          }
}
Output:-geek s
          For
          geeks
```

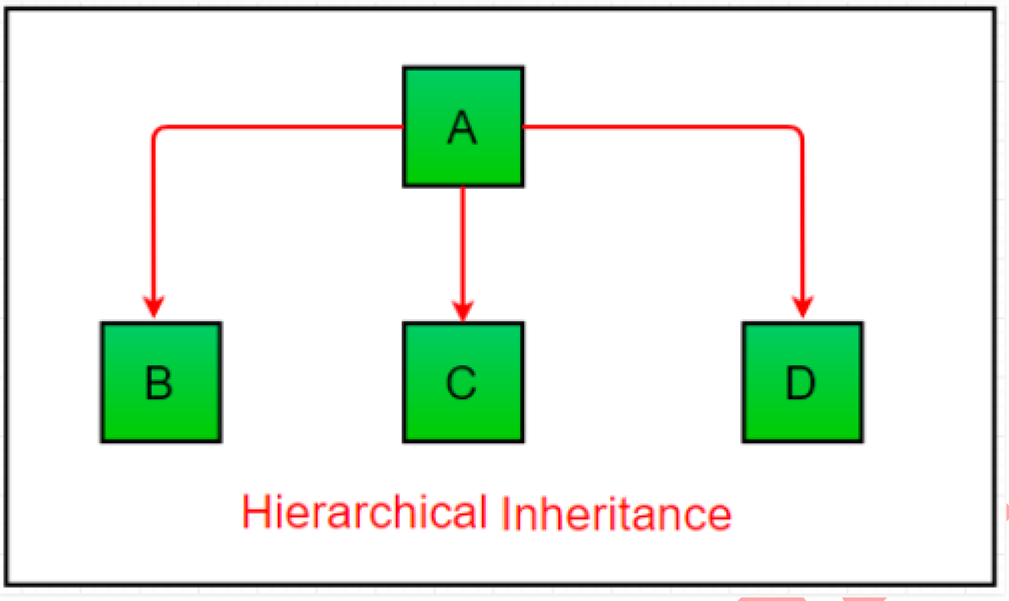
2. **Multilevel Inheritance :** In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In below image, the class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the grandparent's members.



Program-

```
// Java program to illustrate the
// concept of Multilevel inheritance
import java.util.*;
import java.lang.*;
import java.io.*;

class one
{
    public void print_geek()
    {
            System.out.println("Geeks");
    }
}

class two extends one
{
    public void print_for()
    {
            System.out.println("for");
    }
}

class three extends two
{
    public void print_geek()
    {
            System.out.println("Geeks");
    }
}

// Drived class
public class Main
{
    public static void main(String[] args)
    {
            three g = new three();
            g.print_geek();
            g.print_for();
            g.print_geek();
    }
}
```

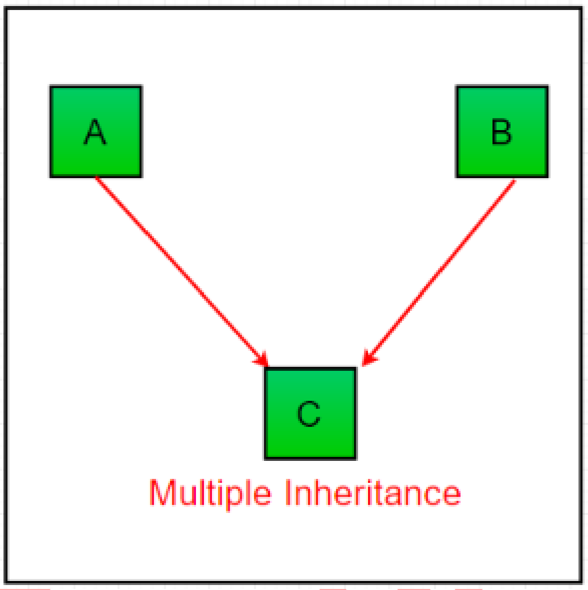
3. **Hierarchical Inheritance:** In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one sub class.In below image, the class A serves as a base class for the derived class B, C and D.



Program-

```
// Java program to illustrate the
// concept of Hierarchical inheritance
import java.util.*;
import java.lang.*;
import java.io.*;

class one
{
    public void print_geek()
    {
            System.out.println("Geeks");
    }
}

class two extends one
{
    public void print_for()
    {
            System.out.println("for");
    }
}

class three extends one
{
    /*............*/
}

// Drived class
public class Main
{
    public static void main(String[] args)
    {
            three g = new three();
            g.print_geek();
            two t = new two();
            t.print_for();
            g.print_geek();
    }
}
```

4. **Multiple Inheritance (Through Interfaces) :** In Multiple inheritance ,one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support multiple inheritance with classes. In java, we can achieve multiple inheritance only through Interfaces. In image below, Class C is derived from interface A and B.

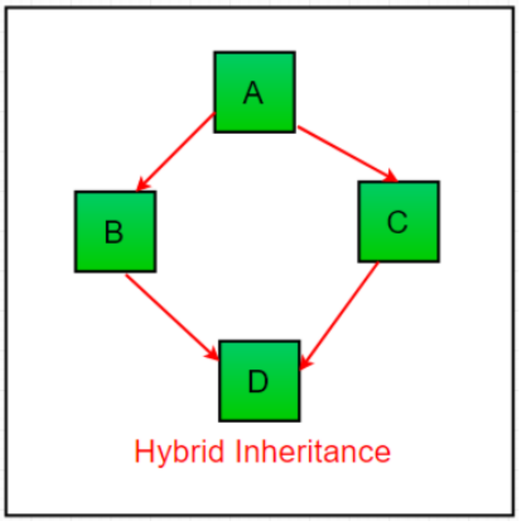


Program-

