**Exercise #1**

**1-Encapsulation:**

Encapsulation is wrapping data and behaviours in a single unit in such a way that data is hidden from the outside world and can only be accessed by the behaviours.

It is done by access modifiers i.e. keep data private and setter/getter/other methods public.

**Example:**

public class A {

private int data1;

public int getData () {

return data1;

}

}

In the above example, the class’s data (variable) i.e.: data1 is kept private and a ‘getData’ method is kept public so it is accessed.

**2-Inheritance:**

Inheritance is defining one class in terms of other in such a way that parent class (super class) data and behaviours are reused in child class (sub class).

It’s Is-A relationship.

**Example:**

public class A

{

private int a;

private int b;

public A () {

this.a= 0; this.b = 0;

}

}

public class B extends A

{

private int c;

public B () {

super();

this.c = 0;

}

}

public class InheritanceTest {

public static void main (String [] args) {

B Obj = new B (5, 5, 5);

}

}

In the above example, the ‘B’ (child) class is inherited from the ‘A’ (parent) class.

**3-Polymorphism:**

Polymorphism is the ability of an object to take on many forms in such a way that correct behaviour is called at runtime.

It is done by overriding of virtual methods and defining of abstract methods.

**Example:**

public class A {

public void func(){

System.out.println("I’m from class A");

}

}

public class B extends A {

public void func(){

System.out.println("I’m from class B");

}

}

public class PolymorphismTest {

public static void main (String [] args) {

A a=new B();

a.func();

}

}

In the above example, the object of ‘A’ class is created, calling the constructor b ‘B’ class. The behaviour is decided at runtime and when ‘func’ is called, “I’m from B class” will print.

**4-Abstraction:**

Abstraction is hiding the implementation and showing only the functionality to child classes in such a way that child classes are made necessary to implement the functionality.

It is done by abstract classes (0-100%) and interfaces (100%).

By abstraction generalising and specialising the objects is achieved.

**Example:**

Public abstract class A {

public abstract int getPosition();

}

public class B extends A {

public int getPosition () {

return 2;

}

}

public class C extends A {

public int getPosition () {

return 3;

}

}

public class AbstractionTest {

public static void main (String [] args) {

B b = new B();

C c = new C();

System.out.println(“B : ” + b.getPosition() “, C : ” + c.getPosition());

}

}

In the above example, the ‘A’ class has an abstract functioni.e.:getPosition(). Both its child classes (B & C) have implemented (must) that function.