Gokul Jayan
20BCE1249

Lab Exercise on Interfaces

Question 1

Assume that there is a part in a machine having three side measurements s1, s2, s3. Its inner and outer volumes are found using the following formulae:
inner volume = 1/3 π s1*s2*s3
outer volume = 4/3 π s1*s2*s3

Define an interface **volume** which has two methods innerVolume and outerVolume. Define a class **Part** which implements this interface, having required attributes and methods, with suitable constructor.
The show() method is used to display all the attributes of the Part class.

CODE:

```
import java.util.Scanner;

interface volume
{
   void innerVolume();
   void outerVolume();
}

class Part implements volume
{

   double s1, s2, s3, inVol, outVol;
   Part()
   {
      s1=0;
      s2=0;
      s3=0;
      inVol=0;
      outVol=0;
   }
   public void set()
   {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter side-1 measurement:");
      s1=in.nextDouble();
      System.out.print("Enter side-2 measurement:");
      s2=in.nextDouble();
      System.out.print("Enter side-3 measurement:");
      s3=in.nextDouble();
   }
   public void innerVolume()
   {
      inVol=(3.14*s1*s2*s3)/3;
   }
   public void outerVolume()
   {
      outVol=(4*3.14*s1*s2*s3)/3;
   }
```

```
    public void show()
    {
        System.out.println("\nPart: ");
        System.out.println("Side-1: "+s1);
        System.out.println("Side-2: "+s2);
        System.out.println("Side-3: "+s3);
        System.out.println("Inner Volume: "+inVol);
        System.out.println("Outer Volume: "+outVol);
    }
}

public class q1
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Part p= new Part();
        p.set();
        p.innerVolume();
        p.outerVolume();
        p.show();
    }
}
```

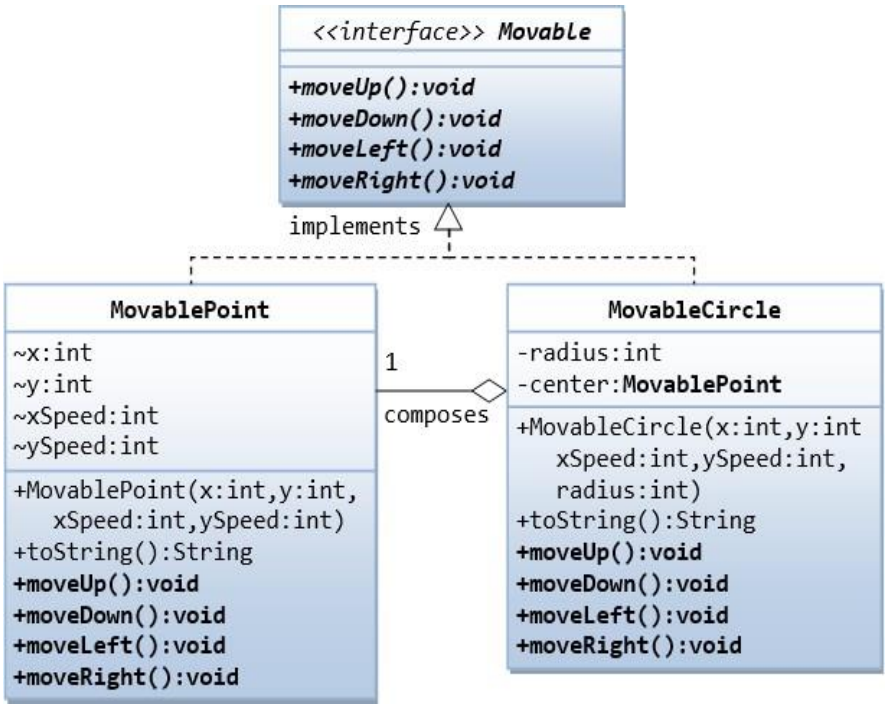
OUTPUT:

```
C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>javac q1.java

C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>java q1
Enter side-1 measurement:2.4
Enter side-2 measurement:1.3
Enter side-3 measurement:4.5

Part:
Side-1: 2.4
Side-2: 1.3
Side-3: 4.5
Inner Volume: 14.6952
Outer Volume: 58.7808
```

Gokul Jayan
20BCE1249

Question 2

Suppose that we have a set of objects with some common behaviors: they could move up, down, left or right. The exact behaviors (such as how to move and how far to move) depend on the objects themselves. Model these common behaviors as an interface called Movable, with abstract methods moveUp(), moveDown(), moveLeft() and moveRight(). The classes that implement the Movable interface will provide actual implementation to these abstract methods.