```
// Java program to illustrate the
// concept of Multiple inheritance
import java.util.*;
import java.lang.*;
import java.io.*;

interface one
{
        public void print_geek();
}

interface two
{
        public void print_for();
}

interface three extends one,two
```

```
{
        public void print_geek();
}
class child implements three
{
        @Override
        public void print_geek() {
                System.out.println("Geeks");
        }

        public void print_for()
        {
                System.out.println("for");
        }
}

// Drived class
public class Main
{
        public static void main(String[] args)
        {
                child c = new child();
                c.print_geek();
                c.print_for();
                c.print_geek();
        }
}
```

5. **Hybrid Inheritance (Through Interfaces):** It is a mix of two or more of the above types of inheritance. Since java doesn't support multiple inheritances with classes, the hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through Interfaces.



This keyword

This is a reference variable that refers to the current object. It can be used inside the method or constructer of class.

```
class Main
{
  int id;
  String name;
  Main(int id, String name)
  {
    this.id=id;
    this.name=name;
  }
 void display()
  {
    System.out.println(id+" "+name);
  }

    public static void main(String[] args)
    {
            Main s1=new Main(1,"ram");
            Main s2=new Main(5,"karan");
            s1.display();
      s2.display();
    }
}
```

Super keyword

The super keyword is used to refer immediate parent class object.
Usage of super keyword- super is used to refer immediate parent class instance variable. Super () is used to invoked immediate parent class constructer. Super is used to invoked immediate parent class constructor. Super is used to invoked immediate parent class method.
To differentiate between base class feature and derived class features must be preceded by super keyword.

Super at variable level: - syntax- *super. Base class member name;*

In order to different between the data member of base class and derived class, in the context of derived class the base data members must be preceded by super keyword.

```
class A
{
  int a=1000;
}
class B extends A
{
  int a=2000;
  void display()
  {
    System.out.println(super.a);    //super variable
  }
}
class Main
{
            public static void main(String[] args)
            {
                        B s=new B();
                        s.display();
            }
}
```

Output: - 1000

Super at method level: - super keyword use when base class method name and derived class method name have same name.

```
class A
{
   void mess()
   {
      System.out.println("hellow");
   }
}
class B extends A
{
   void mess()
   {
      System.out.println("hi");
   }
   void display()
   {
      super.mess();
   }
}
class Main
{
            public static void main(String[] args)
            {
                        B s=new B();
                        s.display();
            }
}
```

Output :- hallow

Polymorphism (method overloading and overgrinding)

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Types of polymorphism

1. Compile time polymorphism- it is handle by compiler
   a. Static polymorphism
      i. method overloading
2. Run time polymorphism- it is handle by JVM
   a. Dynamic polymorphism
      i. Method overriding

Method overloading	Method overriding
Same name	Same name
Same class	Different class
Different argument 1. No. of argument 2. Sequence of argument 3. Type of argument	Same arguments 1. no. of argument 2. sequence of argument 3. type of arguments
	Inheritance concept

Method overloading:-

```
 Class test
{
            Void show()
            {
                        System.out.print("hellow");
            }
            Void show()
            {
                        Syste.out.print("world");
            }
            Public static void main(string args[])
            {
                        Test.s=new test();
                        t.show();
            }
}
```

Here this program compiler will be confused that t.show() what function will be called it is an error called ambiguity error.

Method overriding-

```
Class one
{
            Void show()
            {
                        System.out.println("hellow");
            }
}
Class two extends one
{
            Void show()
```

```
          {
                    System.out.println("world");
          }
          Public static void main(string args[])
          {
                    One.a=new one();
                    a.show();          //hellow
                    Tow.b=new two();
                    b.show();          // two
          }
}
```

Abstract class & method

Abstract class- java abstract classes are used to declare common characteristics of subclasses with abstract keyword. It can only be used as super classes for other classes that extend the abstract classes. Like any other class, an abstract class can contain fields that describe the characteristics and method that describe the action that a class can perform. We cannot create object of abstract class but you can create reference variable of abstract class.

Note- In c++ we can make to abstract class if class contains at list one pure virtual function. There is no abstract keyword in c++.

Syntax:-

```
abstract class A
{
          //its cannot create objects.
}
Class B extends A
{

}
Public class Example
{
          Public static void main(string args[])
          {
                    B obj=new B();          or          A obj=new B();  //refrence variable
          }
}
```

Abstract method- it is same to pure virtual function in c++. A method which is declared as abstract and does not have implementation is known as an abstract method. An abstract class may be consist a method abstract or not but if we can declare an abstract method in a class than it is compulsory to make this class to abstract.

Note if any subclass inherit to any super class which is consist on abstract method than we will do override to abstract method in sub class compulsory. Because if we cannot override abstract method in sub class than no object crate of subclass also.

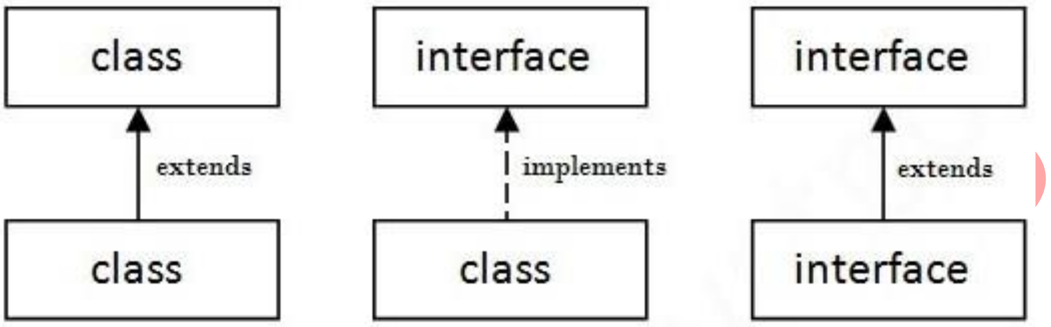
Syntax:-

