Chapter 6: Java Object Oriented Programming

Classes and Objects:

- Class : ➔ a template that describes the state and behavior.
- Object : ➔ at runtime, when JVM encounters the new keyword, it will use the appropriate class to make an object which is instance of that class. That object will have its own state and access to all of the behaviors defined by its class.

State and Behavior:

State (instance variables) : ➔

- **Each object** (instance of a class) will have its **own unique set of instance variable** as defined in the **class**.
- Collectively, the values assigned to an objects instances variables make up the **objects state**.

Behavior (methods): ➔

- When a programmer creates a class, programmer creates methods for that class.
- Methods are where the **class logic** is stored.
- Methods are where the **real work** gets done.
- They are where **algorithms get executed** and **data gets manipulated**.

Encapsulation:

If we want **maintainability**, **flexibility** and **extensibility** our design must include encapsulation by:

- Keep **instance variable private**.
- Keep the **accessor methods public**.

Access Modifiers:

public members:

All classes can access all public class members (methods or variables) of a class.

private members:

The private class members can **only** be accessed **by the class** in which they are declared.

default members:

The default class members can be accessed when the declaring and accessing class belongs to the **same package**.

protected members:

The protected class members can be accessed by the class that **extends the class** (i.e. subclass) in the **same package or different package**.

Instance Variables v/s Local Variables:

Instance Variables	Local Variables
They get default values	They don't get any default values.
They can use all access modifiers	They cannot use access modifiers

Method Overloading:

- In method overloading, we can create methods that have the **same name**, but the methods differ in the **type , sequence and/or number** of parameters.
- When an overloaded method is invoked, Java uses the type and/or numbers of arguments as its guide to determine which version of the overloaded method to actually call.

Rules for Method Overloading:

1. Overloaded methods **MUST** change the argument list. (number, data type and / or sequence of parameters).
2. Overloaded methods **CAN** change the return type.
3. Overloaded methods **CAN** change the access modifier.

Constructors:

- Constructor is a special method which is used for initialization of objects.
- Constructor method has same name as that of the class.
- Constructor method will not have any return type.
- Constructor cannot be called explicitly (its always implicit)
- Constructor method can be declared with and without parameter.
  - **Zero Argument Constructor:** Constructor that does not take as input any parameters is called zero argument constructor.
  - **Parameterized Constructor:** Constructor that takes as input some parameters is called parameterized constructor.
  - **Constructor Overloading:** Defining constructors with different parameters is called constructor overloading.

Rules for Using Constructors:

1) If the class does not contain any constructor then compiler will add a DC in the class.
2) If the class contains any constructor then compiler will not add anything.
3) Constructor can have all access **modifiers** (private, public protected and default).
4). Constructor cannot be declared final, static or abstract.