Write a complete Java program to define the interface, two client classes and a driver program to demonstrate the above classes and methods.

CODE:

```
import java.util.Scanner;

interface Movable
{
   void moveUp();
   void moveDown();
   void moveLeft();
   void moveRight();
}

class MovablePoint implements Movable
{
   public int x, y, xSpeed, ySpeed;
   Scanner in = new Scanner(System.in);

   MovablePoint(int x, int y, int xSpeed, int ySpeed)
   {
      this.x=x;
```

```
        this.y=y;
        this.xSpeed=xSpeed;
        this.ySpeed=ySpeed;
    }

    public String toString()
    {
        return "("+x+", "+y+")";
    }

    public void moveUp()
    {
        System.out.println("Before Movement: "+toString());
        y+=ySpeed;
        System.out.println("After Movement : "+toString());
    }

    public void moveDown()
    {
        System.out.println("Before Movement: "+toString());
        y-=ySpeed;
        System.out.println("After Movement : "+toString());
    }

    public void moveLeft()
    {
        System.out.println("Before Movement: "+toString());
        x-=xSpeed;
        System.out.println("After Movement : "+toString());
    }
    public void moveRight()
    {
        System.out.println("Before Movement: "+toString());
        x+=xSpeed;
        System.out.println("After Movement : "+toString());
    }

}

class MovableCircle implements Movable
{
    public int x, y, xSpeed, ySpeed, radius;
    Scanner in = new Scanner(System.in);

    MovableCircle()
    {
        x=0;
        y=0;
        xSpeed=0;
        ySpeed=0;
        radius=0;
    }

    Movable center =new MovablePoint(x,y,xSpeed,ySpeed);
```

```
    MovableCircle(int x, int y, int xSpeed, int ySpeed, int radius)
    {
        this.x=x;
        this.y=y;
        this.xSpeed=xSpeed;
        this.ySpeed=ySpeed;
        this.radius=radius;
    }

    public String toString()
    {
        return "("+x+", "+y+")";
    }

    public void moveUp()
    {
        System.out.println("Before Movement: "+toString());
        y+=ySpeed;
        System.out.println("After Movement : "+toString());
    }

    public void moveDown()
    {
        System.out.println("Before Movement: "+toString());
        y-=ySpeed;
        System.out.println("After Movement : "+toString());
    }

    public void moveLeft()
    {
        System.out.println("Before Movement: "+toString());
        x-=xSpeed;
        System.out.println("After Movement : "+toString());
    }
    public void moveRight()
    {
        System.out.println("Before Movement: "+toString());
        x+=xSpeed;
        System.out.println("After Movement : "+toString());
    }
}

public class q2
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        int ch;

        System.out.println("\nPoint\n");

        System.out.print("Enter x-coordinate:");
        int x=in.nextInt();
        System.out.print("Enter y-coordinate:");
        int y=in.nextInt();
```

```
        System.out.print("Enter x-Speed: ");
        int xSpeed=in.nextInt();
        System.out.print("Enter y-Speed: ");
        int ySpeed=in.nextInt();

        Movable m= new MovablePoint( x, y, xSpeed, ySpeed);
        while(true)
        {
           System.out.println("\n1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit");
           System.out.print("Enter your choice: ");
           ch=in.nextInt();
           if(ch==1)
           m.moveUp();
           else if(ch==2)
           m.moveDown();
           else if(ch==3)
           m.moveLeft();
           else if(ch==4)
           m.moveRight();
           else if(ch==5)
           break;
        }

        System.out.println("\n\nCircle Center\n");

        System.out.print("Enter x-coordinate:");
        x=in.nextInt();
        System.out.print("Enter y-coordinate:");
        y=in.nextInt();
        System.out.print("Enter x-Speed: ");
        xSpeed=in.nextInt();
        System.out.print("Enter y-Speed: ");
        ySpeed=in.nextInt();
        System.out.print("Enter radius: ");
        int radius=in.nextInt();
        Movable c= new MovableCircle(x, y, xSpeed, ySpeed, radius);
        while(true)
        {
           System.out.println("\n1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit");
           System.out.print("Enter your choice: ");
           ch=in.nextInt();
           if(ch==1)
           c.moveUp();
           else if(ch==2)
           c.moveDown();
           else if(ch==3)
           c.moveLeft();
           else if(ch==4)
           c.moveRight();
           else if(ch==5)
           break;
        }
     }
}
```

