# PRE-ASSESSMENT PACK

Define the following key Object Orientated Programming (OOP) principles with examples:

**• Encapsulation**

**• Inheritance**

**• Polymorphism**

**• Abstraction**

Encapsulation

Encapsulation is one of the four essential object oriented programming concepts.

It is is a method of wrapping the variables and methods as a single unit. Encapsulation is also known as data hiding; as in encapsulation, the variables of a class will be hidden from other classes and can only be accessed through the methods of their current class. In order to achieve encapsulation in java, the user will have to declare the variables in a class as private and provide getter and setter methods to modify and view the variable values.

An example of encapsulation is shown below:

public class Show {

private String showName;

private Cinema cinema;

public Show(String showName, Cinema cinema)

{

this.showName = showName;

this.cinema = cinema;

}

public void setShowName(String showName)

{

this.showName = showName;

}

public void setCinema( Cinema cinema)

{

this.cinema = cinema;

}

Inheritance

Inheritance is a process where one class develops the methods and fields of another. The class that inherits the properties of another is known as a subclass and the class whose properties are inherited is known as a superclass. The keyword **extends** is used to inherit the properties of a class.

An example of inheritance is shown below:

Public class Person

{

private string name;

private int birthDate;

public Person (string name, int birthD);

{

name =n;

birthDate = birthD;

}

public class Student extends Person

{

private String top;

public Student (string n, int birthD, String top)

{

super (n, birthD);

top = t;

}

Polymorphism

The most common use of polymorphism in object-oriented programming occurs when a parent class reference is used to a child class object. The only possible way to access an object is through a reference variable and the reference variable can be of only one type.

The reference variable can be reassigned to other objects as well if they are not declared final. A reference variable can be declared as a class or an interface type.

An example of polymorphism is shown below:

Public class Teacher {

private string name;

private string class;

Public Teacher (string name, string class) {

System.out.println(“Teacher”)

this.name = name;

this.class = class;

}

**2nd class**

Public class Student extends Teacher {

Private int classNumber;

Public Student (String name, String class, int classNumber) {

Super (name, class);

setStudent (classNumber);

}

Abstraction

Abstraction in object-oriented programming is a procedure of hiding the implementation details from the user. The user will only have the information on what the object does instead of how it does it. Abstraction is achieved using abstract classes and interfaces. If a class contains the keyword **abstract** in its declaration, it is an abstract class.

To use an abstract class, it is necessary to inherit it from another class, provide implementations to the abstract methods in it.

An example of abstraction is shown below:

Public class Main {

Public static void main (String args[]) {

FourWheeler test = new Mercedes();

Test run();

}

}

abstract class FourWheeler {

public abstract void run();

}

Class Mercedes extends FourWheeler {

Public void run() {

System.out.println(“Working”)

}

}