```
Abstract class A
{
          Abstract void fun();        // abstract method
}
Class B extends b
{
          Void fun()
          {
                code
        }
}
Class Main
{
        Public static void main(string args[])
        {

        }
}
```

Interface in java

An interface like that of an abstract class cannot be instantiated. Interface does not have constructer. We use implement keyword for inherit any interface.

All method of interface default public and abstract And all fields public, static (constant, it is not initialize by object) and final.



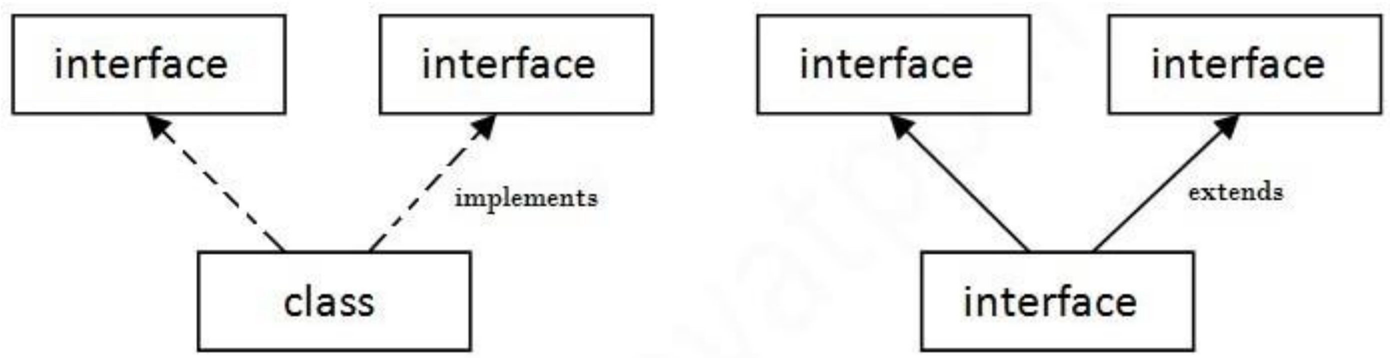
Syntax:-

```
 interface <interface_name>
{
      // declare constant fields
         // declare methods that abstract
          // by default.
}
```

Program-

```
interface printable{
void print();
}

class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
Output- hello
```

Multiple inheritances is possible in java by interface.



Multiple Inheritance in Java

Thread in java

A tread is an independent path of execution within a program. Many threads can run concurrently within a program. Multithreading refters to two or more tasks execution concurrently within a single program. Every thread in java is crated and controlled by the java.lang.Thread class.

There are two ways to created thread in java

1. Implement the Runnable interface (java.lang.Runnable)
2. By extending the Thread class(java .lang.thread)

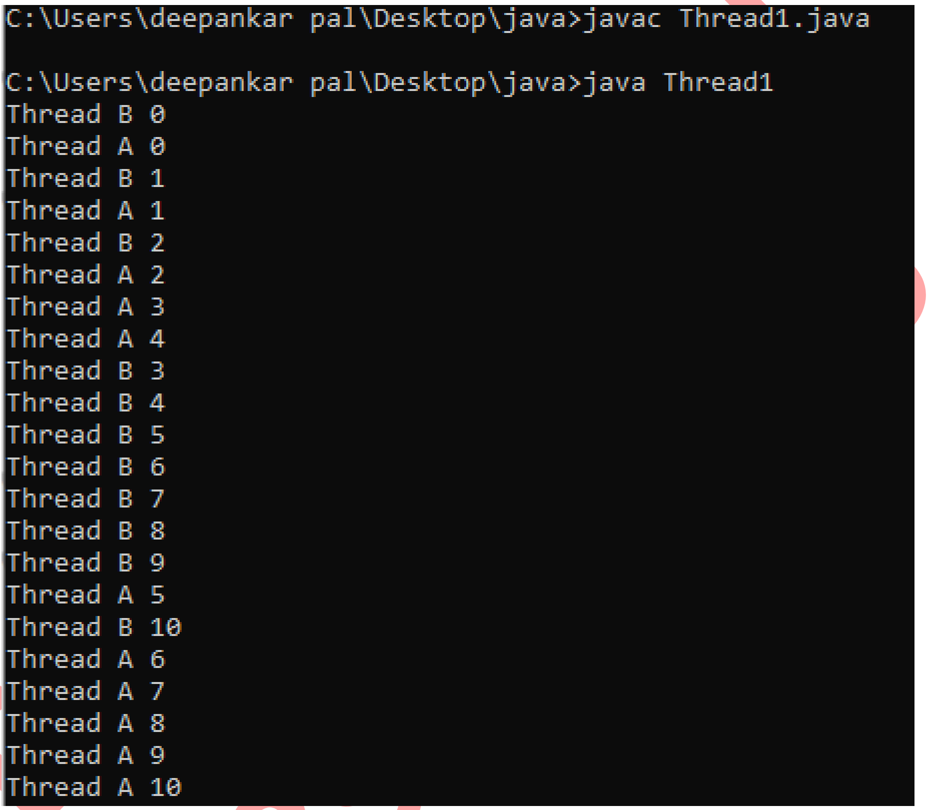
Runnable Interface-

Creating thread -> attach code to the thread -> Executing thread

Program-

