Java Class

- **What is a class?**

In Java everything is encapsulated under classes. Class is the core of Java language. It can be defined as a template that describe the behaviours and states of a particular entity.

A class defines new data type. Once defined this new type can be used to create object of that type.

Object is an instance of class. You may also call it as physical existence of a logical template class.

In Java, to declare a class, 'class' keyword is used. A class contain both data and methods that operate on that data. The data or variables defined within a class are called instance variables and the code that operates on this data is known as methods.

Thus, the instance variables and methods are known as class members.

- **Rules for Java Class**

1. A class can have only public or default (i.e. no modifier) access specifier.

2. It can be either abstract, final or concrete (normal class).

3. It must have the class keyword, and class must be followed by a legal identifier.

4. The variables and methods are declared within a set of curly braces.

5. A Java class can contain fields, methods, constructors, and blocks. Let's see a general structure of a class.

- **Java class Syntax**

```
class  class_name {
    field;
    method;
}
```

- **Example**

Suppose, Student is a class and student's name, roll number, age are it’s fields and info() is a method. Then class will look like below.

```
class Student
{
 String name;
 int rollno;
 int age;
void info(){
 // some code
}
}
```

This is how a class look structurally. It is a blueprint for an object. We can call it’s fields and methods by using the object.

The fields declared inside the class are known as instance variables. It gets memory when an object is created at runtime.

Methods in the class are similar to the functions that are used to perform operations and represent behaviour of an object.

Java Object

- **What is an object?**

Object is an instance of a class while class is a blueprint of an object. An object represents the class and consists of properties and behaviour.

Properties refer to the fields declared within the class and behaviour represents to the methods available in the class.

In real world, we can understand object as a cell phone that has it’s properties like: name, cost, colour etc and behaviour like calling, chatting etc.

So, we can say that object is a real world entity. Some real-world objects are: ball, fan, car etc.

There is a syntax to create an object in the Java.

- **Java Object Syntax**

className variable_name = new className();

Here, className is the name of class that can be anything like: Student that we declared in the above example.

variable_name is name of reference variable that is used to hold the reference of created object.

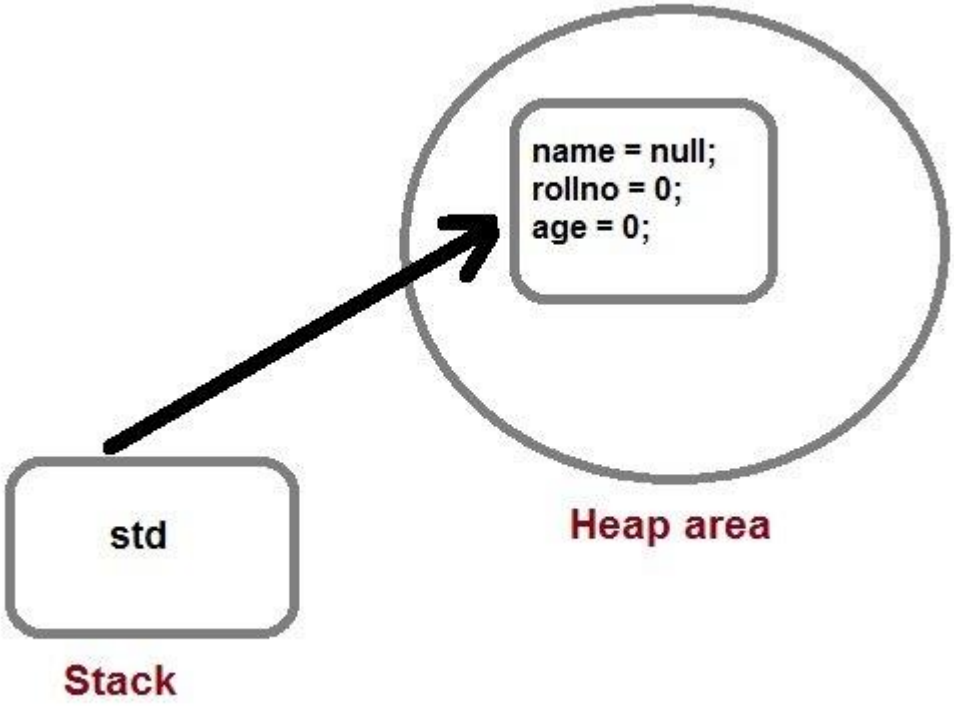
The new is a keyword which is used to allocate memory for the object.

- **Example: Object creation**

Student std = new Student();

Here, std is an object that represents the class Student during runtime.

The new keyword creates an actual physical copy of the object and assigns it to the std variable. It will have physical existence and get memory in heap area. The new operator dynamically allocates memory for an object.



In this image, we can get an idea how an object refers to the memory area.

Now let's understand object and class combinedly by using a real example.

Example: Creating a Class and it's object

```
class Student{
        String name;
        int rollno;
        int age;
       void info(){
         System.out.println("Name: "+name);
         System.out.println("Roll Number: "+rollno);
         System.out.println("Age: "+age);
       }
```

```
    public static void main(String[] args) {
        Student student = new Student();
        // Accessing and property value
        student.name = "Ramesh";
        student.rollno = 253;
        student.age = 25;
        // Calling method
        student.info();
    }
}
```

In this example, we created a class Student and it's object. Here you may be surprised of seeing main() method but don't worry it is just an entry point of the program by which JVM starts execution.

Here, we used main method to create object of Student class and access it's fields and methods.

Example: Class fields

```
class Student{
     String name;
     int rollno;
     int age;
    void info(){
      System.out.println("Name: "+name);
      System.out.println("Roll Number: "+rollno);
      System.out.println("Age: "+age);
    }
    public static void main(String[] args) {
        Student student = new Student();
```

```
        // Calling method
        student.info();
    }
}
```

In case, if we don't initialized values of class fields then they are initialized with their default values.

- **Default values of instance variables**

int, byte, short, long -> 0

float, double → 0.0

string or any reference = null

boolean → false

These values are initialized by the default constructor of JVM during object creation at runtime.