OBJECT

Object: 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).

An object is an instance of a class.

Characteristics of an object in Java

An object has three characteristics:

1. **State:** State represents properties of an object. It is represented by instance variables/attributes of an object. The properties of an object are important because the outcome of functions depends on the properties.

2. **Behavior:** Behavior represents functionality or actions. It is represented by methods in Java.

3. **Identity:** Identity represents the unique name of an object. It differentiates one object from the other. The unique name of an object is used to identify the object.

3 Ways to initialize object

There are 3 ways to initialize objects in Java.

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

What are the different ways to create an object in Java?

There are many ways to create an object in java. They are:

- By new keyword
- By newInstance() method
- By clone() method
- By deserialization
- By factory method etc.

Anonymous object: An object which has no reference variable is called an anonymous object in Java.

For example: **new** Student();

We can pass the parameter to the constructor like this:

new Student("Shubh");

If you want to call a method through the anonymous object then you can write code like this:

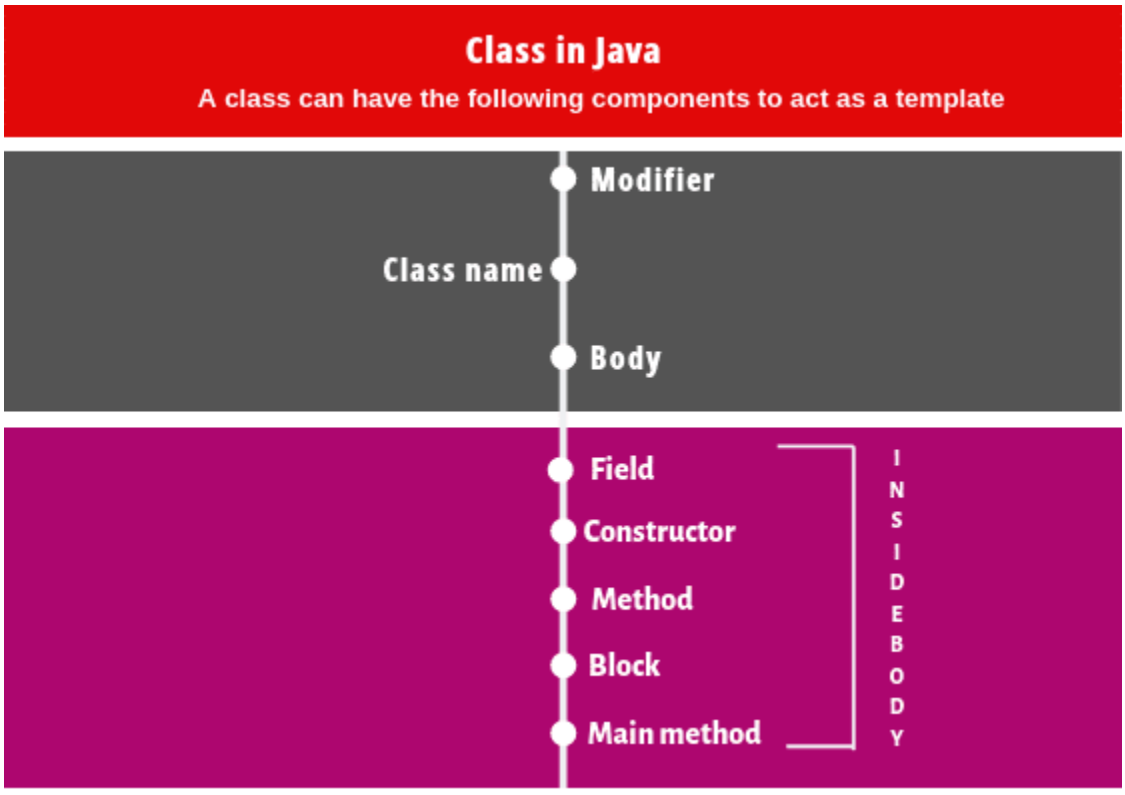
new Student().display();

We can also pass the parameter to the calling method like this:

new Student().display("Shubh", 25);

CLASS

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



A class can contain any of the following **variable types:**

- **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
- **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
- **Class variables/Static variables** − Class variables are variables declared within a class, outside any method, with the static keyword.

Scope of a variable means simply the region of the program where a variable is accessible.

1. The scope of local variables always remains inside the constructor, method, and block. It can not be accessible from outside the constructor, method, and block.

2. The scope of instance variables is within the class. They can be accessed from all the methods, constructors, and from the beginning of the program block to the end of the program block in the class.

3. The scope of static variables is also within the class. All the methods, constructors, and blocks within class can access static variables by using the class name.

Can we call Class as Data type in Java? - Yes, a class is also considered a user-defined data type. This is because a user creates a class.

Reference variables: The only way you can access an object is through a **reference variable**. A reference variable is declared to be of a specific type and that type can never be changed.
Reference variables can be declared as static variables, instance variables, method parameters, or local variables.
A reference variable that is declared as final can't never be reassigned to refer to a different object. The data within the object can be modified, but the reference variable cannot be changed.