```
import java.util.*;
class A implements Runnable
{
   public void run()
   {
      int i;
      for(i=0;i<=10;i++)
      {
         System.out.println("Thread A "+i);
      }
   }
}
class B implements Runnable
{
   public void run()
   {
      int i;
      for(i=0;i<=10;i++)
      {
         System.out.println("Thread B "+i);
      }
   }
}

class Thread1
{
   public static void main(String args[])
   {
     Thread t1=new Thread(new A());
     Thread t2=new Thread(new B());
      t1.start();
      t2.start();
   }
}
```

Output-



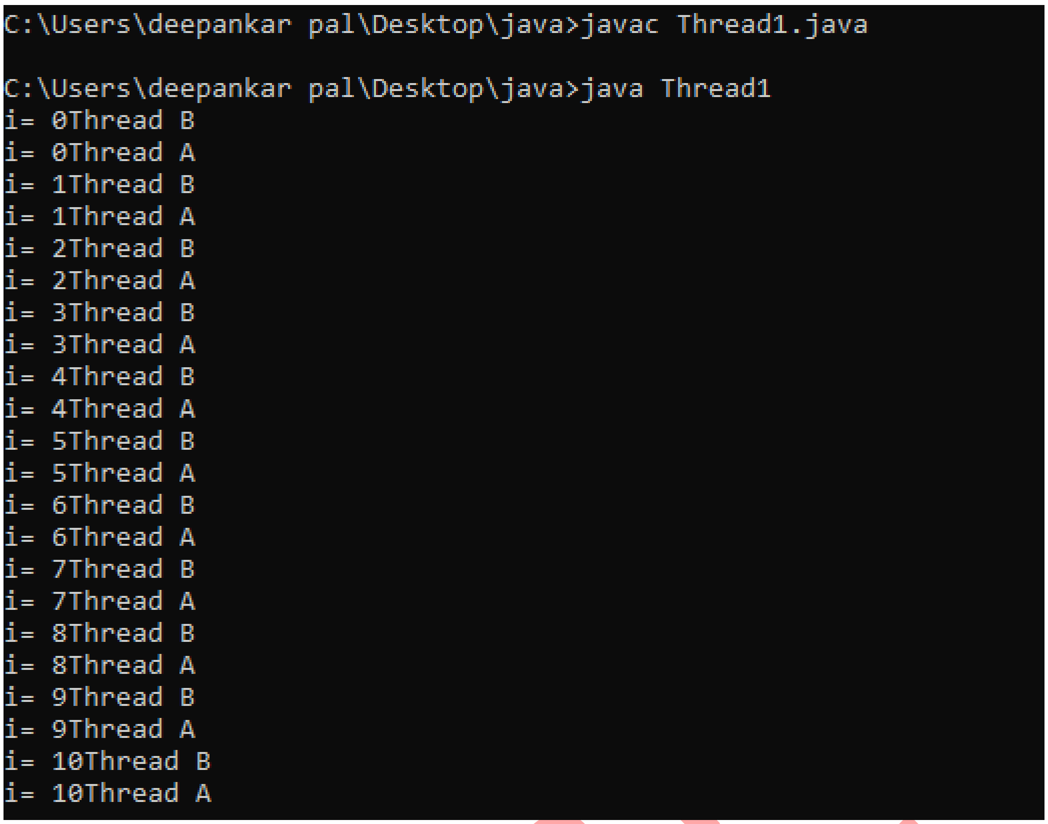
```
C:\Users\deepankar pal\Desktop\java>javac Thread1.java

C:\Users\deepankar pal\Desktop\java>java Thread1
Thread B 0
Thread A 0
Thread B 1
Thread A 1
Thread B 2
Thread A 2
Thread A 3
Thread A 4
Thread B 3
Thread B 4
Thread B 5
Thread B 6
Thread B 7
Thread B 8
Thread B 9
Thread A 5
Thread B 10
Thread A 6
Thread A 7
Thread A 8
Thread A 9
Thread A 10
```

Thread class-

Program-

```
//import java.util.*;
class A extends Thread
{
   public void run()
   {
      int i;
      for(i=0;i<=10;i++)
      {
–         System.out.println("i= "+i+"Thread A");
      }
   }
}
```

```
class B extends Thread
{
   public void run()
   {
      int i;
      for(i=0;i<=10;i++)
      {
         System.out.println("i= "+i+"Thread B");
      }
   }
}

class Thread1
{
   public static void main(String args[])
   {
     A o1=new A();
      B o2=new B();
      o1.start();
      o2.start();
   }
}
```



