# ASSIGNMENT 5 : LAB 4

**QUESTION 1** - Fun with Polymorphism

Create a hierarchy of shapes with the classes Shape, Circle, Rectangle and Square. (Hint: Square is a special Rectangle whose length=breadth)

Each class has its own printDescription() method which prints its dimensions and area.

Create a ShapeManager class which instantiates the various shapes and calls for the description of each shape using the method manageShape(Shape).

```
void manageShape(Shape myShape)  {

   myShape.printDescription();

}
```
The constructors of the other classes are as  follows:

- Circle(double radius)
- Rectangle(double length, double breadth)
- Square(double side)

Declare the constructor of each class with a print statement.

**SOURCE CODE :**

File :- Shape.java

```
/*
 * @author harsh
 */


public class Shape {
   public void Shape()
   {
      System.out.println("IT CALLS A DEFAULT");
   }
   void printDescription()
```

```
  {

      System.out.println("I AM A SHAPE\n");

  }
}
```

File : Rectangle.java

```
public class Rectangle extends Shape {
  private double length;
  private double breadth;
  double area;

  public Rectangle(){

      System.out.println("Calling Rectangle Default Constructor");
  }

  public Rectangle(double length,double breadth){

      this.length=length;
      this.breadth=breadth;
      area=length*breadth;

  }
  @Override
      public void printDescription(){

      System.out.println("I AM A RECTANGLE");
```

```
         System.out.println("Area of Rectangle= " +getArea());


         System.out.print("and also,");
        super.printDescription();



    }


    public double getArea() {
        area=length*breadth;
        return area;


    }
}
```

File :- Square.java

```
public class Square extends Rectangle {
    private double side;
    //private double area;


    public Square(){
        System.out.println("Calling Square Default Constructor");
    }


    public Square(double side){
        super(side, side);



        this.side=side;
        //area
        //System.out.println("Calling Square Default Constructor");
        area=side*side;



    }
    @Override
        public void printDescription(){
        System.out.println("I AM A SQUARE");


          System.out.println("Area of Square= "+area);
             System.out.print("and also,");
        super.printDescription();


    }
}
```

File:- Circle.java

```
public class Circle extends Shape {
    private double radius;
    private double area;
    public Circle(){
        System.out.println("Calling Circle Default Constructor");
    }




    public Circle(double radius){
        this.radius=radius;
        area=22/7*radius*radius;



    }
```

```
   @Override
      public void printDescription(){



      System.out.println("I AM A CIRCLE");
         System.out.println("Area of Circle = "+area);
         System.out.print("and also,");
         super.printDescription();
      }
}
```

File :- ShapeManager.java

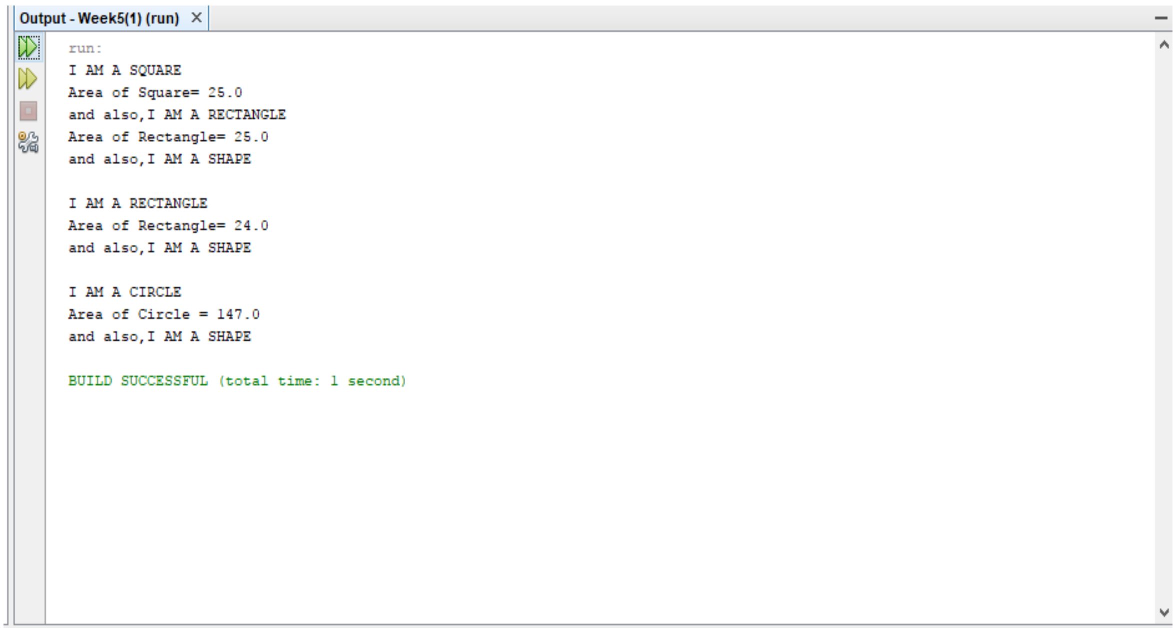
```
public class ShapeManager {
   public static void main(String args[]){
   Shape myShape = new Shape();
   ShapeManager pg=new ShapeManager();

   Rectangle myRect = new Rectangle(4,6);

  Square mySquare = new Square(5);
    Circle myCircle= new Circle(7);
   pg.manageshape(mySquare);
   pg.manageshape(myRect);
   pg.manageshape(myCircle);


   }
   public void manageshape(Shape myShape)
   {
    myShape.printDescription();
   }
   }
```

