06/12/2022

TCR INNOVATION

CORE JAVA PROGRAMMING

E-BOOK

Visit Us:

https://tcrinnovation.co.in/

TCR INNOVATION

Navi Mumbai,
Maharashtra
India

INTRO

- Java is an object-oriented, platform-independent programming language.
- High level Robust Java codes are compiled into byte code r machine independent code.
- This byte code is run on JVM (Java Virtual Machine)
- Object oriented – mimic real-life objects

What is Java programming language

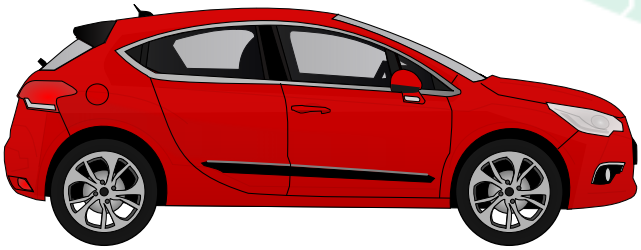
- A programming language used by developers to create various applications
- How hard is this programming language to learn ·
- Where exactly Java is used

What is the goal of this Course

- Understanding What exactly is Java Programming and how it works.
- Learning how to create basic applications with Java.
- Understanding how to work with basic and more advanced Java features

JAVA- OBJECT ORIENTED

- Objects have properties and behaviours
- Object-oriented programming is a method used for designing a program using classes and objects.
- Object-oriented programming is also called the core of java.



PROPERTIES	BEHAVIOURS
• color • horsepower • production year	• start engine • stop engine • lights on • lights off

TOOLS YOU'LL NEED

All you need Development Kit

Complies code

Execute Java applications

Creates runnable Java files

Generates Java documentation

1. JDK (Java Development Kit)

Intelligent text editor for programming

Highlights language keywords

Provides hints

Underlines errors

Version Control

2. IDE (Integrated Development Environment) IntelliJ idea

JDK install for Windows

Install here: Java Development Kit (JDK)
https://www.oracle.com/technetwork/java/javase/downloads/index.html

IntelliJ install for Windows

JetBrains

What is JVM?

JVM (Java Virtual Machine) is an abstract machine that enables your computer to run a Java program. When you run the Java program, Java compiler first compiles your Java code to bytecode. Then, the JVM translates bytecode into native machine code (set of instructions that a computer's CPU executes directly). Java is a platform-independent language. It's because when you write Java code, it's ultimately written for JVM but not your physical machine (computer). Since JVM executes the Java bytecode which is platformindependent, Java is platform-independent.

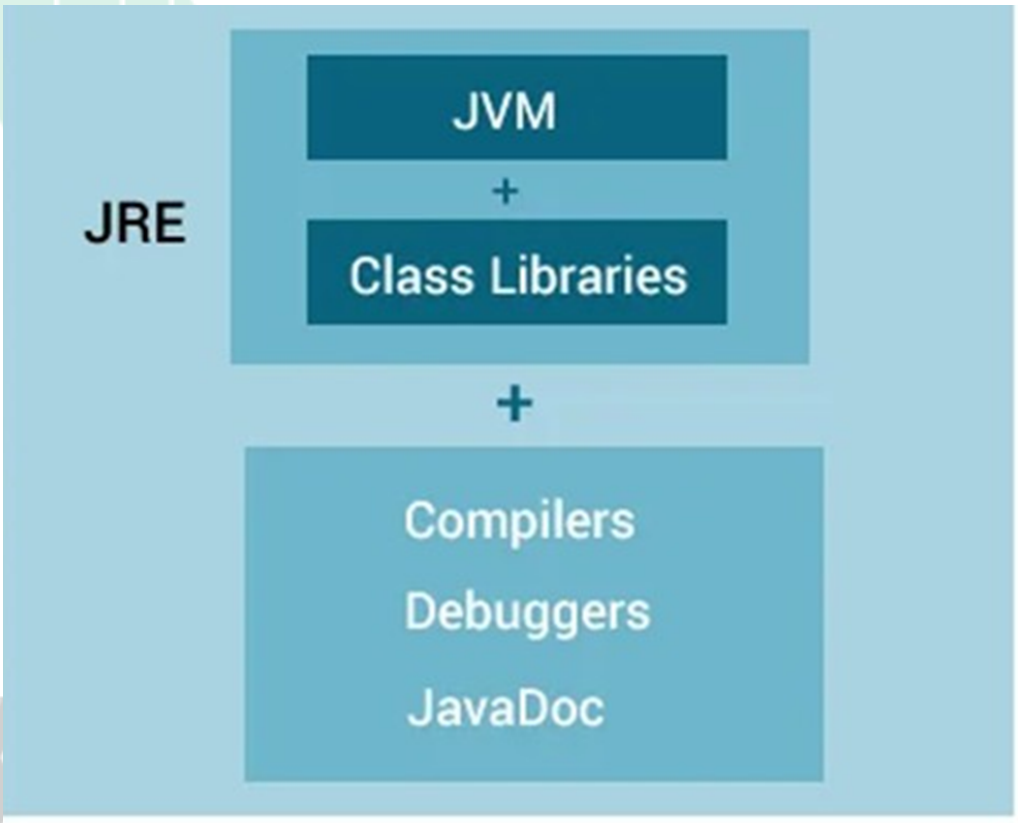
What is JRE?

JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications. JRE is the superset of JVM. If you need to run Java programs, but not develop them, JRE is what you need.

What is JDK?

JDK (Java Development Kit) is a software development kit required to develop applications in Java. When you download JDK, JRE is also downloaded with it. In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger,

Relationship between JVM, JRE, and JDK



TCR Innovation Core Java

Your first Java program

```
class Hello
 { public static void main(String[]
args)
 { System.out.println ("Hello World
program")
```

what above program consists of and its keypoints.

class : class keyword is used to declare classes in Java public : It is an access specifier. Public means this function is visible to all. static : static is again a keyword used to make a function static. To execute a static function you do not have to create an Object of the class. The main() method here is called by JVM, without creating any object for class. void : It is the return type, meaning this function will not return anything. main : main() method is the most important method in a Java program. This is the method which is executed, hence all the logic must be inside the main() method. If a java class is not having a main() method, it causes compilation error. String[] args : This represents an array whose type is String and name is args. We will discuss more about array in Java Array section. System.out.println : This is used to print anything on the console like printf in C language.

Variables in Java

Variable is a container which holds a value Variable assigned with a data type It's name of a memory location. Three types of variables available Local, Instance and static

To declare the variable in Java

datatype variableName; datatype refers to type of variable which can any like: int, float etc. and variableName can be any like: empId, amount, price etc. int age=30 ; double interestRate=8.5d;

Three types of variables available

LOCAL

Local variables are declared in method, constructor or block. Local variables are initialized when method, constructor or block start and will be destroyed once its end. Class not aware of any variable scope inside a method A local variable cannot be defines with “static” keyword.

```
float getDiscount(int
price) { float discount;
discount=price*(20/100);
return discount; }
```

INSTANCE

•Instance variables are variables that are declare inside a class but outside any method, constructor or block.

•Instance variable are also variable of object commonly known as field or property.

