CREATED BY:
Muhammad Bilal
Arslan Ahmad
Shaad

JAVA CHAPTER NO 5

Instructor:
Muhammad Naveed

Chapter No 5

Object Oriented Programming

Q: Explain subclass and inheritance?

In object-oriented programming, inheritance enables new objects to take on the properties of existing objects. A class that is used as the basis for inheritance is called a superclass or base class. A class that inherits from a superclass is called a subclass or derived class. The terms parent class and child class are also acceptable terms to use respectively. A child inherits visible properties and methods from its parent while adding additional properties and methods of its own.

Subclasses and super classes can be understood in terms of the is a relationship. A subclass is a more specific instance of a superclass. For example, an orange **is a** citrus fruit, which **is a** fruit. A shepherd **is a** dog, which **is an** animal.

Here is a Chocolate class below(as an example):

```
package cakes {
  public class ChocolateCake {


    public function ChocolateCake(){}


    public function bake():void{
      trace("The chocolate cake is baking");
    }


    public function frost():void{
      trace("The chocolate cake now has frosting");
    }
```

```
        public function putCandlesOnCake(numberOfCandles:int):void{
          trace("Putting " + numberOfCandles + " candles on the cake. ");
        }
    }
}
```

It's a simple class with three methods: bake(), frost(), and putCandlesOnCake().

Q: What is the use of extend keyword?

A subclass can inherit all data members and methods of superclass using extend keyword.

Inheritance is a major feature of OPP (object oriented programming). It allows us to reuse the code. The class which is reused is called parent class. User class is called child class.

We use the keyword extend to define a child class that uses a parent class.

Syntax:

```
Class super {
              // code
             }
Class sub extend super {
              // code
              }
```

Examples:

Superclass	Subclass
Student	Graduate Students, Undergraduate Student.
Shape	Circle, Rectangle, Triangle.
Quadrilateral	Square, Rectangle.
Employee	Faculty member, Staff members.

“**Is a**” relationship represents inheritance.

Object of subclass is an object of superclass.

Q: Define protected?

Protected data members or methods is accessible only from within its package.

Other classes can be used by inheritance only subclass can use protected data member or methods.

Q: What is overloading?

Defining multiple methods with same name but different signatures is known as overloading.

Overloading methods must have different parameters and they may have different return type.

Example:

```
Class circle {
Static int Add(int a, int b)
{
 return a+b;
}
Static int Add (int x, int y, int z)
{
 return x+y+z;
 }
}
```

Q: What is overriding?

When a method of super class is redefined in a subclass with same name and signature but different implementation details, this is known as overriding.

⇨ **Rules regarding to overriding methods:**

1. Methods that are static, final and private cannot be overridden.
2. Protected method can override only these methods that don't have any access modifier.
3. Overriding method cannot have a more restricted access than the original method.
4. The overriding method cannot throw any new exception.

Example:

```
Class Animal {
   public void move()
{
 System.out.println("Animal Moves")
   }
}
```

Q: What is access control?

Access modifiers define the access privileges of class, interface, constructors, methods and data members.

Access modifiers consist of:

Public.

Private.

Protected.

If no modifier is present then default access is package private.

Q: What is composition?

Composition is a design technique to represent ‘has a’ relationship.

- ⇨ Car has an engine.
- ⇨ Person has a Job.

Using inheritance:

- ⇨ Use keyword ‘extend’.
- ⇨ Private data members of superclass are not directly accessible to subclass.

Example:

```
Class point {
        protected int x, y;        ──────▶   Data Members
        public point()
          {
            x=0;
            y=0;
          }
public point (int a, int b)                   ──────▶   Constructor
  {
    x=a;
    y=b;
  }
public void setpoint (int a, int b)
     x=a;
     y=b;                                     ──────▶   Methods
  }
}
```

```
public class Circle extends point{
        protected double rad;
   public circle ()
     {
        Setradius(0);
     }
public void SetRadius (double r)
 {
      Rad=;
}
public double area ()
  {
    return 3.2438*rad*rad;
  }
}
```

Using extend keyword getting the radius of circle

Q: Explain polymorphism?

Polymorphism in java is a concept by which we can perform a single action by different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

If you overload static method in java, it is the example of compile time polymorphism.

Q Define runtime polymorphism?

Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

Q: What is encapsulation?

Encapsulation is one of the fourth fundamental concept of object oriented programming.

Encapsulation in java is a mechanism of wrapping the data (variables) and code acting on the data 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. Therefore it is also known as data hiding.

Declare variables of a class as private.

Provide public setter and getter methods to modify and view the variable values.

- ⇨ The field's of a class can be made read only or write only.
- ⇨ Class can have total control over what is stored in its fields.
- ⇨ The users of a class do not know how the class stores its data.
- ⇨ A class can change the data type of field and user if class do not need to change any of their code.

Example:

```
public class Car{

private String name;

private double topspeed;

public Car( ) { }

public String getName( ) {

return name;

}
public void setName (String name) {
this.name=name;
}
public void setTopSpeed(double SpeedMPH){
Topspeed = SpeedMPH;
}
public double getTopSpeedMPH( ) {
return TopSpeed;
}
public double getTopSpeedKMH( ) {
return TopSpeed*1.609344;
  }
}
```

CHAPTER ENDED