What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the name from Oak to Java.

**Platform**: Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

**Simple.java**

1. **class** Simple{
2. **public** **static** **void** main(String args[]){
3. System.out.println("Hello Java");
4. }
5. }

## Application

1. Desktop Applications such as acrobat reader, media player, antivirus, etc.
2. Web Applications such as irctc.co.in, javatpoint.com, etc.
3. Enterprise Applications such as banking applications.
4. Mobile
5. Embedded System
6. Smart Card
7. Robotics
8. Games, etc.

### **What is an object in Java**

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system. An object has three characteristics:

**An object is an instance of a class.** A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

**Object Definitions:**

* An object is *a real-world entity*.
* An object is *a runtime entity*.
* The object is *an entity which has state and behavior*.
* The object is *an instance of a class*.

### **Object and Class Example: main within the class**

1. /Java Program to illustrate how to define a class and fields
2. //Defining a Student class.
3. **class** Student{
4. //defining fields
5. **int** id;//field or data member or instance variable
6. String name;
7. //creating main method inside the Student class
8. **public** **static** **void** main(String args[]){
9. //Creating an object or instance
10. Student s1=**new** Student();//creating an object of Student
11. //Printing values of the object
12. System.out.println(s1.id);//accessing member through reference variable
13. System.out.println(s1.name);
14. }  }
15. Output:
16. 0
17. null

### **Object and Class Example: main outside the class**

1. **class** Student{
2. **int** id;
3. String name;
4. }
5. //Creating another class TestStudent1 which contains the main method
6. **class** TestStudent1{
7. **public** **static** **void** main(String args[]){
8. Student s1=**new** Student();
9. System.out.println(s1.id);
10. System.out.println(s1.name);
11. }
12. }

Output:

**Compile by: javac TestStudent1.java**

**Run by: java TestStudent1**

0  
null

3 Ways to initialize object

There are 3 ways to initialize object in Java.

1. By reference variable
2. By method
3. By constructor

### **) Object and Class Example: Initialization through reference**

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

*File: TestStudent2.java*

1. **class** Student{
2. **int** id;
3. String name;
4. }
5. **class** TestStudent2{
6. **public** **static** **void** main(String args[]){
7. Student s1=**new** Student();
8. s1.id=101;
9. s1.name="Sonoo";
10. System.out.println(s1.id+" "+s1.name);//printing members with a white space
11. }
12. }
13. Output:
14. 101 Sonoo

### **2) Object and Class Example: Initialization through method**

1. **class** Student{
2. **int** rollno;
3. String name;
4. **void** insertRecord(**int** r, String n){
5. rollno=r;
6. name=n;
7. }
8. **void** displayInformation(){System.out.println(rollno+" "+name);}
9. }
10. **class** TestStudent4{
11. **public** **static** **void** main(String args[]){
12. Student s1=**new** Student();
13. Student s2=**new** Student();
14. s1.insertRecord(111,"Karan");
15. s2.insertRecord(222,"Aryan");
16. s1.displayInformation();
17. s2.displayInformation();
18. }
19. }

Output:

111 Karan

222 Aryan

### **3) Object and Class Example: Initialization through a constructor**

# **Constructors in Java**

In [Java](https://www.javatpoint.com/java-tutorial)

, a constructor is a block of codes similar to the method. It is called when an instance of the [class](https://www.javatpoint.com/object-and-class-in-java)

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.

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

### **Example of default constructor that displays the default values**

1. //which displays the default values
2. **class** Student3{
3. **int** id;
4. String name;
5. //method to display the value of id and name
6. **void** display(){System.out.println(id+" "+name);}
8. **public** **static** **void** main(String args[]){
9. //creating objects
10. Student3 s1=**new** Student3();
11. Student3 s2=**new** Student3();
12. //displaying values of the object
13. s1.display();
14. s2.display();
15. }
16. }

Output:

0 null

0 null

### **Java Parameterized Constructor**

A constructor which has a specific number of parameters is called a parameterized constructor.

1. **class** Student4{
2. **int** id;
3. String name;
4. //creating a parameterized constructor
5. Student4(**int** i,String n){
6. id = i;
7. name = n;
8. }
9. //method to display the values
10. **void** display(){System.out.println(id+" "+name);}
12. **public** **static** **void** main(String args[]){
13. //creating objects and passing values
14. Student4 s1 = **new** Student4(111,"Karan");
15. Student4 s2 = **new** Student4(222,"Aryan");
16. //calling method to display the values of object
17. s1.display();
18. s2.display();
19. }
20. }

class Student5{

int id;

String name;

int age;

Student5(int i,String n){

id = i;

name = n;

}

Student5(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[]){

Student5 s1 = new Student5(111,"Karan");

Student5 s2 = new Student5(222,"Aryan",25);

s1.display();

s2.display();

}

}

OUTPUT

**Compile by: javac Student5.java**

**Run by: java Student5**

111 Karan 0  
222 Aryan 25

### Java Constructors vs. Methods**Can constructor perform other tasks instead of initialization?**

Yes, like object creation, starting a thread, calling a method, etc. You can perform any operation in the constructor as you perform in the method.