•They are referred as object variable. Each object has its own copy of each variable and thus, it doesn't effect the instance variable if one object changes the value of the variable.

```
class Student
{
 String name; int age
; }
```

STATIC

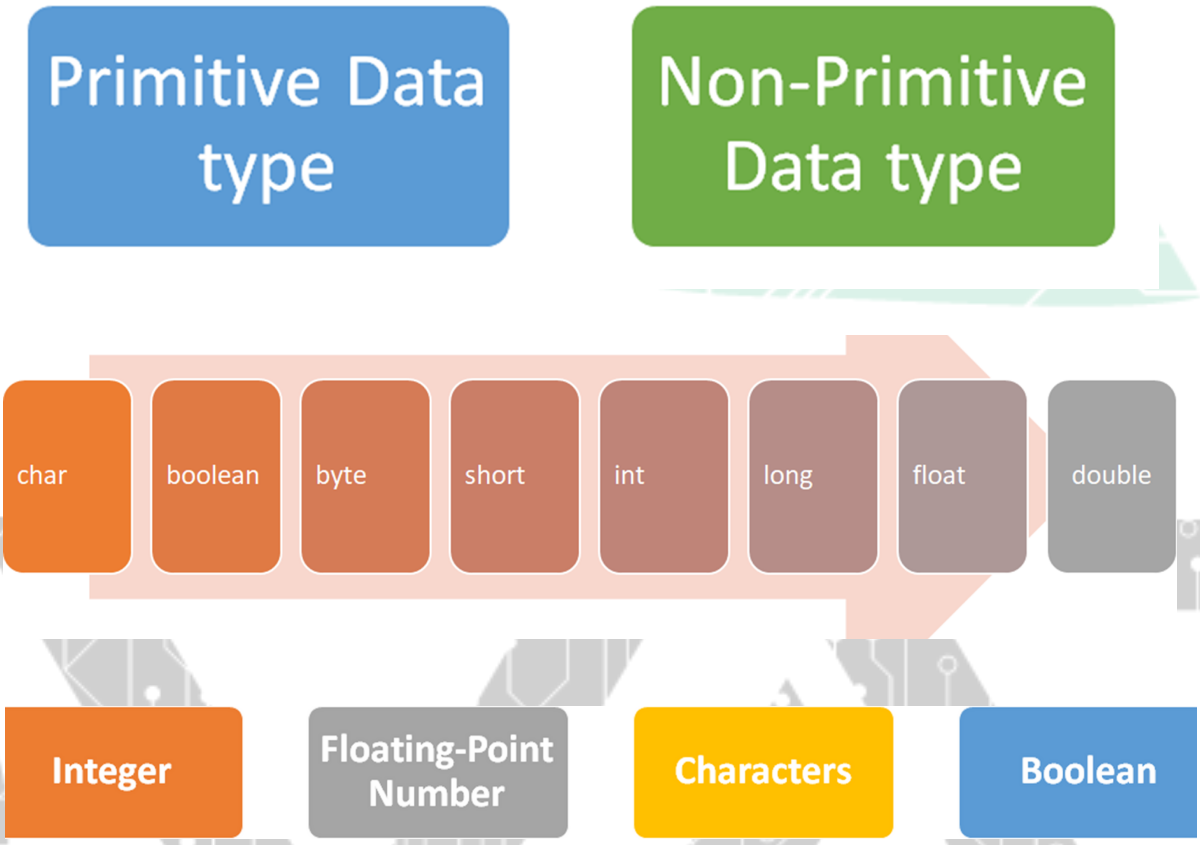
Static are class variables declared with static keyword. Static variables are initialized only once. Create single copy and share with all instances of class.

class Student { String name; int age; static int instituteCode=1101; }

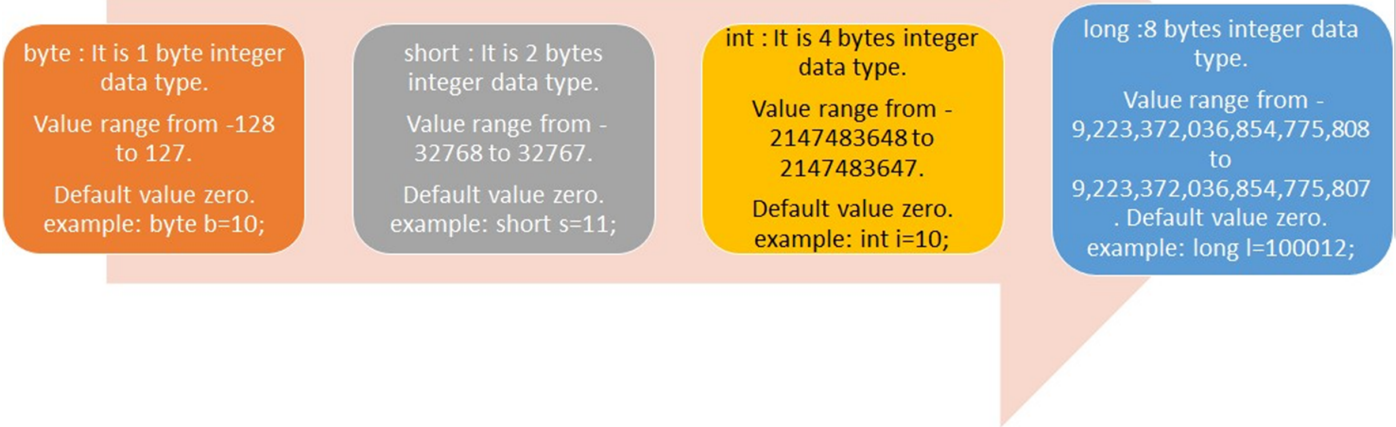
Data Types in Java

•Java language has a rich implementation of data types. Data types specify size and the type of values that can be stored in an identifier.

•In java, data types are classified into two categories :



Integer



Floating-Point Number

Characters

This group represent char, which represent symbols in a character set like letters and numbers. char ch = 'S';

Boolean

•This group represent Boolean, which is a special type for representing true/false values. They are defined constant of the language.

boolean b=true;

Java Operators

.Arithmetic Operators

•Arithmetic operators are used to perform arithmetic operations on variables and data. For example

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operation (Remainder after division)

Assignment Operators

Assignment operators are used in Java to assign values to variables.

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operation (Remainder after division)

Logical Operators

•Logical operators are used to check whether an expression is true or false. They are used in decision making.

&& (Logical AND)	• expression1 && expression2 • true only if both expression1 and expression2 are true
|| (Logical OR)	• expression1 || expression2 • true if either expression1 or expression2 is true
! (Logical NOT)	• !expression • true if expression is false and vice versa

Java Input and Output

•Java Output System.out.println() : It prints string inside the quotes. System.out.print() : It prints string inside the quotes similar like print() method. Then the cursor moves to the beginning of the next line.

Java Input

•Java provides different ways to get input from the user using the object of Scanner class. •In order to use the object of Scanner, we need to import java.util.Scanner package

Java Comments

•comments are a portion of the program that are completely ignored by Java compilers. They are mainly used to help programmers to understand the code. For example,

•

•

```
// declare and initialize two variables
int a =1;
int b = 3;
// print the output
System.out.println("This is output");
```

Types of Comments in Java

- In Java, there are two types of comments:

1.single-line comment
2.multi-line comment

Single-line Comment

- A single-line comment starts and ends in the same line. To write a single-line comment, we can use the // symbol. • // "Hello, World!" program example class Main { public static void main(String[] args) { { // prints "Hello, World!" System.out.println("Hello, World!"); } } Single-line Comment

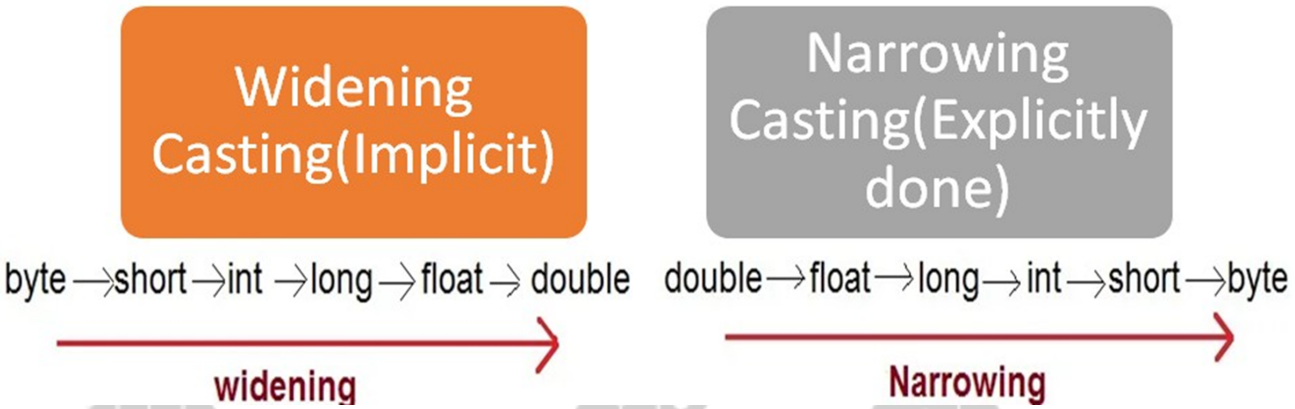
Multi-line Comment

•When we want to write comments in multiple lines, we can use the multi-line comment. To write multi-line comments, we can use the /* .... */ symbol. /* This is an example of multi-line comment. * The program prints "Hello, World!" to the standard output. */ class HelloWorld { public static void main(String[] args) { { System.out.println("Hello, World!"); } } Multi-line Comment

Type Casting in Java

·Casting is a process of changing one type value to another type. In Java, we can cast one type of value to another type. It is known as type casting. · int x = 10; byte y = (byte)x;

In Java, type casting is classified into two types



Widening or Automatic type converion

•Automatic Type casting take place when, •the two types are compatible •the target type is larger than the source type

Widening or Automatic type converion

```
public class Test
{
public static void main(String[] args)
{
int i = 100;
long l = i; //no explicit type casting
required
float f = l; //no explicit type casting
required
System.out.println("Int value "+i);
System.out.println("Long value "+l);
System.out.println("Float value "+f);
}
}
```

Widening or Automatic type converion

Narrowing or Explicit type conversion

When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting. If we don't perform casting then compiler reports compile time error.