**OUTPUT:**



```
run:
I AM A SQUARE
Area of Square= 25.0
and also,I AM A RECTANGLE
Area of Rectangle= 25.0
and also,I AM A SHAPE

I AM A RECTANGLE
Area of Rectangle= 24.0
and also,I AM A SHAPE

I AM A CIRCLE
Area of Circle = 147.0
and also,I AM A SHAPE

BUILD SUCCESSFUL (total time: 1 second)
```

1. Is there any compile time or run-time error when a Circle, Rectangle or Square instance is given as parameter to the manageShape method? Explain.

Answer: No, there is no error.

There is no error because Circle, Rectangle and Square inherits from the class Shape. So, they inherit Shape's properties as well. Hence we can easily pass them.

2. Why does the Circle, Rectangle and Square instances print the correct description, even though the input parameter is declared to be of type Shape? Explain.

Answer: This is because Circle, Rectangle and Square, all are a type of Shape. They inherit the properties of Shape, so basically they are a Shape too.

**3.** How can an instance of Square print the description "I am a Rectangle of length x and breadth x. My area is ...”?

Answer: This can be done by calling the immediate parent class of Square using the keyword super in the printDescription() method of Square class. The syntax will be super.printDescription();

4. In the following code snippet, explain why Line#2 gives an error (what is the error?).

Object myObj = new Rectangle(...);

Rectangle myRect = (Rectangle)myObj; // Line#1

Square mySq = (Square)myObj; // Line #2

Answer: The error is:”Rectangle cannot be cast to Square”. A Square is a type of Rectangle but not the vice-versa.

5. Describe and explain the output from the following code snippet
Square mySq = new Square(...)

Rectangle myRect = (Rectangle)mySq;

myRect.printDescription();

Answer: It calls the default constructor of Square class and then passes to Rectangle class with the value passed into the parameterized constructor of Sqaure and prints the area of the square.

6. Trace how Java invokes the constructors in an inheritance hierarchy when you call the constructor Square(..)

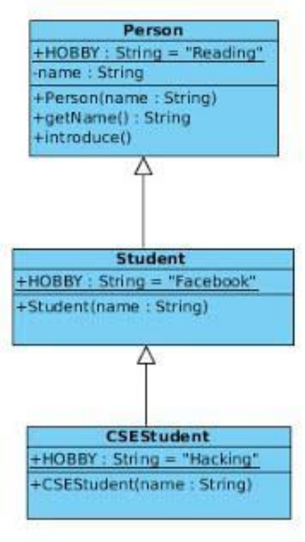
Answer: Default Constructor of Shape is invoked

Parameterised Constructor of Rectangle is invoked

Parameterised Constructor of Square is invoked

**Question 2:**

Construct the following inheritance hierarchy.
Arif is a CSE Student at IEM, who secretly moonlights as a hacker. Have Arif introduce himself
(1) at a get-together for student leaders of various colleges in Salt Lake
(2) at a closed-door Hacker Society meeting
(3) at his cousin's birthday party where he meets a beautiful girl who is a Tagore fan.



**SOURCE CODE :**

**PERSON**

```
import java.util.Scanner;
/**
*
* @author Harsh
*/
public class Person
{
  public static String hobby="Reading";
  private static String name;
  public Person(String name)
  {
     this.name=name;
  }
  public String getname()
  {
     return name;
  }
  public void introduce()
  {
     System.out.print("Hello, my name is "+getname()+" and my Hobby is ");
  }
}
```

