***Constructor Declarations***

* A **class** contains **constructors** that are invoked to create objects from of this **class**.
* Constructor declarations look like method declarations—except that they use the name of the class and have no return type.
* The compiler automatically provides a no-argument, default constructor for any class without constructors. This default constructor will call the no-argument constructor of the superclass. In this situation, the compiler will complain if the superclass doesn't have a no-argument constructor so you must verify that it does. If your class has no explicit superclass, then it has an implicit superclass of Object, which *does* have a no-argument constructor.
* A constructor can’t be defined as it should be defined as public
* Constructors can also take paramaters, which is used to initialize attributes that we will be able to create objects from this **class**

***Encapsulation***

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit.

In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.

**Encapsulation** is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java :

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

The public setXXX() and getXXX() methods are the access points of the instance variables of the EncapTest class. Normally, these methods are referred as getters and setters. Therefore, any class that wants to access the variables should access them through these getters and setters.