Narrowing or Explicit type conversion

public class Test { public static void main(String[] args) { double d = 100.04; long l = (long)d; //explicit type casting required int i = (int)l; //explicit type casting required System.out.println("Double value "+d); System.out.println("Long value "+l); System.out.println("Int value "+i); } } TCR Innovation Co

Java Decision Making

•Java decision-making statements allows you to make
decision, based upon the result of a condition.
•All the programs in Java have set of statements, which
are executed sequentially in the order in which they
appear. This happens when jumping of statements or
repetition of certain calculations is not necessary.

Java supports following decision-making statements:

•if statement •
if-else statement
•else-if statement
•Conditional Operator
•switch statement

if Statements

•In Java, if statement is used for testing the conditions. The condition matches the statement it returns true else it returns false. There are four types of If statement they are:
•There are four types of if statement in Java: i i
. if statement
ii . if-else statement
iii. if-else-if ladder
iv. nested if statement

i. if statement

•The if statement is a single conditional based statement that executes only if the provided condition is true. •If Statement Syntax:
if(condition) { //code } if Statements T
public class Sample{ public static void main(String args[]){ int a=20, b=30; if(b>a) System.out.println("b is greater"); } } if Statements

ii. if-else Statement

•The if-else statement is used for testing condition. If the condition is true, if block executes otherwise else block executes.
•It is useful in the scenario when we want to perform some operation based on the false result.
•The else block execute only when condition is false.
•The if-else statement is used for testing condition. If the condition is true, if block executes otherwise else block executes.
•It is useful in the scenario when we want to perform some operation based on the false result.
•The else block execute only when condition is false.
•The if-else statement is used for testing condition. If the condition is true, if block executes otherwise else block executes.
•It is useful in the scenario when we want to perform some operation based on the false result.
•The else block execute only when condition is false.

```
if(condition)
 {
//code for true
 }
else
{ //code for fals
 public class Sample {
 public static void main(String args[]) {
 int a = 80, b = 30;
 if (b > a) {
 System.out.println("b is greater");
  } else {
 System.out.println("a is greater");
  }
 public class IfElseDemo1 {                } }
  public static void main(String[] args)
{ int marks=50; i
f(marks > 65)
 { System.out.print("First division"); } else {
System.out.print("Second division"); } } }
```

if-else-if ladder Statement

•In Java, the if-else-if ladder statement is used for testing conditions. It is used for testing one conditi from multiple statements.

•When we have multiple conditions to execute the it is recommend to use if-else-if ladder.

```
if(condition1) { //code for if
condition1 is true } else
if(condition2) { //code for if
condition2 is true } else
if(condition3) { //code for if
condition3 is true } ... else {
//code for all the false conditions
} Tif(condition1) { //code for if
condition1 is true } else
if(condition2) { //code for if
condition2 is true } else
if(condition3) { //code for if
condition3 is true } ... else {
//code for all the false conditions
}
```

Add a little bit of body text

Add a little bit of body text

iii. Nested if statement

•In Java, the Nested if statement is a if inside another if. In this, one if block is created inside another if block when the outer block is true then only the inner block is executed.

```
 if(condition)
 {
//statement if(condition)
 { //statement }
 } Nested if statement public class NestedIfDemo1
{ public static void main(String[] args)
 { int age=25;
int weight=70;
if(age>=18) {
 if(weight>50) { System.out.println("You are
eligible");
} }
 } } }
```

Java switch Statements

•The Java switch statement executes one statement from multiple conditions. It is like if-else-if ladder statement.

```
 switch(variable)
{ case 1:
//execute your code break;
 case n:
//execute your code break;
default:
//execute your code break; } TCR Innovation Core Java
Java switch Statements public class Sample
 { public static void main(String args[])
 { int a = 5;
 switch (a) { case 1: System.out.println("You chose One");
break;
case 2: System.out.println("You chose Two"); break;
case 3: System.out.println("You chose Three"); break;
case 4: System.out.println("You chose Four"); break;
case 5: System.out.println("You chose Five"); break;
default: System.out.println("Invalid Choice. Enter a no
between 1 and 5"); break; } } } T
```

What is Loop?

•A computer is the most suitable machine to perform repetitive tasks and can tirelessly do a task tens of thousands of times. Every programming language has the feature to instruct to do such repetitive tasks with the help of certain form of statements. The process of repeatedly executing a collection of statement is called looping. The statements gets executed many number of times based on the condition. But if the condition is given in such a logic that the repetition continues any number of times with no fixed condition to stop looping those statements, then this type of looping is called infinite looping.

Java Loops

•Sometimes it is necessary in the program to execute the statement several times, and Java loops execute a block of commands a specified number of times

Java supports following types of loops:

while loops

•do while loops

•for loops

• while loops

•Java while loops statement allows to repeatedly run the same block of code, until a condition is met. •while loop is most basic loop in Java. It has one control condition, and executes as long the condition is true. The condition of the loop is tested before the body of the loop executed, hence it is called an entry-controlled loop.

The basic format of while loop statement is: while (condition) { statement(s); Incrementation; }

Java do while loops

•Java do while loops is very similar to the while loops, but it always executes the code block at least once and further more as long as the condition remains true. This is exit-controlled loop

```
 do
 {
statement(s)
; }while( condition );
```

Java for loops

- Java for loops is very similar to Java while loops in that it continues to process a block of code until a statement becomes false, and everything is defined in a single line.

```
 for ( init; condition; increment )
{ statement(s);
 }  public class Sample
{ public static void main(String args[])
 { int n = 1
; for (n = 1; n <= 5; n++)
{ System.out.println("Java for loops:" + n)
; }
 }
 }
```

Java Break Statement

•When a break statement is encountered inside a loop, the loop is immediately terminated and the program control resumes at the next statement following the loop.

•The Java break statement is used to break loop or switch statement. It breaks the current flow of the program at specified condition.

```
 public class BreakExample { public
static void main(String[] args)
 { //using for loop
 for(int i=1;i<=10;i++){ if(i==5)
{ //breaking the loop break; }
System.out.println(i); } } }
```

Java Continue Statement

·The continue statement is used in loop control structure when you need to jump to the next iteration of the loop immediately. It can be used with for loop or while loop. ·The Java continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition. In case of an inner loop, it continues the inner loop only.