```
C:\Users\deepankar pal\Desktop\java>javac Thread1.java

C:\Users\deepankar pal\Desktop\java>java Thread1
i= 0Thread B
i= 0Thread A
i= 1Thread B
i= 1Thread A
i= 2Thread B
i= 2Thread A
i= 3Thread B
i= 3Thread A
i= 4Thread B
i= 4Thread A
i= 5Thread B
i= 5Thread A
i= 6Thread B
i= 6Thread A
i= 7Thread B
i= 7Thread A
i= 8Thread B
i= 8Thread A
i= 9Thread B
i= 9Thread A
i= 10Thread B
i= 10Thread A
```

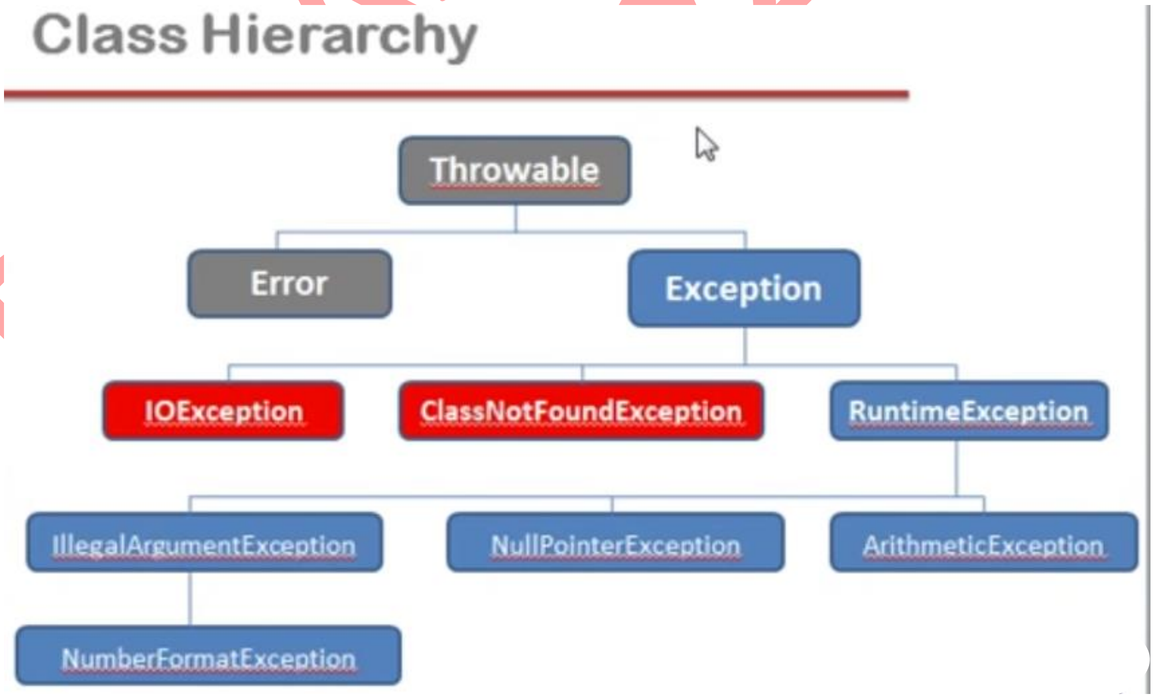
A java thread is always in one of several states which could be running, sleeping, dead, etc.

Sates: - New thread, Runnable, Not runnable, Dead

Exception Handling

Exception in java is any abnormal, unexpected events or extraordinary condition that may occur at runtime.

Java exception handling is used to handle error condition in a program systematically by talking the necessary action.



Java exceptions are raised with the throw keyword and handle within a catch block.

Exceptions can be of two types:

1. checked (compile time exceptions)   2.   Unchecked (run time exception)

Unchecked exception- it is RuntimeException and any of its subclasses.

Eg.   ArrayIndaxOutBunds, NullPointerException and so on are all subclasses of the java.lang.RuntimeException class, which is a subclass of the Exception class.

Exception handling four ways

1. default throw and default catch
2. default throw and our catch
3. our throw and default catch
4. our throw and our catch

```
// default throw and default catch

public class Main

{

        public static void main(String[] args) {

                System.out.println(3/0);

        }

}
```

```
// default throw and our catch

Try
{
}catch(<excetption type><parameter>)
{
}
Finally{
}


public class Main
{
        public static void main(String[] args) {


          try{
            System.out.println(3/0);
            System.out.println("error aate hi ye print nahi hota");
          }
          catch(ArithmeticException e)
          {
            System.out.println("Exception : "+ e.getMessage());
          }
```

```
            finally{
              System.out.println("ye to chal ke hi rahega");
            }
          }
}
```

Throw= a program can explicitly throw an exception using the throw statement beside the implicit exception thrown.

```
//our throw and default catch
public class Main
{
            public static void main(String[] args) {

                int balance=5000;
                int withdraw=6000;
                if(balance< withdraw)
                {
                  throw new ArithmeticException("insufficiant balance");
                }

                balance=balance-withdraw;
                System.out.println("tranjecxtion successfully complte");
                System.out.println("program continue");
            }
}
```

```
//our throw and our catch
public class Main
{
        public static void main(String[] args) {

                int balance=5000;
                int withdraw=6000;
                try{
                if(balance< withdraw)
                {
                  throw new ArithmeticException("insufficiant balance");
                }