Understanding Reference variable

1. Reference variable is used to point object/values.

2. Classes, interfaces, arrays, enumerations, and, annotations are reference types in Java. Reference variables hold the objects/values of reference types in Java.

3. Reference variable can also store **null** value. By default, if no object is passed to a reference variable then it will store a null value.

4. You can access object members using a reference variable using **dot** syntax.

Types of classes in Java:

1. Concrete Class:

A class whose object can be created and whose all methods have a body is called concrete class.

Java object class is a concrete class. A concrete class can have both static and non-static members.

It can extend its superclass, an abstract class, or implement an interface if it implements all their methods. It has no abstract methods.

2. Static Class:

In Java, static is a keyword that can be applied with variables, methods, nested classes, and blocks. We cannot declare a class with a static keyword but the nested class can be declared as static.

When a nested class is defined with a static modifier inside the body of another class, it is known as a static nested class in Java.

3. Abstract Class:

An abstract class is a class which is declared with an abstract keyword. It is just like a normal class but has two differences.

1. We cannot create an object of this class. Only objects of its non-abstract (or concrete) sub-classes can be created.

2. It can have zero or more abstract methods that are not allowed in a non-abstract class (concrete class).

4. Final Class:

When a class is declared with the final keyword, it is called final class in Java. Final class means Restricting Inheritance!. It does not allow itself to be inherited by another class.

In other words, Java classes declared as final cannot be extended (inherited). If you do not want to be a subclass, declare it final. A lot of classes in Java API are final.

For example, String class is the most common predefined final class object in Java.

5. Inner class:

A class declared inside another class is known as a nested class. The class which is a member of another class can be either static or non-static.

When a member class is declared with static, it is known as static nested class. *The member class which is non-static is known as the inner class in java.*

Based on declaration and behaviors, there are basically four types of inner classes in Java. They are as follows:

1. Normal or Regular inner class
2. Method local inner class
3. Anonymous inner class

6. Public Class:

When a class is declared with a public access control modifier, it is called public class. A public class is visible and accessible from anywhere. The instance of public class can be created from any other class.

7. Private class:

An outer class (top-level class) cannot be declared with a private access modifier but an inner class can be declared as private. An inner class is a member of outer class.

8. Singleton Class:

A singleton class in Java is a class that allows only one instance to be created. All its variables and methods will belong to just one instance. Singleton class concept is useful when we need to create only one object of class.

A good example of a singleton is a data class that contains all the data to be used across the entire application. There is no need to create more than one object.

The main disadvantage of a singleton class is that it cannot be reusable. This is because we can create only one object of class.

Predefined Classes in Java

1. Object Class:

Object class is the root of Java class hierarchy which is defined in **java.lang package.** It is the superclass of all other classes. It means that every class inherits methods that are defined in the Object class. Object provides wait(), notify(), and notifyAll() methods for multithreading.

2. Math Class:

Math class is present in the **java.lang package**. All the methods present in this class are static. So, we do not need to create an object of class to use methods. Math class's methods are used for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

For example:

double root;

root = Math.sqrt(10); // It will find a square root of 10.

3. String Class:

String class is present in **java.lang package**. It represents character strings. It is a powerful concept that provides a lot of methods to work on strings. With the help of these methods, we can perform operations on strings such as concatenating, converting, comparing, replacing strings, trimming, etc.

Since strings are constant(immutable), therefore, their values cannot be changed once they are created. For example: String str = "abc";.

4. System Class:

System class is present in the **java.lang package**. All the methods and variables of system class are class methods and variables. They are defined with a static keyword that makes them unique.

We can directly use system class methods and variables in a program. We access them using class names instead of creating an object of class.

5. Random Class:

Random class is present in the **java.util package**. It is used to generate a list of random numbers. Random numbers are used in creating simulations or models of real-world situations, with programs.

6. Scanner Class:

Scanner class is present in the **java.util package**. It is used to read input from a keyboard or text file. When the scanner class receives input from the keyboard then it breaks input into several parts which are called tokens. These tokens are retrieved from the scanner object using methods such as next(), nextByte(), nextInt(), etc.

7. Wrapper Class:

A class whose object warps or contains a primitive data type is called wrapper class. It is present in the **java.lang package.** It is used to convert primitive data type into object form. When we create an object to a wrapper class, it contains a field where we can store a primitive data type.

That is, we can wrap a primitive value into a wrapper class object. The list of wrapper classes defined in java.lang package is Character, Byte, Short, Integer, Long, Float, Double, and Boolean.

Constructor

Definition: A constructor in java is a block of code, similar to a method that is used to initialize the state of an object in a class.