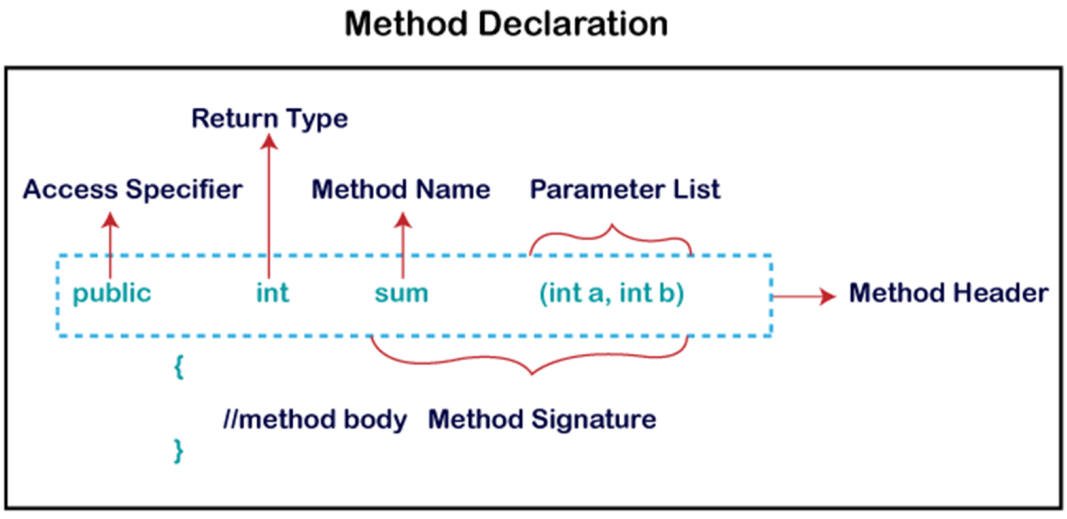
```
 public class ContinueExample
{ public static void main(String[] args)
 { //for loop for(int i=1;i<=10;i++){ if(i==5){
//using continue statement continue;//
it will skip the rest statement }
System.out.println(i); } } }
```

Method in Java

•A method is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the reusability of code. We write a method once and use it many times. We do not require to write code again and again. It also provides the easy modification and readability of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

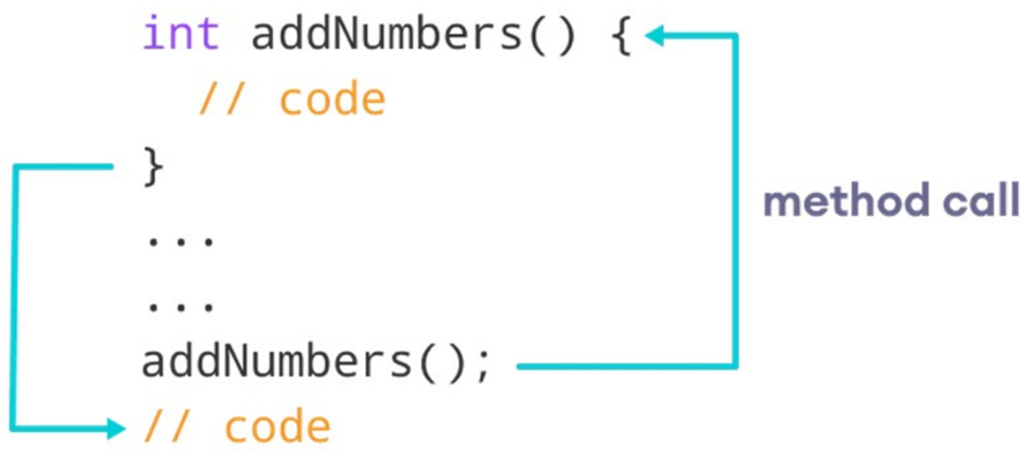
Method Declaration

•The method declaration provides information about method attributes, such as visibility, return-type, name, and arguments. It has six components that are known as method header, as we have shown in the following figure



•Method Signature: Every method has a method signature. It is a part of the method declaration. It includes the method name and parameter list.
•Access Specifier: Access specifier or modifier is the access type of the method. It specifies the visibility of the method. Java provides four types of access specifier: •Public: The method is accessible by all classes when we use public specifier in our application.
•Private: When we use a private access specifier, the method is accessible only in the classes in which it is defined.
•Protected: When we use protected access specifier, the method is accessible within the same package or subclasses in a different package.
•Default: When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only

Calling a Method in Java



Java Method Return Type

•A Java method may or may not return a value to the function call. We use the return statement to return any value. For example

Static Method

•A method that has static keyword is known as static method. In other words, a method that belongs to a class rather than an instance of a class is known as a static method. Create a static method by using the keyword static before the method name.

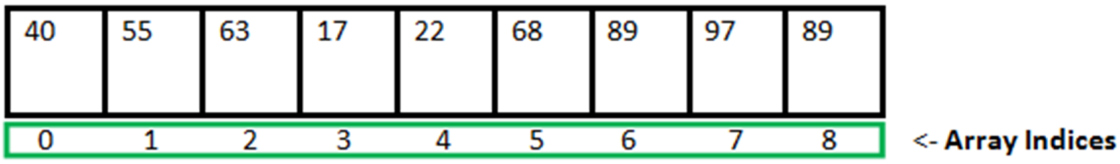
•The main advantage of a static method is that we can call it without creating an object. It can access static data members and also change the value of it. It is invoked by using the class name. The best example of a static method is the main() method.

Instance Method

•The method of the class is known as an instance method. It is a non-static method defined in the class. Before calling or invoking the instance method, it is necessary to create an object of its class. Let's see an example of an instance method.

Java Arrays

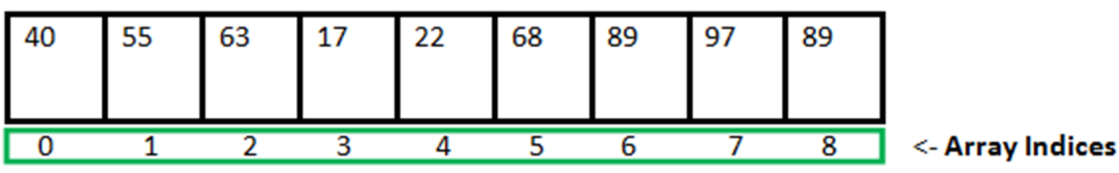
An array is a collection of similar types of data. For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 nam



Array Length = 9
First Index = 0
Last Index = 8

How to declare an array in Java?

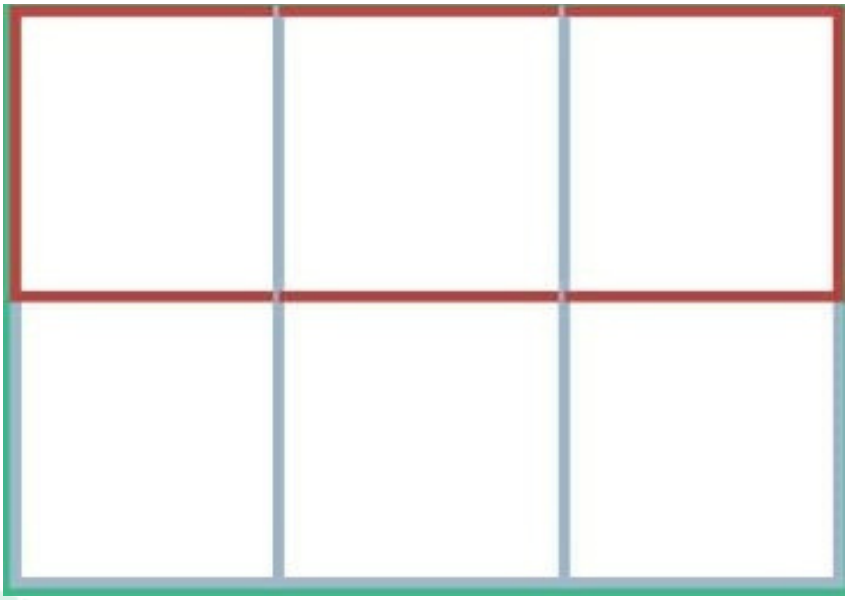
dataType[] arrayName;
double[] data;



Array Length = 9
First Index = 0
Last Index = 8

Array

```
// declare an array
double[] data; //
allocate memory
data = new
Double[10] ; Or Int []
data = new int[10];
```



How to Initialize Arrays in Java?

```
//declare and initialize and array
int[] age = {12, 4, 5, 2, 5}; Or //
declare an array int[] age = new
int[5]; // initialize array age[0] =
12; age[1] = 4; age[2] = 5;
```

How to Initialize Arrays in Java?

• Array indices always start from 0. That is, the first element of an array is at index 0. •If the size of an array is n, then the last element of the array will be at index n-1

How to Access Elements of an Array in Java?

Access Array Elements class Main { public static void main(String[] args) { // create an array int[] age = {12, 4, 5, 2, 5}; // access each array elements System.out.println("Accessing Elements of Array:"); System.out.println("First Element: " + age[0]); System.out.println("Second Element: " + age[1]); System.out.println("Third Element: " + age[2]); System.out.println("Fourth Element: " + age[3]); System.out.println("Fifth Element: " + age[4]); } }

Multidimensional Arrays

•A multidimensional array is an array containing one or more arrays.