                balance=balance-withdraw;
                System.out.println("tranjecxtion successfully complte");
                }
                catch(ArithmeticException e)
                {
                 System.out.println("exception = "+ e.getMessage());
                }
                System.out.println("program continue");
        }
}
```

File handling-

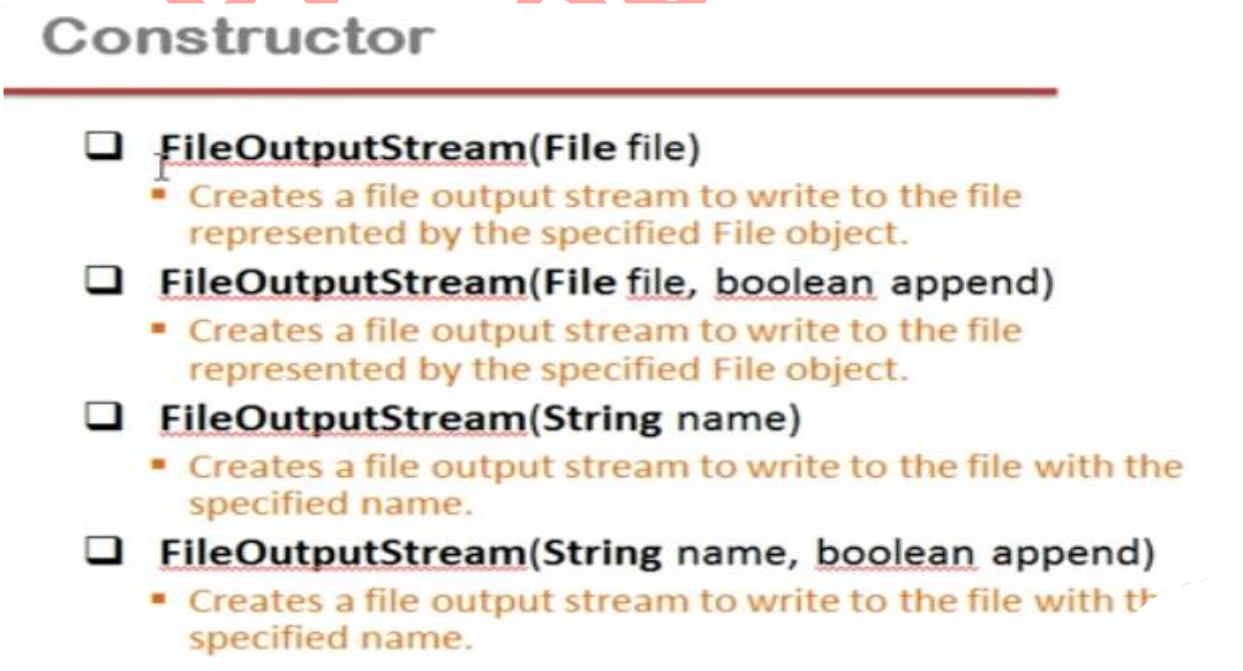
The java.io package contains nearly every class you might ever need to perform input and output (i/o) in java.
All these streams represent an input source and an output destination.
Java provides strong but flexible support for I/O related to files.
File handling is an integral part of nearly all programming projects.
Files provide the means by which a program store data.
Streams represent a source (which generates the data in the form of stream) and a destination (which consumes or read data available as stream)
Stream support a huge range of source and destination including disk file, array, other device, other program etc.
Types of stream
Modern version of java defines two types of stream: byte and character.
.byte stream- provides a convenient means for handling input and output of bytes.
.character streams-are designed for handling the input and output of characters.

- Java I/O (input and output) is use to process the input and produce the output bases on the input.
- 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 file handling in java by java IO API.

Stream- a stream is a sequence of data. In java a stream is composed of bytes. It's called a stream because it's like a stream of water that continues to flow.
Kind of stream

1. **InPutStream:- it is used to read data from a source.**
2. **OutPutStream:- it is used for writing data to a destination.**

Writing to file- writing data to file means storing data in the file.
FileOutputStream is meant for writing streams of raw bytes.
FileOutputStream is subclass of OutputStream.


Constructor

- FileOutputStream(File file)
  - Creates a file output stream to write to the file represented by the specified File object.
- FileOutputStream(File file, boolean append)
  - Creates a file output stream to write to the file represented by the specified File object.
- FileOutputStream(String name)
  - Creates a file output stream to write to the file with the specified name.
- FileOutputStream(String name, boolean append)
  - Creates a file output stream to write to the file with th specified name.

```
//write into data
import java.io.*;
class Ffwrite
{
   public static void main(String args[]) throws IOException
   {
      int i;
      FileOutputStream fout=new FileOutputStream("./pal.txt",true);
      String s="My name is deep";

      char ch[]=s.toCharArray(); //string convert to cahr array
      for(i=0; i<s.length();i++)
      {
         fout.write(ch[i]);
      }
      fout.close();
   }
}
```

