Assignment 11

① Explain the following with Example.

ⓐ Inheritance [IS-A relationship (parent child)]

↳ where one class allows to inherit the features of another class. means one object acquires all the properties & behaviours of a parent object.

↳ The idea behind inheritance in Java is that you can create new classes that are built upon existing classes.

↳ when you inherit from an existing class you can reuse methods & fields of the parent class.

(Ex)
```
class Employee {
    float Salary = 40000; }
class Programmer extends Employee {
    int bonus = 10000;
    Public static void main (String args[]) {
        Programmer P = new Programmer();
        S.O.P(P.salary);
        S.O.P (P.bonus);
    }
}
```

O/P: 40000
10000

Adv: ① Method overriding can be achieved.

② Code reusability

⑥ Abstraction

↳ data Abstraction is the property by virtue of which only the essential details are displayed to the user.

↳ Abstration is achieved by Interfaces & abstract classes.

Ex: A car is viewed as a car rather than its individual components

↳ Abstract class and methods

1) the class declared using abstract keyword is called abstract class.

2) the method declared without implementation is called abstract method. & must be redefined in the subclass, making overriding is compulsory.

(or) either make subclass itself abstract.

3). Any class that contains one or more abstract methods must also be declared with abstract keyword.

Adv: ① It reduces the complexity of viewing the things

② avoids code duplication & increases reusability.

③ helps to increase security of an application & program as only important details are provided to the user.

Ex) Abstract class ~~Shape~~ Base {

~~String color~~;

abstract ~~double area~~ void fun1();

}

```
Derived
class Circle extends Shape Base.
{
    void fun1() {
    S.O.P ("derived fun1() called"); }
}
class Test {
PSVM(...) {
Base b = new Derived();
    b.fun1();
}
}
```

O/p: derived fun() called.

(c) Polymorphism

↳ In Java it is the ability of an object to take many forms means single thing (task) done in many form.

(d) Encapsulation

↳ In Java is a mechanism of wrapping the data (variable) & code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, & can be accessed only through the methods of their current class. So it is calling as data hiding.

To achieve it : (1) declare the variables of a class as private.

(2) Provide public setter & getter methods to modify & view the variable values.

(6) class Test {

Private String name;

Public String getName {

return name; }

Public void setName (String name)

{ this.name = name; } }

class Main {

PSVM (...) {

Test t1 = new Test ();

t1.setName ("Archu");

S.O.P (t1.getName ());

} }

O/P : Archu.

(7) Explain difference b/w Method Overloading & Overriding

(*) Method Overloading

↳ We can create multiple methods of the same name in the same class & all methods works in different ways.

Two ways to overload method

(1) by changing number of arguments

(2) by changing the data type.

Ex ① changing no/ of argument

```
class Adder {
  static int add (int a, int b) {
      return a+b; }
  static int add (int a, int b, int c)
      { return a+b+c; }

class Test {
  psvm (...)
  S.O.P ( Adder.add (11,11)); // 22
  S.O.P ( Adder.add (11,11,11)); // 33
  }
  }
```

Ex ② changing data type of arguments

```
class Adder {
  static int add ( int a, int b ) {
      return a+b; }
  static double add (double a, double b)
      { return a+b; }

class Test {
  psvm(...) {
  S.O.P ( Adder.add ( 11,11)); // 22
  S.O.P (Adder.add (11.5, 11.5)); // 22.10
  } }
```

③ Method Overriding

↳ If Subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java

rules: ① method must have the same name & same parameter as in the parent class.

② there must be an IS-A relationship

Ex:
```
class Vehicle {
    void run() {
    S.O.P ("vehicle is running"); }
    }
class Bike extends Vehicle {
    PSVM (.....) {
    Bike obj = new Bike();
        obj.run(); // Vehicle → Bike is running
    }
}
```
```
void run() {
    S.O.P ("Bike is running");
}
```

③ multiple (class) inheritance is not allowed in Java Explain. Also explain how it supports multiple inheritance through interfaces.

→ Java does not support multiple inheritance bec/ of two reasons:

① In Java, every class is a child of object class. when it inherits from more than one super class, sub gets the ambiguity to acquire the prop-erty of object class.

② In Java every class has a constructor, if we write it explicitly or not at all. the first stmt is calling super() to invoke the super class constructor, if the class has more than one super class, it gets confused

So when one class extends from more than one super class, we get compile time error.

↳ The only way to implement multiple inheritance is to implements multiple interfaces in a class. In java one class can implements two or more interfaces.