•Two-dimensional arrays store the data in rows and columns:

```
int[][] Array = new int[2][5];
```

```
int[][] Array = new int[2][5];
```

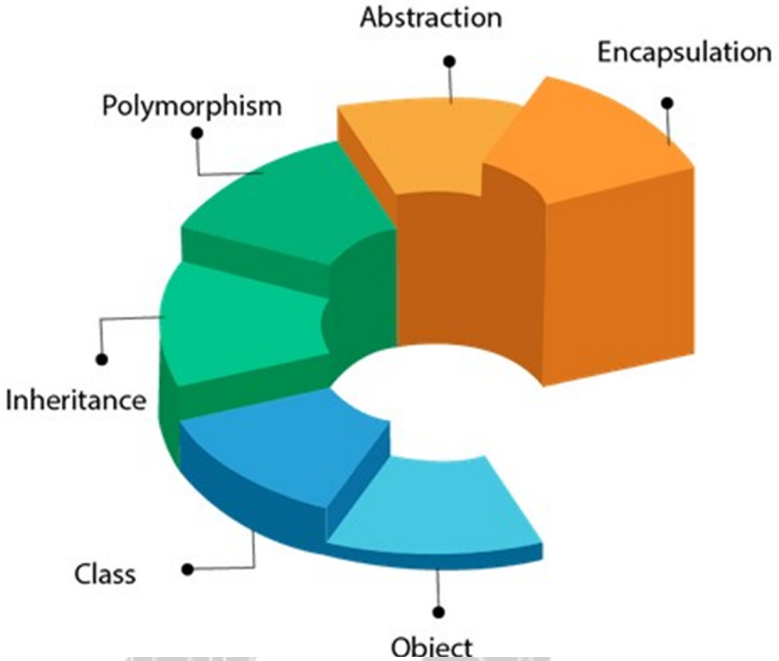
	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Object-oriented programming

•Reaching this milestone courtesy of its object-oriented nature. Java is organized in such a way that everything you program in it becomes either a class or an object.
•Object-oriented programming is about creating objects that contain both data and methods.

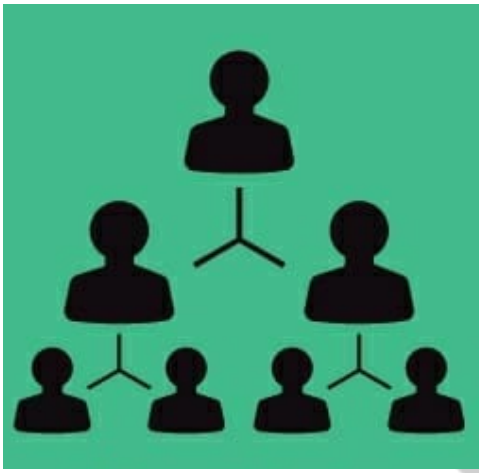
Advantages of Object-oriented programming

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Java code DRY "Don't Repeat Yourself" , and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time



Inheritance

Inheritance is one of the Basic Concepts of OOPs in which one object acquires the properties and behaviors of the parent object. It's creating a parent-child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software.



Polymorphism

•Polymorphism refers to one of the OOPs concepts in Java which is the ability of a variable, object or function to take on multiple forms. For example, in English, the verb run has a different meaning if you use it with a laptop, a foot race, and business. Here, we understand the meaning of run based on the other words used along with it. The same also applied to Polymorphism.

Abstraction

•Abstraction is one of the OOP Concepts in Java which is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application. Lets understand this one of the OOPs Concepts with example, while driving a car, you do not have to be concerned with its internal working. Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

Encapsulation

•Encapsulation is one of the best Java OOPs concepts of wrapping the data and code. In this OOPs concept, the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class. For example - in school, a student cannot exist without a class.

Java Class

•A class is a blueprint or template for the object. Before we create an object, we first need to define the class. •It is a logical entity. It can't be physical

Create a class in Java

```
class
ClassName { //
fields //
methods }
```

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

Java Class

```
class Bicycle
 { // state or field
private int gear = 5; //
behavior or method
public void
braking() {
System.out.println("Wo
rking of Braking"); } }
```

Java Objects

•An object is called an instance of a class. For example, suppose Bicycle is a class then MountainBicycle, SportsBicycle, TouringBicycle, etc can be considered as objects of the class.

Creating an Object in Java

```
className object = new
className(); // for Bicycle class
Bicycle sportsBicycle = new
Bicycle(); Bicycle touringBicycle
= new Bicycle();
```

Key Differences Between Java Classes and Objects

•A class is a blueprint for creating objects
•A class is a logical entity
•The keyword used is "class"
•A class is designed or declared only once
•The computer does not allocate memory when you declare a class •Objects:
•An object is a copy of a class •An object is a physical entity
•The keyword used is "new"
•You can create any number of objects using one single class
•The computer allocates memory when you declare a class

Constructors in Java

•A constructor in Java is similar to a method that is invoked when an object of the class is created.

•A constructor is a special method that is used to initialize an object. Every class has a constructor either implicitly or explicitly.

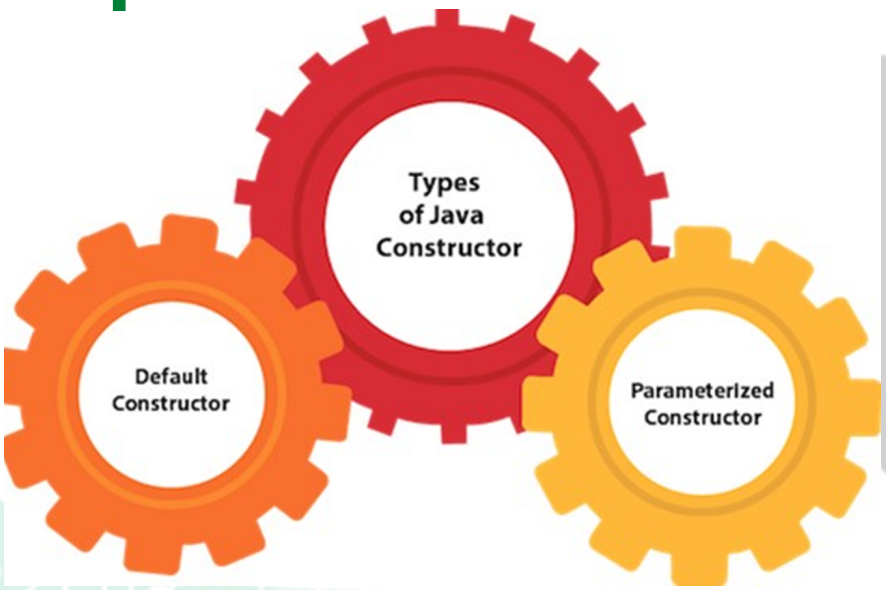
•A constructor has same name as the class name in which it is declared.

•Constructor must have no explicit return type.

•Constructor in Java can not be abstract, static, final .

Constructor Example

```
class Car {
String name ;
String model;
Car(
)//Constructor
{name ="";
model=""; }
```



Types of Constructor

- Java Supports two types of constructors: 1.Default Constructor 2.Parameterized constructor

Default Constructor

- In Java, a constructor is said to be default construct if it does not have any parameter. Default construct can be either user defined or provided by JVM.
- If a class does not contain any constructor then during runtime JVM generates a default constructo which is known as system define default constructo
- If a class contain a constructor with no parameter then it is known as default constructor defined by user. In this case JVM does not create default constructor.

```
public class Demo1
 { public static void
main (String args[]
)
{ Demo1 obj = new
Demo1 () ;
}}
```

User Define Default Constructor

- Constructor which is defined in the class by the programmer is known as user-defined default constructor.

```
class AddDemo1 {
AddDemo1() { int a=10;
int b=5; int c; c=a+b;
System.out.println("*****
Default
Constructor*****");
System.out.println("Total
of 10 + 5 = "+c); } public
static void main(String
args[]) { AddDemo1
obj=new AddDemo1(); } }
```

Constructor vs Method in Java

Constructor	Method
It is a block of code which instantiate a newly created object	It is a collection of statements, always return a value depends upon its execution
They are invoked implicitly	They are invoked explicitly
It does not have any return type	It may return a value
Its name should be same as the class name	Its name should not be same as the class name
If there is no constructor then the default constructor is created by the compiler itself.	In case of method, there is no default method provided

Static in Java

- The static keyword in Java is used for memory management mainly.
- Static is a keyword in Java which is used to declare static stuffs. It can be used to create variable, method block or a class.
- Static variable or method can be accessed without instance of a class because it belongs to class.
- Static members are common for all the instances of the class but non-static members are different for each instance of the class

Child Class:

The class that extends the features of another class is known as child class, sub class or derived class

this keyword

- In Java, this is a keyword which is used to refer current object of a class. we can it to refer any member of the class. It means we can access any instance variable and method by using this keyword. •
- The main purpose of using this keyword is to solve the confusion when we have same variable name for instance and local variables.

