Default Constructor

This is a constructor with no parameters. If you don't define a constructor for a class, it is automatically created by the compiler. However, if you define any constructor for your class, a default constructor is no longer automatically created.

Parameterized Constructor

Method overloading is supported by Java and constructor is a special method. So, can add another constructor to the class which will assign user-given values to the members. The job of the parameterized constructor is to assign specific values to members.

this keyword

The object which invokes a method is called the implicit or hidden object. Sometimes a method may need to refer to this implicit object. To allow this, Java defines the **this** keyword. The **this** keyword always refers to the current or implicit object.

Uses of this:

1. **To resolve ambiguity between a local and instance variable:** When the local variable has the same name as the instance variable, the local variable hides the instance variable. Hence, this can be used to specify instance variables.
2. **To return the implicit object:** In some cases, a method may want to return the implicit object itself. In such a case, this can be returned.
3. **To invoke methods and access instance members:** Instance members and methods can be accessed using "this" inside a method.
4. **To invoke one constructor within another:** From within a constructor, you can also use the **this** keyword to call another constructor in the same class. Doing so is called an *explicit constructor invocation*.

Chaining of constructor in same class

```
class Test
{
    int a;
    Test()
    {
          this(0);  //invoke constructor
    }
    Test(int a)
    {
          this.a = a; //resolve name conflict
    }
    Test modify()
    {
          this.a = 100 ; //access instance member
```

```
            return this; // return implicit object
        }
        void display()
        {
        System.out.println("Value of a = " + a);
        }
        public static void main(String[] args)
        {
            Test obj = new Test();
            obj.display(); // Value of a = 0
            obj.modify();
            obj.display(); // Value of a = 100
        }
}
```