```
//read from file
import java.io.*;
class Ffread
{
   public static void main(String args[]) throws IOException
   {
      FileInputStream fin=new FileInputStream("./pal.txt");
      int i=0;
         while((i=fin.read())!=-1){
          System.out.print((char)i);
         }
      fin.close();
   }
}
```

GUI (graphical user interface)

GUI application can be classified in 2 categories.

1. Applet 2. Application

Applets are small program written in java, executed in browser.

Applications are program written in java, executed in desktop.

We can create GUI application in two ways.

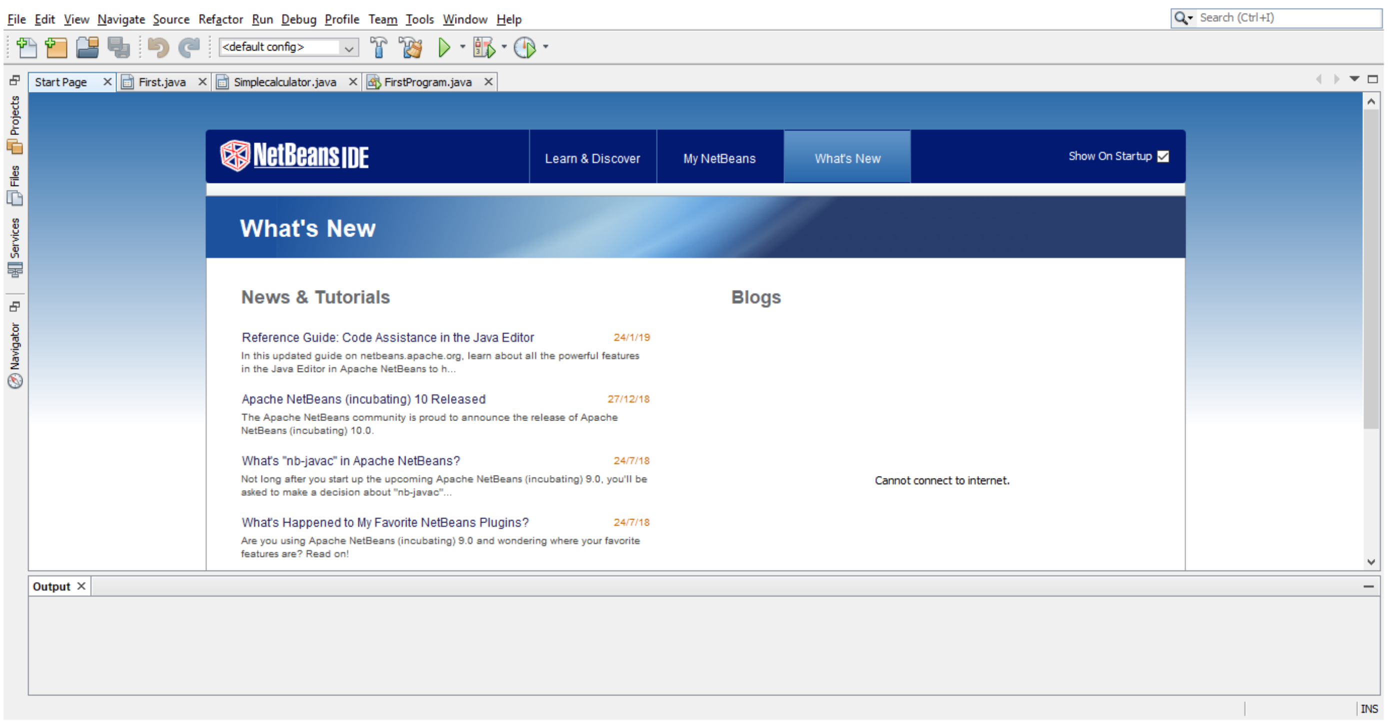
1. AWT(abstract window toolkit) 2. swing

We use IDE (Integrated Development Environment) for crate GUI application in java. For example java net beans, eclipse. A IDE is combination of text editor, java, java, tool, api.

Here we use net beans for GUI

Download link- https://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-3413139-esa.html

After download and installation first screen show



What's New

News & Tutorials

Reference Guide: Code Assistance in the Java Editor 24/1/19
In this updated guide on netbeans.apache.org, learn about all the powerful features in the Java Editor in Apache NetBeans to h...