Parent Class:

Add a little bit of body teThe class whose properties and functionalities are used(inherited) by another class is known as parent class, super class or Base class.xt

Inheritance in Java

The process by which one class acquires the properties(data members) and functionalities(methods) of another class is called inheritance. The aim of inheritance is to provide the reusability of code so that a class has to write only the unique features and rest of the common properties and functionalities can be extended from the another class.

Why use inheritance in java

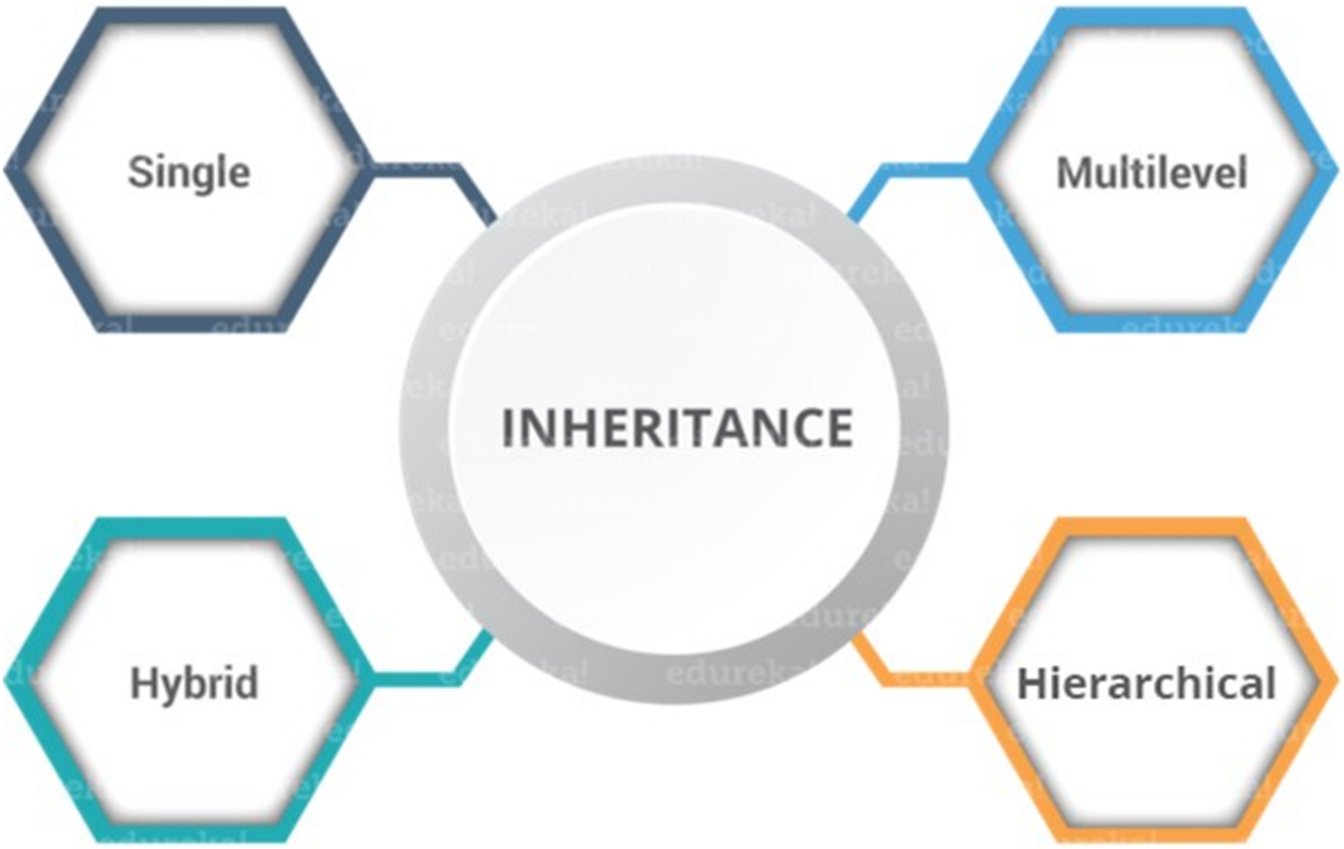
- For Method Overriding (so runtime polymorphism can be achieved).
- For Code Reusability.

OOP

- Inheritance is a process of defining a new class based on an existing class by extending its common data members and methods.
- Inheritance allows us to reuse of code, it improves reusability in your java application.
- Note: The biggest advantage of Inheritance is that the code that is already present in base class need not be rewritten in the child class.
- This means that the data members(instance variables) and methods of the parent class can be used in the child class as.

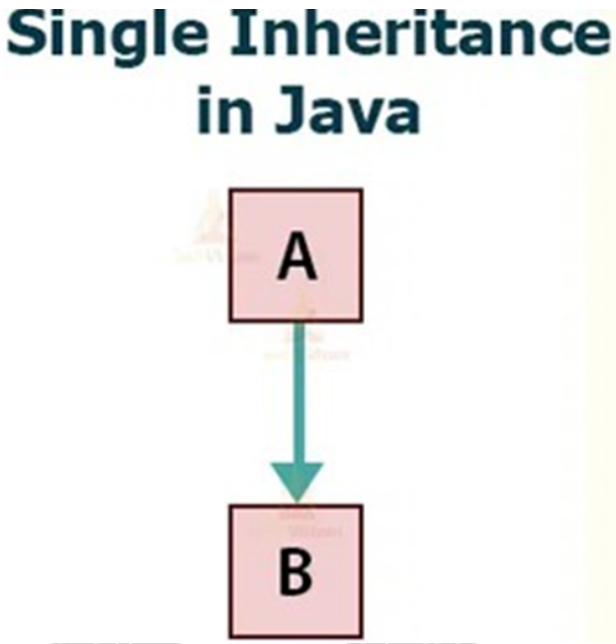
Syntax of Java Inheritance

```
class Subclass-name
extends Superclass-
name { //methods and
        fields } T
```


Single
Multilevel
INHERITANCE
Hybrid
Hierarchical

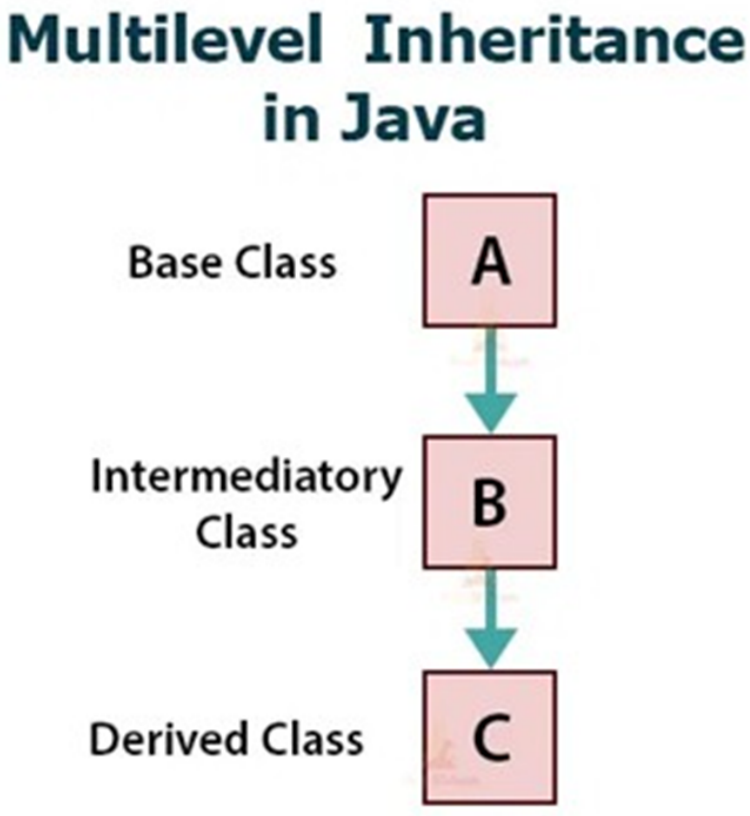
Single Inheritance

•In single inheritance, one class inherits the properties of another. It enables a derived class to inherit the properties and behavior from a single parent class. This will, in turn, enable code reusability as well as add new features to the existing code