# **Java static keyword**

The **static keyword** in [Java](https://www.javatpoint.com/java-tutorial) is used for memory management mainly. We can apply static keyword with [variables](https://www.javatpoint.com/java-variables), methods, blocks and [nested classes](https://www.javatpoint.com/java-inner-class). The static keyword belongs to the class than an instance of the class.

The static can be:

1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class

### **Example of static variable**

1. //Java Program to demonstrate the use of static variable
2. **class** Student{
3. **int** rollno;//instance variable
4. String name;
5. **static** String college ="ITS";//static variable
6. //constructor
7. Student(**int** r, String n){
8. rollno = r;
9. name = n;
10. }
11. //method to display the values
12. **void** display (){System.out.println(rollno+" "+name+" "+college);}
13. }
14. //Test class to show the values of objects
15. **public** **class** TestStaticVariable1{
16. **public** **static** **void** main(String args[]){
17. Student s1 = **new** Student(111,"Karan");
18. Student s2 = **new** Student(222,"Aryan");
19. //we can change the college of all objects by the single line of code
20. //Student.college="BBDIT";
21. s1.display();
22. s2.display();
23. }
24. }

Output:

111 Karan ITS

222 Aryan ITS

## 2) Java static method

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

* A static method belongs to the class rather than the object of a class.
* A static method can be invoked without the need for creating an instance of a class.
* A static method can access static data member and can change the value of it.

### **Example of static method**

1. **class** Student{
2. **int** rollno;
3. String name;
4. **static** String college = "ITS";
5. //static method to change the value of static variable
6. **static** **void** change(){
7. college = "BBDIT";
8. }
9. //constructor to initialize the variable
10. Student(**int** r, String n){
11. rollno = r;
12. name = n;
13. }
14. //method to display values
15. **void** display(){System.out.println(rollno+" "+name+" "+college);}
16. }
17. //Test class to create and display the values of object
18. **public** **class** TestStaticMethod{
19. **public** **static** **void** main(String args[]){
20. Student.change();//calling change method
21. //creating objects
22. Student s1 = **new** Student(111,"Karan");
23. Student s2 = **new** Student(222,"Aryan");
24. Student s3 = **new** Student(333,"Sonoo");
25. //calling display method
26. s1.display();
27. s2.display();
28. s3.display();
29. }
30. }

Output:111 Karan BBDIT

222 Aryan BBDIT

333 Sonoo BBDIT

# **this keyword in Java**

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



## Usage of Java this keyword

Here is given the 6 usage of java this keyword.



# **Inheritance in Java**

**Inheritance in Java** is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of [OOPs](https://www.javatpoint.com/java-oops-concepts)

(Object Oriented programming system).

The idea behind inheritance in Java is that you can create new [classes](https://www.javatpoint.com/object-and-class-in-java)

that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the **IS-A relationship** which is also known as a parent-child relationship.

### **Why use inheritance in java**

* For [Method Overriding](https://www.javatpoint.com/method-overriding-in-java)

(so [runtime polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)

can be achieved).

* For Code Reusability.

### **Terms used in Inheritance**

* **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
* **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
* **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
* **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

### **The syntax of Java Inheritance**

1. **class** Subclass-name **extends** Superclass-name
2. {
3. //methods and fields
4. }

The **extends keyword** indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

1. **class** Employee{
2. **float** salary=40000;
3. }
4. **class** Programmer **extends** Employee{
5. **int** bonus=10000;
6. **public** **static** **void** main(String args[]){
7. Programmer p=**new** Programmer();
8. System.out.println("Programmer salary is:"+p.salary);
9. System.out.println("Bonus of Programmer is:"+p.bonus);
10. }
11. }
12. Programmer salary is:40000.oorammer salary is:40000.0
13. Boonus of programmer is:10000
14. Programmer salary is:40000.0
15. Bonus of programmer is:10000

# Polymorphism in Java

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

**Real life example of polymorphism:** A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.

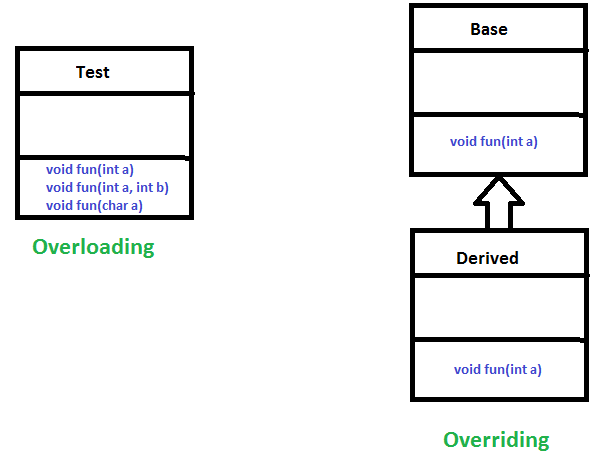
The word “poly” means many and “morphs” means forms, So it means many forms.

**In Java polymorphism is mainly divided into two types:**

* Compile time Polymorphism
* Runtime Polymorphism

**1. Compile-time polymorphism**: It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading. But **Java doesn’t support the Operator Overloading**.

**2.Method Overloading**: When there are multiple functions with same name but different parameters then these functions are said to be **overloaded**. Functions can be overloaded by **change in number of arguments** or/and **change in type of arguments**.

[](https://www.geeksforgeeks.org/overloading-in-java/)