Apache NetBeans (incubating) 10 Released 27/12/18
The Apache NetBeans community is proud to announce the release of Apache NetBeans (incubating) 10.0.

What's "nb-javac" in Apache NetBeans? 24/7/18
Not long after you start up the upcoming Apache NetBeans (incubating) 9.0, you'll be asked to make a decision about "nb-javac"...

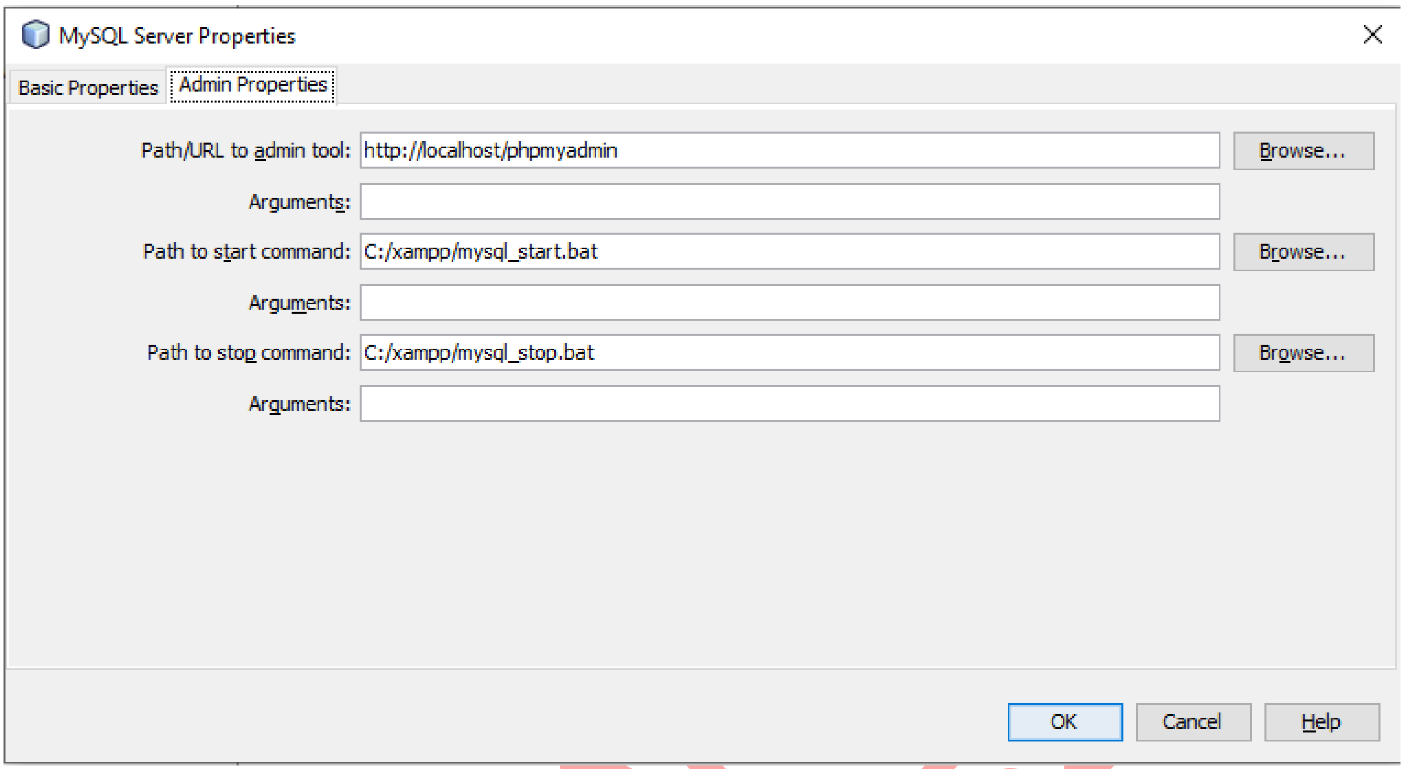
What's Happened to My Favorite NetBeans Plugins? 24/7/18
Are you using Apache NetBeans (incubating) 9.0 and wondering where your favorite features are? Read on!

Blogs

Cannot connect to internet.

Database

Default setting


MySQL Server Properties

Basic Properties | Admin Properties

Path/URL to admin tool: http://localhost/phpmyadmin

Arguments:

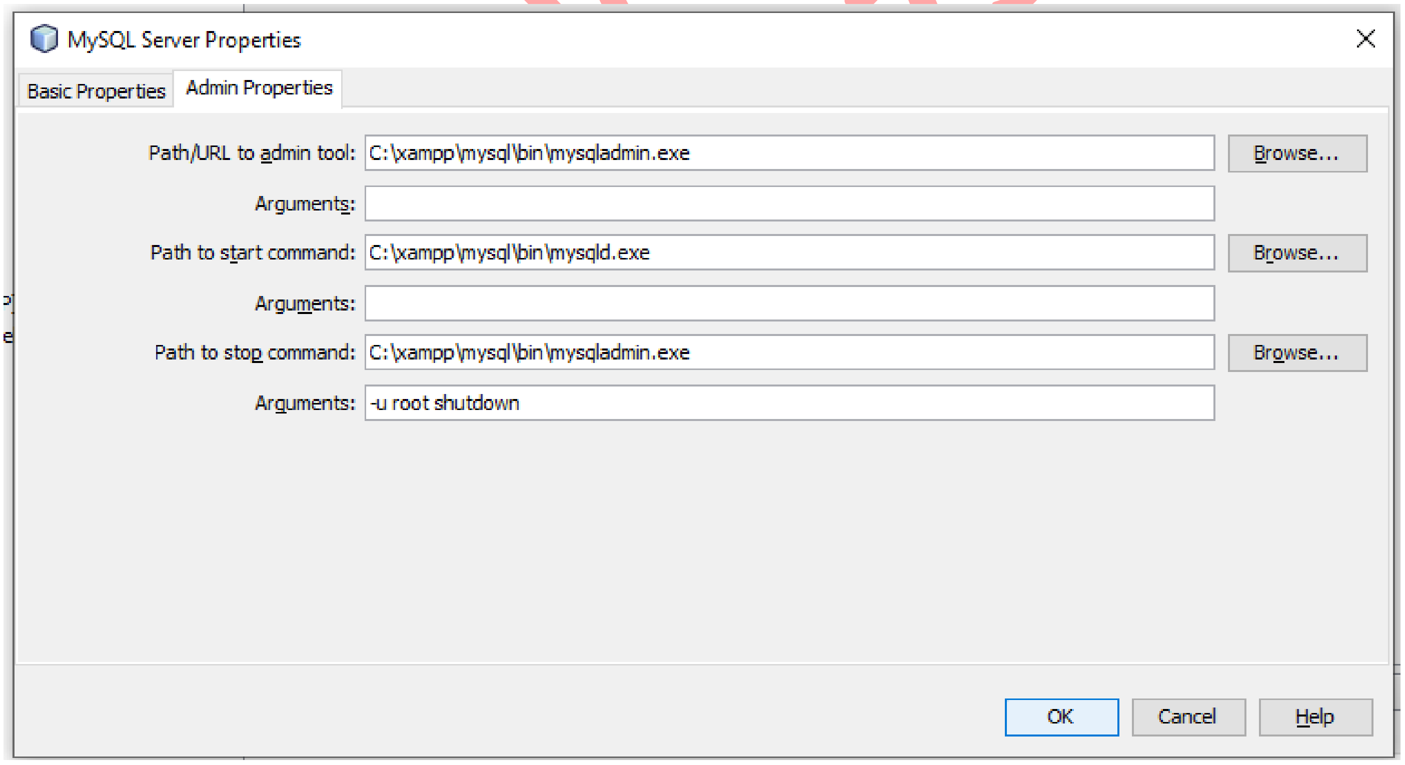
Path to start command: C:/xampp/mysql_start.bat

Arguments:

Path to stop command: C:/xampp/mysql_stop.bat

Arguments:

We set this configuration


MySQL Server Properties

Basic Properties | Admin Properties

Path/URL to admin tool: C:\xampp\mysql\bin\mysqladmin.exe

Arguments:

Path to start command: C:\xampp\mysql\bin\mysqld.exe

Arguments:

Path to stop command: C:\xampp\mysql\bin\mysqladmin.exe

Arguments: -u root shutdown