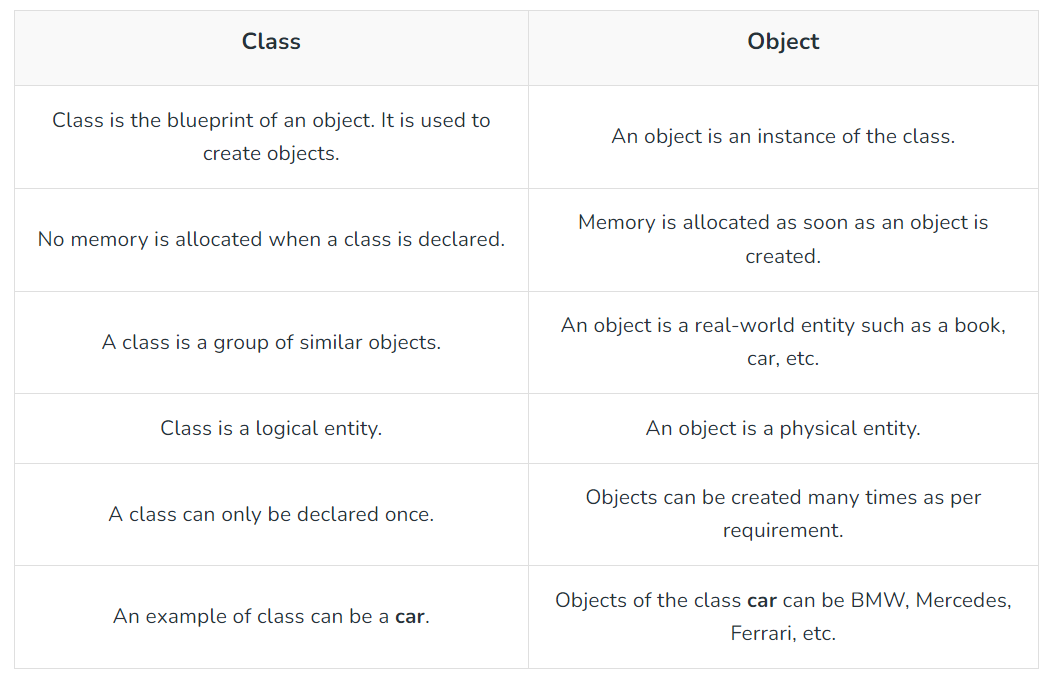
**Classes and Objects in Java**

In Java, **classes and objects**are basic concepts of [Object Oriented Programming (OOPs)](https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/) that are used to represent real-world concepts and entities. The class represents a group of objects having similar properties and behavior. For example, the animal type **Dog** is a class while a particular dog named **Tommy** is an object of the **Dog** class. In this article, we will discuss **Java classes and objects** and how to implement them in our program.

**Difference Between Java Classes and Objects**

The main differences between class and object in Java are as follows:



## Java Classes

A **class in Java** is a set of objects which shares common characteristics and common properties. It is a user-defined blueprint or prototype from which objects are created. For example, Student is a class while a particular student named Ravi is an object.

### Properties of Java Classes

* Class is not a real-world entity. It is just a template or blueprint or prototype from which objects are created.
* Class does not occupy memory.
* Class is a group of variables of different data types and a group of methods.
* A Class in Java can contain:
  + Data member
  + Method
  + Constructor
  + Nested Class
  + Interface

### Class Declaration in Java

access\_modifier***class*** *<*class\_name*>*

*{*

*data member;*

*method;*

*constructor;*

*nested class;*

*interface;*

*}*

### Components of Java Classes

In general, class declarations can include these components, in order:

* **Modifiers**: A class can be public or has default access (Refer [this](https://www.geeksforgeeks.org/access-specifiers-for-classes-or-interfaces-in-java/)for details).

In Java, methods and data members can be encapsulated by the following four access modifiers. The access modifiers are listed according to their restrictiveness order.   
1) **private** (accessible within the class where defined)   
2) **default** or package-private (when no access modifier is specified)   
3) **protected** (accessible only to classes that subclass your class directly within the current or different package)  
4) **public** (accessible from any class)

But, the classes and interfaces themselves can have only two access modifiers when declared outside any other class.   
1) **public**  
2) **default** (when no access modifier is specified)

**Note**: *Nested interfaces and classes can have all access modifiers.*  
**Note**: *We cannot declare class/interface with private or protected access modifiers.*

For example, the following program fails in the compilation.

* **Class keyword:** Class keyword is used to create a class.
* **Class name**: The name should begin with an initial letter (capitalized by convention).
* **Superclass (if any)**: The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
* **Interfaces(if any)**: A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
* **Body**: The class body is surrounded by braces, { }.

Constructors are used for initializing new objects. Fields are variables that provide the state of the class and its objects, and methods are used to implement the behavior of the class and its objects. There are various types of classes that are used in real-time applications such as [nested classes](https://www.geeksforgeeks.org/inner-class-java/), [anonymous classes](https://www.geeksforgeeks.org/anonymous-inner-class-java/) and [lambda expressions](https://www.geeksforgeeks.org/lambda-expressions-java-8/).

*// Java Class example*

**class** **Student** {

*// data member (also instance variable)*

int id;

*// data member (also instance variable)*

String n;

**public** **static** void main(String args[]) {

*// creating an object of*

*// Student*

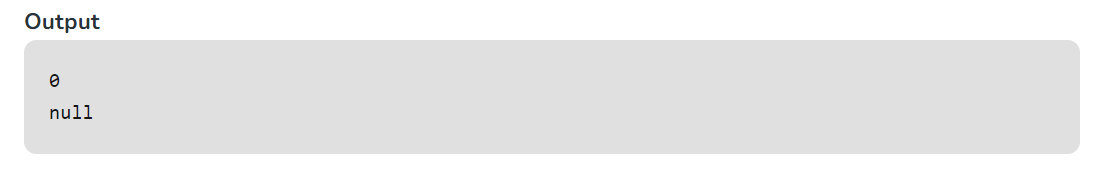
Student s1 = **new** Student();

System.out.println(s1.id);

System.out.println(s1.n);

}

}



**Example 2**: Here, the below Java code demonstrates ***creating an object using the***[***newInstance()***](https://www.geeksforgeeks.org/constructor-newinstance-method-in-java-with-examples/)***method.***

*// Creation of Object*

*// Using new Instance*

**class** **Geeks** {

*// Declaring and initializing string*

String n = "GeeksForGeeks";

*// Main driver method*

**public** **static** void main(String[] args) {

*// Try block to check for exceptions*

**try** {

*// Correcting the class name to match "Geeks"*

Class<?> c = Class.forName("Geeks");

*// Creating an object of the main class using reflection*

Geeks o = (Geeks) c.getDeclaredConstructor().newInstance();

*// Print and display*

System.out.println(o.n);

}

**catch** (ClassNotFoundException e) {

e.printStackTrace();

}

**catch** (InstantiationException e) {

e.printStackTrace();

}

**catch** (IllegalAccessException e) {

e.printStackTrace();

}

**catch** (NoSuchMethodException e) {

e.printStackTrace();

}

**catch** (Exception e) {

e.printStackTrace();

}

}

}