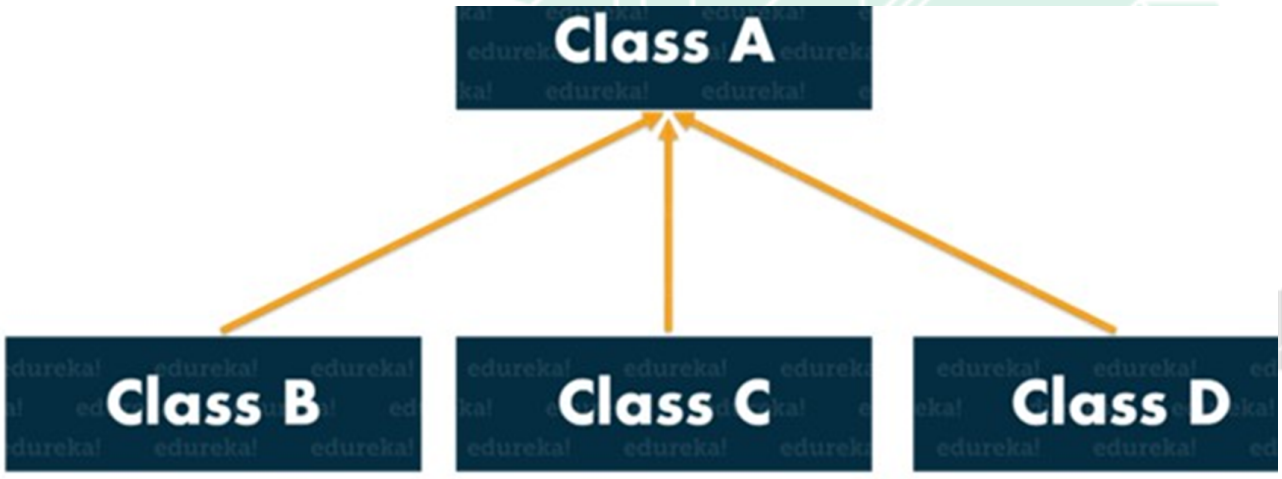
Multi-level Inheritance

- When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.



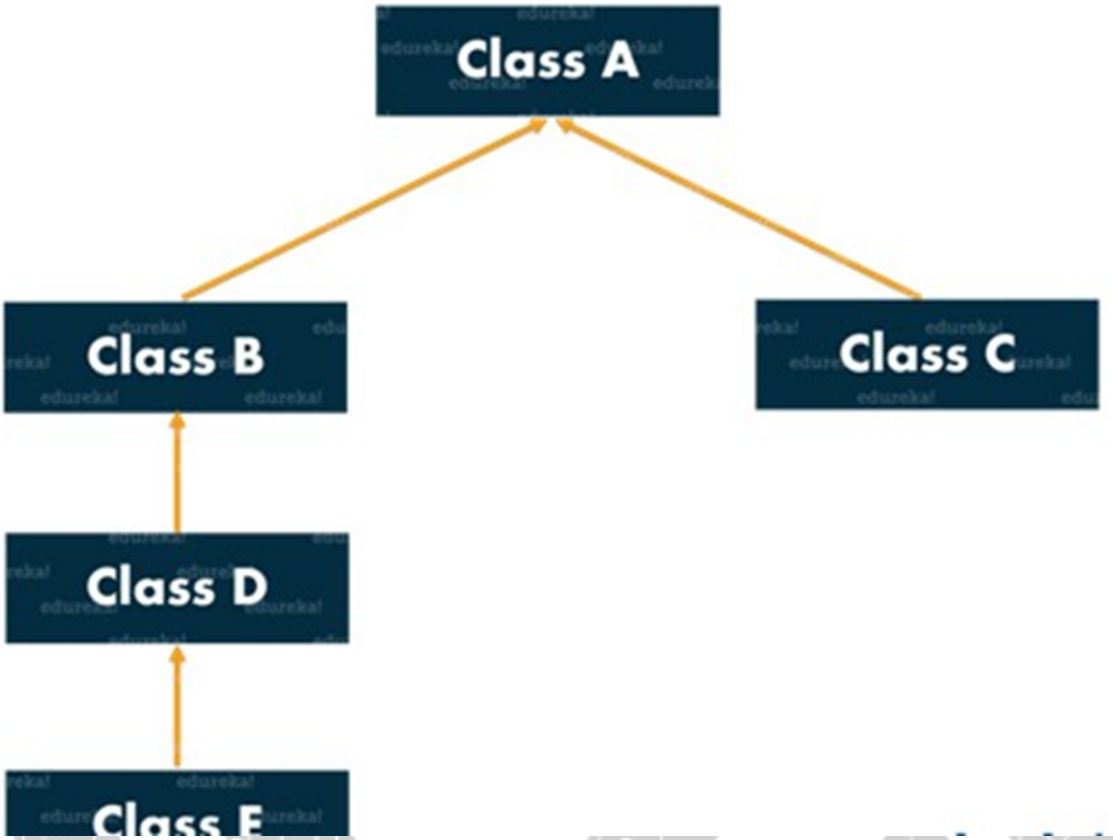
Hierarchical Inheritance

When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.



Hybrid Inheritance

•Hybrid inheritance is a combination of two or more types of inheritance.



Super Keyword

- 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.

Usage of Java super Keyword

1.super can be used to refer immediate parent class instance variable

. 2 .super can be used to invoke immediate parent class method.

3 .super() can be used to invoke immediate parent class constructor

Final Keyword In Java

1.super can be used to refer immediate parent class instance variable.

2.super can be used to invoke immediate parent class method.

3.super() can be used to invoke immediate parent class constructor.

Java final variable

If you make any variable as final, you cannot change the value of final variable(It will be constant)

Java Polymorphism

- If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.
- In Java, we use method overloading and method overriding to achieve polymorphism.
- Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

Method Overloading in Java

- If a class has multiple methods having same name but different in parameters, it is known as Method Overloading.
- If we have to perform only one operation, having same name of the methods increases the readability of the program.
- Suppose you have to perform addition of the given numbers but there can be any number of arguments, if you write the method such as a(int,int) for two parameters, and b(int,int,int) for three parameters then it may be difficult for you as well as other programmers to understand the behavior of the method because its name differs.

Method overloading increases the readability of the program. Different ways to overload the method By changing the no. of arguments By changing the datatype

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. class Adder{ static int add(int a,int b){return a+b;} static int add(int a,int b,int c){return a+b+c;} } class TestOverloading1{ public static void main(String[] args){ System.out.println(Adder.add(11,11)); System.out.println(Adder.add(11,11,11)); }}

Method Overriding in Java

•If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java

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

•Usage of Java Method Overriding

•Method overriding is used to provide the specific implementation of a method which is already provided by its superclass

. •Method overriding is used for runtime polymorphism

Runtime Polymorphism in Java

- Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.
- In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable. 1

Abstraction in Java

Abstraction is a process of hiding the implementation details and showing only functionality to the user. 1.

Abstract class in Java

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Interface

- 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.
- In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. 1

Why use Java interface?

There are mainly two reasons to use interface. ·It is used to achieve abstraction·By interface, we can support the functionality of multiple inheritance.