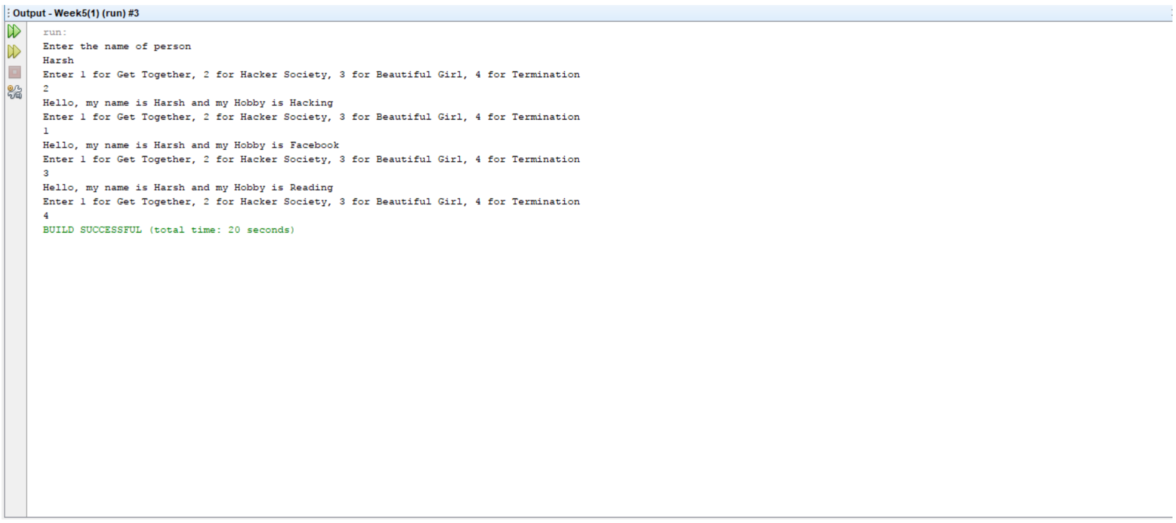
**STUDENT**

```
import java.util.Scanner;
/**
*
* @author Harsh
*/
public class Student extends Person
{
  public static String hobby="Facebook";
  public Student(String name)
  {
     super(name) ;  //Gets the value of name from parent class
  }
  public void introduce()
  {
     System.out.print("Hello, my name is "+getname()+" and my Hobby is ");
  }
}
```

**CSESTUDENT**

```
import java.util.Scanner;
/**
*
* @author Harsh
*/
public class CSEStudent extends Student
{
  public static String hobby="Hacking";
  public CSEStudent(String name)
  {
     super(name);    //Gets the value of name from parent class
  }
  public void introduce()
  {
     System.out.print("Hello, my name is "+getname()+" and my Hobby is ");
  }
  public static void main(String[] args)
  {
     int flag=0,ch;
     String n;
     Scanner sc=new Scanner(System.in);
     System.out.println("Enter the name of person");
     n=sc.next();
     CSEStudent cse=new CSEStudent(n); //Creates object of own class
     Student s=(Student)cse; //Creates object of Student class
     Person p=(Person)cse;  //Creates object of Person class
     while(flag==0)
     {
        System.out.println("Enter 1 for Get Together, 2 for Hacker Society, 3 for Beautiful Girl, 4 for
Termination");
        ch=sc.nextInt();
        switch(ch)
        {
          case 1:
            s.introduce();  //Introduces from Student class
            System.out.println(s.hobby);
            break;
          case 2:
            cse.introduce();  //Introduces from CSEStudent class
            System.out.println(cse.hobby);
            break;
          case 3:
            p.introduce();  //Introduces from Person class
            System.out.println(p.hobby);
            break;
          case 4:   //Termination
            flag=1;
            break;
          default:  //Invalid cases
            System.out.println("Invalid Choice. Please Try Again");
        }
      }
    }
}
```

**Output:**



Output - Week5(1) (run) #3

```
run:
Enter the name of person
Harsh
Enter 1 for Get Together, 2 for Hacker Society, 3 for Beautiful Girl, 4 for Termination
2
Hello, my name is Harsh and my Hobby is Hacking
Enter 1 for Get Together, 2 for Hacker Society, 3 for Beautiful Girl, 4 for Termination
1
Hello, my name is Harsh and my Hobby is Facebook
Enter 1 for Get Together, 2 for Hacker Society, 3 for Beautiful Girl, 4 for Termination
3
Hello, my name is Harsh and my Hobby is Reading
Enter 1 for Get Together, 2 for Hacker Society, 3 for Beautiful Girl, 4 for Termination
4
BUILD SUCCESSFUL (total time: 20 seconds)
```

1. What happens when the method introduce() is declared as "final" in the Person class? Explain.

   Answer**:** We get compile-time error. This is because the introduce() function in the other two classes are non-final and hence they cannot override the introduce() function in Person class.

2. What happens when the String constant HOBBY is declared as "final" in the Person class? Explain.

   Answer: Since the variable hobby is declared final inside the Person class, its value won't get changed inside that class. But it will be changed in the other two child class. Hence the program gives the same output, without any errors.

3. If the attribute name were declared with package accessibility (#) in the Person class, how an you access it from the child-classes. Can you do the same when name is declared as private? Which is the better way? Explain.

   Answer**:** If the name was in a different package, we had to import that package from the child classes where the name attribute was used. But if name is declared private and present in a different package, then it cannot be accessed by other classes outside that package or even outside that class. So, the better way is to declare it in a different package.