**OOPS Part-3 and Access Modifiers Part-2**

**OOPS Part 3**

**Constructor:**

1. Object creation is not enough, compulsory we should perform initialization then only the object is in a position to provide the response properly.

2. Whenever we are creating an object some piece of the code will be executed automatically to perform initialization of an object this piece of the code is nothing but constructor.

3. Hence the main objective of constructor is to perform initialization of an object.

**Example:**

**class Student**

**{**

**String name;**

**int rollno;**

**Student(String name,int rollno) //Constructor**

**{**

**this.name=name;**

**this.rollno=rollno;**

**}**

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

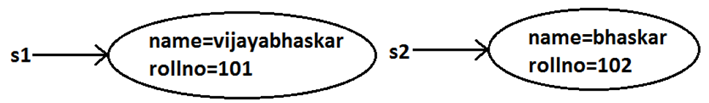
**{**

**Student s1=new Student("vijayabhaskar",101);**

**Student s2=new Student("bhaskar",102);**

**}**

**}**

****

**Rules to write constructors:**

1. Name of the constructor and name of the class must be same.

2. Return type concept is not applicable for constructor even void also by mistake if we are declaring the return type for the constructor we won't get any compile time error and runtime error compiler simply treats it as a method.

**Default constructor:**

1. For every class in java including abstract classes also constructor concept is applicable.

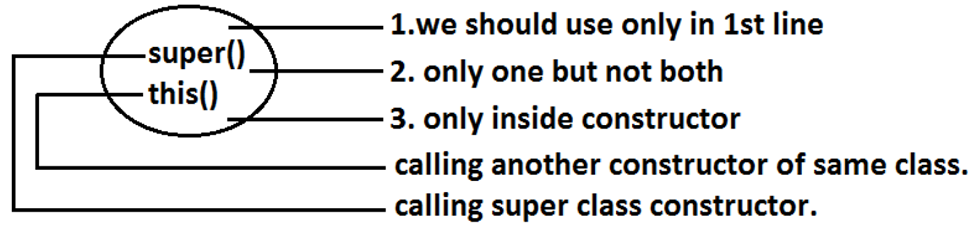
2. If we are not writing at least one constructor then compiler will generate default constructor.

3. If we are writing at least one constructor then compiler won't generate any default constructor. Hence every class contains either compiler generated constructor (or) programmer written constructor but not both simultaneously.

| **Programmers Code** | **Compiler Generated Code** |
| --- | --- |
| class Test { } | class Test {  Test() {  super();  }  } |
| public class Test { } | public class Test {  public Test() {  super();  }  } |
| class Test  void Test() { } | class Test {  Test() {  super();  }  void Test() { }  } |
| class Test {  Test(int i) { }  } | class Test {  Test(int i) {  super();  }  } |
| class Test {  Test() { }  } | class Test {  Test() {  super();  } |
| class Test {  Test(int i) {  this();  }  Test() { }  } | class Test {  Test(int i) {  this();  }  Test() {  super();  } |

**Super() vs this():**

The 1st line inside every constructor should be either super() or this() if we are not writing anything compiler will always generate super().

****

**Overloaded Constructers:**

A class can contain more than one constructor and all these constructors having the same name but different arguments and hence these constructors are considered as overloaded constructors.

**class Test {**

**Test(double d){**

**System.out.println("double-argument constructor");**

**}**

**Test(int i) {**

**this(10.5);**

**System.out.println("int-argument constructor");**

**}**

**Test() {**

**this(10);**

**System.out.println("no-argument constructor");**

**}**

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

**Test t1=new Test(); //no-argument constructor/int-argument**

**//constructor/double-argument constructor**

**Test t2=new Test(10);**

**//int-argument constructor/double-argument constructor**

**Test t3=new Test(10.5);//double-argument constructor**

**}**

**}**

**Note:** Compiler is responsible for the following checkings.

1. Compiler will check whether the programmer wrote any constructor or not. If he didn't write at least one constructor then compiler will generate default constructor.

2. If the programmer wrote any constructor then compiler will check whether he wrote super() or this() in the 1st line or not. If his not writing any of these compiler will always write (generate) super().

**Access Modifiers Part-2**

**Final modifier:**

Final is the modifier applicable for classes, methods and variables.

**Final Method :** Whatever the methods parent has by default available to the child. If the child is not allowed to override any method, that method we have to declare with final in parent class. That is final methods cannot overridden.

**Final Class:** If a class declared as the final then we can’t create the child class that is inheritanceconcept is not applicable for final classes.

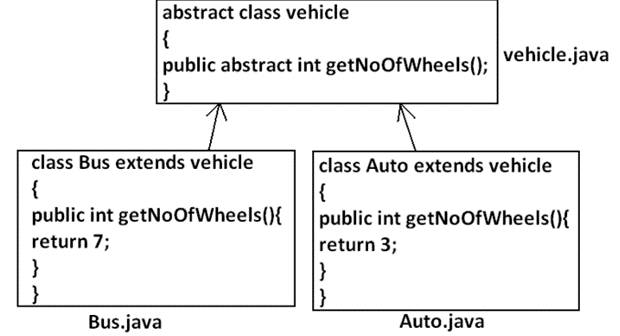
**Final variable:** When the final keyword is used with a variable of primitive data types such as int, float, etc), the value of the variable cannot be changed.

**Abstract modifier:**

Abstract is the modifier applicable only for methods and classes but not for variables.

**Abstract Method :** Even though we don't have implementation still we can declare a method with abstract modifier.

That is abstract methods have only declaration but not implementation. Hence abstract method declaration should compulsory ends with semicolon.

****

**Abstract Class:** For any java class if we are not allow to create an object such type of class we have to declare with abstract modifier that is for abstract class instantiation is not possible.

**What is the difference between abstract class and abstract method ?**If a class contain at least on abstract method then compulsory the correspondingclass should be declare with abstract modifier. Because implementation is notcomplete and hence we can't create object of that class. Even though class doesn't contain any abstract methods still we can declare the class as abstract that is an abstract class can contain zero no of abstract methodsalso.