Java Package

A java package is a group of similar types of classes, interfaces and sub-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-defined packages

82

Advantage of Java Package

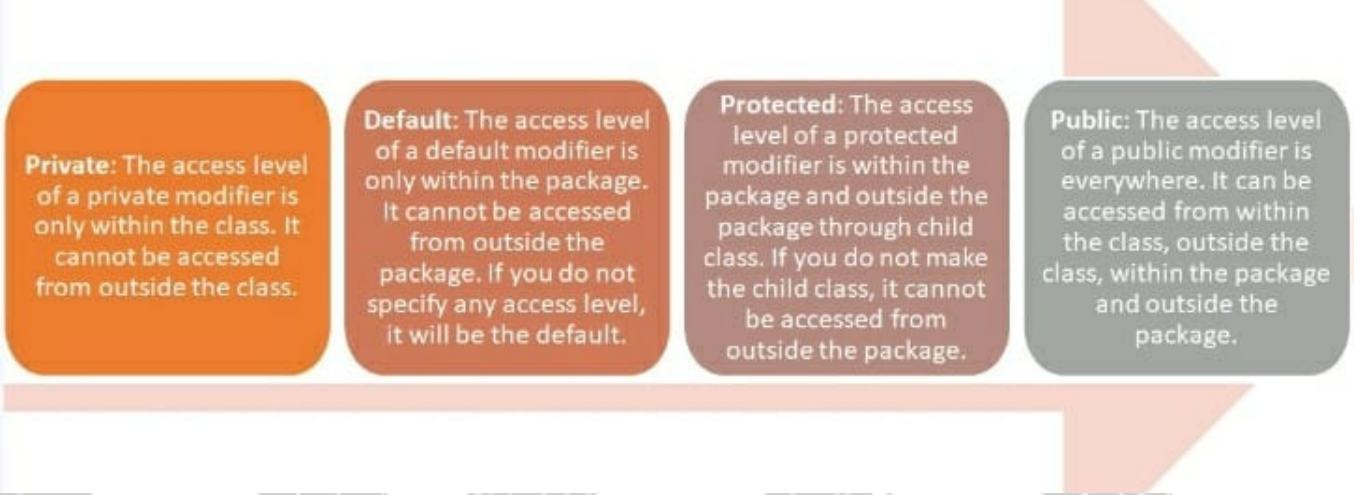
1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

Access Modifiers in Java

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types of Java access modifiers:



1) Private

The private access modifier is accessible only within the class. E.g Simple example of private access modifier

2)Default

f you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package. It cannot be accessed from outside the package. It provides more accessibility than private. But, it is more restrictive than protected, and public. 1

3)Public

The public access modifier is accessible everywhere. It has the widest scope among all other modifiers

4)Protected

- The protected access modifier is accessible within package and outside the package but through inheritance only.
- The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.
- It provides more accessibility than the default modifer

Encapsulation in Java

- Encapsulation in Java is a process of wrapping code and data together into a single unit, for example, a capsule which is mixed of several medicines.
- We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

Advantage of Encapsulation in Java

- By providing only a setter or getter method, you can make the class read-only or write-only. In other words, you can skip the getter or setter methods.
- It provides you the control over the data. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.
- It is a way to achieve data hiding in Java because other class will not be able to access the data through the private data members. 1.

String Handling

1.Strings Class

•A combination of characters is a string.

•String s = new string () ;

String Arrays

We can also create and use string arrays. String myarray[] = new String [3] ;

Java StringBuffer class

Java String Buffer class is used to create mutable (modifiable) string. The String Buffer class in java is same as String class except it is mutable i.e. it can be changed

append()

This method will concatenate the string representation of any type of data to the end of the StringBuffer object. append() method has several overloaded forms.

insert()

•This method inserts one string into another. Here are few forms of insert() method

reverse()

•This method reverses the characters within a StringBuffer object

replace()

This method replaces the string from specified start index to the end index

capacity()

•This method returns the current capacity of StringBuffer object

Exception Handling in Java

- 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.
- When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things
- . When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).
- Java try and catch
- The try statement allows you to define a block of code to be tested for errors while it is being executed.
- The catch statement allows you to define a block of code to be executed, if an error occurs in the try block. The try and catch keywords come in pairs:

Try block

The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.

```
try{ //statements
that may cause an exception }
```

Catch block

A catch block is where you handle the exceptions, this block must follow the try block. A single try block can have several catch blocks associated with it. You can catch different exceptions in different catch blocks. When an exception occurs in try block, the corresponding catch block that handles that particular exception executes. For example if an arithmetic exception occurs in try block then the statements enclosed in catch block for arithmetic exception executes.

Java Finally block – Exception handling

A finally block contains all the statements that must be executed whether exception occurs or not. The statements present in this block will always execute regardless of whether exception occurs in try block or not such as closing a connection, stream etc.

```
try {
//Statements that
may cause an
exception
 } catch {
 //Handling
exception
 } finally {
//Statements to
be executed
```

Java FileOutputStream Class

- java File Output Stream is an output stream used for writing data to a File
- If you have to write primitive values into a file, use File Output Stream class.
- You can write byte-oriented as well as character-oriented data through FileOutputStream class

void write(byte[] ary):Description It is used to write ary.length bytes from the byte array to the file output stream. void write(int b):It is used to write the specified byte to the file output stream. void close():It is used to closes the file output stream.

Multithreading

•Multithreading is a conceptual programming concept where a program (process) is divided into two or more subprograms (process), which can be implemented at the same time in paraller. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread define a separate path of execution.

•Enable progrrammers to do multiple things at one time. •Programmes can divide a long program into threads and execute them in parallel which eventually increases there speed of the program execution.
•Improved performance and concurrency.
•Simultaneous access to multiple applications.

Thread class:

Thread class provide constructors and methods to create and perform operations on a thread .Thread class extends Object class and implements Runnable interface. Commonly used Constructors of Thread class: Thread() Thread(String name) Thread(Runnable r) Thread(Runnable r,String name)

Java Thread Example by extending Thread class

Class Multi extends Thread{ public void run(){ System. out. println("thread is running... "); } public static void main(String args[]){ Multi t1=new Multi(); t1.start(); } } Java Thread Example by extending Thread class T

Swing in Java

- java swing consists of a set of components or widgets that are used to develop Graphical User Interface Applications.
- Java Swing is Lightweight GUI Toolkit that includes Buttons, Textboxes, etc...
- Java Swing components are platform-independent. Swing are pluggable look and fee

Introduction to Swing Classes

- JPanel : JPanel is Swing's version of AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent
- . JFrame : JFrame is Swing's version of Frame and is descended directly from Frame class. The component which is added to the Frame, is refered as its Content.
- JWindow : This is Swing's version of Window and has descended directly from Window class. Like Window it uses BorderLayout by default
- . JLabel : JLabel has descended from JComponent, and is used to create text labels.
- JButton : JButton class provides the functioning of push button. JButton allows an icon, string or both associated with a button
- . JTextField : JTextFields allow editing of a single line of text.

JButton

JButton class is used to create a push-button on the UI. The button can contain some display text or image. It generates an event when clicked and double-clicked. JButton okBtn = new JButton(“Ok”);

JLabel

JLabel class is used to render a read-only text label or images on the UI. It does not generate any event. JLabel textLbl = new JLabel(“This is a text label. ”);

JTextField

JTextField renders an editable single-line text box. A user can input non-formatted text in the box. To initialize the text field, call its constructor and pass an optional integer parameter to it. This parameter sets the width of the box measured by the number of columns. It does not limit the number of characters that can be input in the box

JPasswordField

JPassword Field is a subclass of J TextField class. It renders a text-box that masks the user input text with bullet points. This is used for inserting passwords into the application.

JCheckBox

JCheckBox renders a check-box with a label. The check-box has two states – on/off. When selected, the state is on and a small tick is displayed in the box.

JRadioButton

•JRadioButton is used to render a group of radio buttons in the UI. A user can select one choice from the group

JMenuBar, JMenu and JMenuItem

- public class JMenuBar extends JComponent implements MenuElement, Accessible ·
- public class JMenu extends JMenuItem implements MenuElement, Accessible ·
- public class JMenuItem extends AbstractButton implements Accessible, MenuElement

JProgressBar

- In Java, Swing toolkit contains a JProgressBar Class. It is used for creating a progress bar of a task.
- 1.JProgressBar()
- 2.JProgressBar(int min, int max)
- 3.JProgressBar(int orient)
- 4.JProgressBar(intorient,int min, int max)