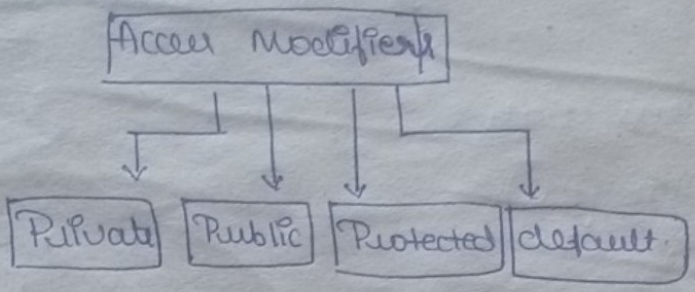
(Ex)
```
interface Test1 {
    Public void show();
}
interface Test2 {
    Public void show();
}
class Subclass implements Test1, Test2 {
    Public void show() {
    S.O.P("A class can implements more than one class");
    }
    PSVM(...) {
    Subclass obj = new Subclass();
        obj.show();
    }
}
```

(4) difference b/w abstract & interfaces?

Abstract class	Interfaces
(1) have abstract & non-abstract methods	(1) have only abstract methods. Since Java 8, it can have default & static methods.
(2) doesn't support multiple inheritance.	(2) support multiple inheri-tance.
(3) can have final, non-final, stati-c & non-static variable.	(3) has only static & final variable.

(4) Can provide the Implement -ation of Interface	(4) Can't provide the Implementation of abstract class
(5) The abstract keyword is used to declare abstract class	(5) the Interface keyword is used to declare Inter -face
(6) Can extend another Java class & Implement multiple Java Interfaces	(6) Can extend another Java Interface only.
(7) Can be extended using keyword "extends"	(7) Implemented using keyword "Implements".
(8) It can have class members like private, protected, etc	(8) Members are public by default.
(9) (Ex) public abstract class Shape { public abstract void draw(); }	(9) (Ex) public Interface Drawable { void draw(); }

Access Modifiers in Java



(1) Private → accessible within the class only.

↳ The methods & data members declared as Private are accessible only within the class in which they are declared.

↳ ~~int~~ class & interface can not be declared as Private.

Ex:- Package A1;

Class A {

Private void display()

{ S.O.P("Simple Snippets"); }

}

```
class B
{
    PSum(...)
    {
        A obj = new A();
        obj.display(); }
}
```

O/P: error: display() has private access in A.
obj.display();

② default Access Modifier

↳ these are accessible only within package

↳ The data members, class & methods which are not declared using any access modifiers i.e having default access mod/ are accessible only within the same package.

③ Protected

↳ The protected access mod/ is accessible within Package & outside the package but through inheritance only.

→ It can be applied on the data member, method & constructor.

→ It can't be applied for class.

RMP

Ex: Package mypack1;

```
Public class MySupperclass {
    Protected int x;
    Protected void showX()
    {
        S.O.P ("Value of x is:" +x);
    } }
```

Another Package:

```
Package JavaApplication1;
import mypack1.MySupperclass; imp
Public class JavaApplication1 extends MySuperClass{
PSVM(...)
{
    JavaApplication1 obj = new JavaApplication1();
    obj.x = 5;
    obj.showX();
} }
```

O/P: Value of x is: 5

④ Public: accesible everywhere in the Program.

Ex:

```
Package A1;
Public class A
{
Public void display()
{
S.O.P ("Simple Snippets")
} }
```

```
Package P2;
import P1.*;
class B.                    // outside the Package.
{
  PSVm( . . )
  {  A obj = new A ;
     obj . display ();
  }
}
```

Acces modifier	within class	within Package	Outside package by Subclass only	outside Package.
Private	Y	N	N	N
default	Y	Y	N	N
Protected	Y	Y	Y	N
Public.	Y	Y	Y	Y

(*) Polymorphism in Java (multiple forms).

↳ Perform single task in different forms. Note:

2 types: (1) compile time (Static) (overloading)

(2) Run time (dynamic) (Overriding)

(1) Run time Poly : (dynamic method dispatch) is a process in which a call to an overridden method is resolved at run-time rather than compile time.

(*) In this process an overridden method is called through the reference variable of a superclass.

Thus this happen only

Ex: Method overriding

```
class Parent
{
    void show()
    {
        S.O.P("Parent's show()");
    }
class child extends Parent {
    {
        @override
        void show()
        {
            S.O.P("child's show()");
        }
    }
class Main.
{
    psvm(...)
    {
        Parent obj1 = new Parent();
        obj1.show();

        Parent obj2 = new child();
        obj2.show();
    }
}
```

reference

O/P: Parent's show
child's show.

Note: ① final methods, static methods, Private methods
: can't be overridden → constructor.

The overriding method must have same return type (or subtype).

3) Overriding & Access Modifier:

↳ The AM for an overriding method can allow more, but not less, access than the overridden method. For Example, a protected instance method in the Super class can be made public, but not private in the Subclass.

4) Overriding and Constructor:

↳ We can not override constructor as parent & child class can never have constructor with same name.

5) Overriding & abstract method:

→ Abstract methods in an Interface or abstract class are meant to be overridden in derived concrete classes otherwise compile time error will be thrown.

6) Invoking overridden method from sub-class.

→ We can call Parent class method in overriding method using Super keyword.