Basically, there are two types of constructors in java. They are as:

1. **Default Constructor** (No-argument constructor)
2. **Parameterized Constructor** (Argument constructor)

There are another type of constructor named **copy constructor**: A constructor which is used to copy the data of one object to another object of the same class type is called copy constructor in Java.

```java
class Student{
    int roll;
    // make default constructor
    public Student(){

    }
    // make a parameterized constructor
    public Student(int roll){
        this.roll = roll;
    }
    // make a copy constructor
    public Student(Student s){
        roll = s.roll;
    }
    public void change(){
        roll--;
    }
    public void printRoll(){
        System.out.println(roll);
    }
}

public class MyClass {
    public static void main(String args[]) {
      Student s1 = new Student(17);
      s1.printRoll();

      Student s2 = new Student(s1); // instantiating s2 with a copy consturctor (which copies the fields of s1)
      s2.change();
      s2.printRoll();

      s1.printRoll(); // s1 roll will not change
    }
}
```

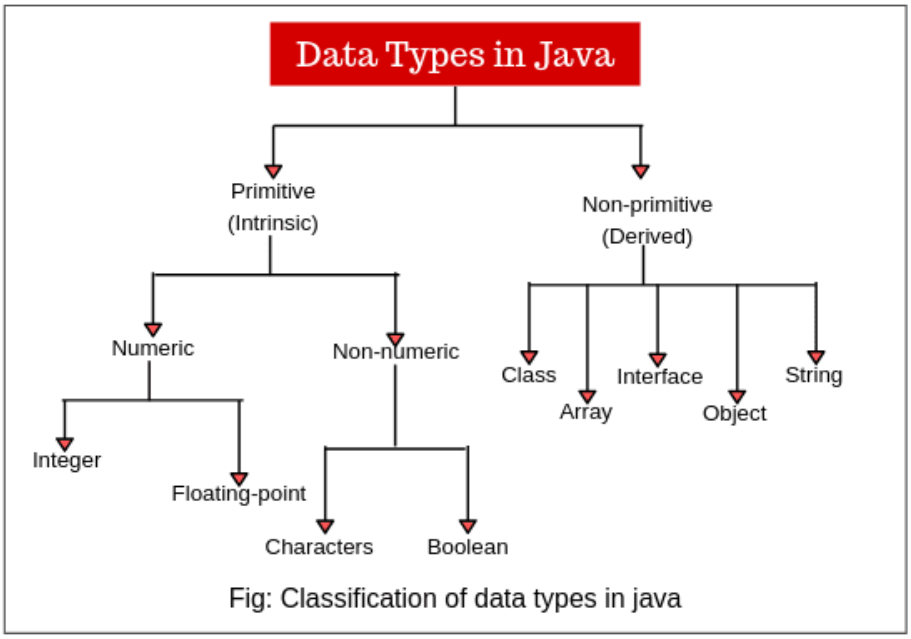


Fig: Classification of data types in java

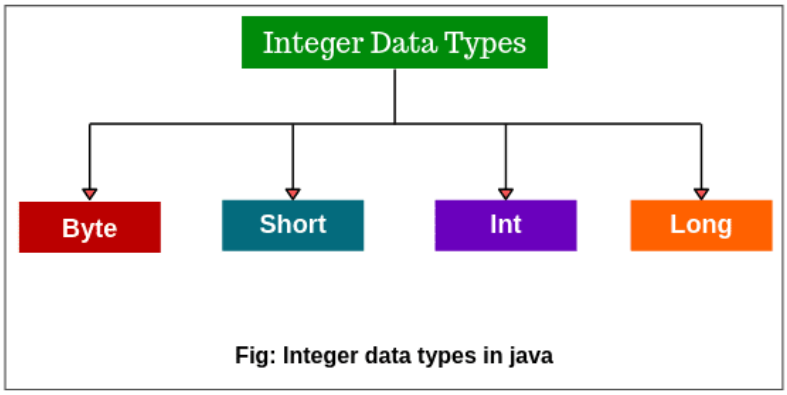


Fig: Integer data types in java

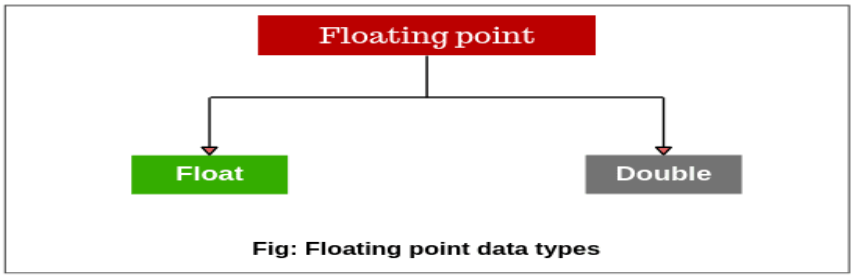


Fig: Floating point data types

Modifiers

Definition: Modifiers are keywords that you add to those definitions to change their meanings.

Java language has a wide variety of modifiers, including the following −

- Java Access Modifiers
- Non Access Modifiers

Access Control Modifiers

Java provides a number of access modifiers to set access levels for classes, variables, methods and constructors. The four access levels are −

- Visible to the package, the **default**. No modifiers are needed.
- Visible to the class only (**private**).
- Visible to the world (**public**).
- Visible to the package and all subclasses (**protected**).

Non-Access Modifiers

Java provides a number of non-access modifiers to achieve many other functionality.

- The ***static*** modifier for creating class methods and class variables.
- The ***final*** modifier for finalizing the implementations of classes, methods, and variables.
- The ***abstract*** modifier for creating abstract classes and methods.
- The ***synchronized*** and ***volatile*** modifiers, which are used for threads.