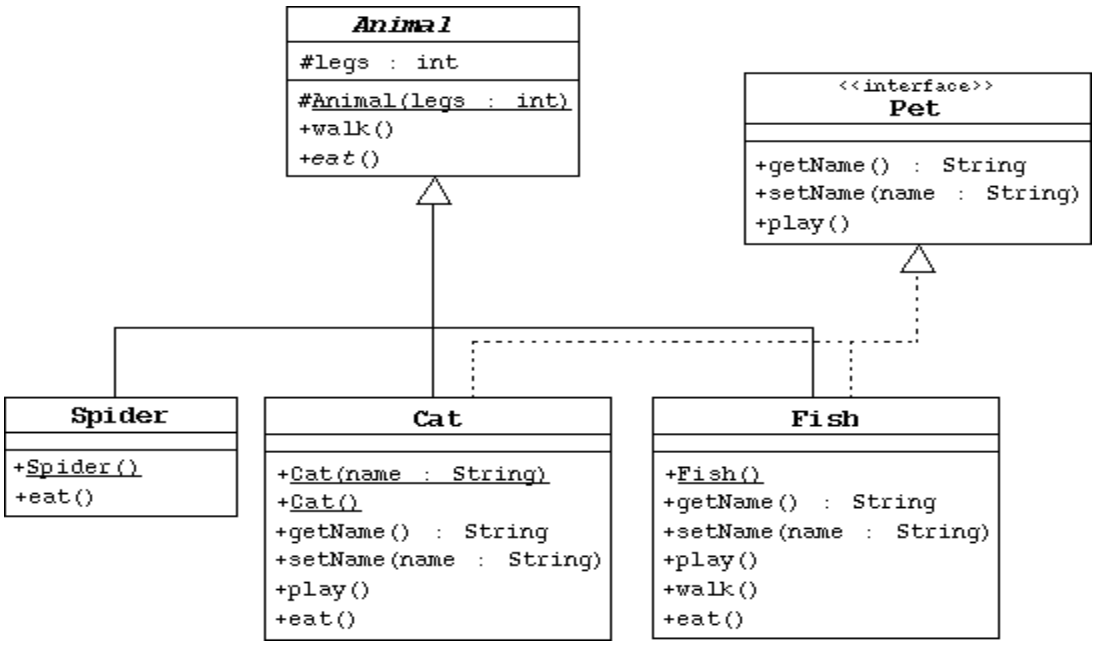
Gokul Jayan
20BCE1249

OUTPUT:

```
C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>javac q2.java

C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>java q2

Point

Enter x-coordinate:2
Enter y-coordinate:3
Enter x-Speed: 1
Enter y-Speed: 1

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 1
Before Movement: (2, 3)
After Movement : (2, 4)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 3
Before Movement: (2, 4)
After Movement : (1, 4)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 2
Before Movement: (1, 4)
After Movement : (1, 3)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 4
Before Movement: (1, 3)
After Movement : (2, 3)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 5

Circle Center

Enter x-coordinate:5
Enter y-coordinate:4
Enter x-Speed: 1
Enter y-Speed: 1
Enter radius: 3

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 1
Before Movement: (5, 4)
After Movement : (5, 5)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 3
Before Movement: (5, 5)
After Movement : (4, 5)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 2
Before Movement: (4, 5)
After Movement : (4, 4)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 4
Before Movement: (4, 4)
After Movement : (5, 4)

1. Move Up  2.Move Down  3.Move Left  4.Move Right  5.Exit
Enter your choice: 5
```

Gokul Jayan
20BCE1249

Question 3



1. Create the Animal class, which is the abstract superclass of all animals.

   - Declare a protected integer attribute called legs, which records the number of legs for this animal.
   - Define a protected constructor that initializes the legs attribute.
   - Declare an abstract method eat.
   - Declare a concrete method walk that prints out something about how the animals walks (include the number of legs).

2. Create the Spider class.

   - The Spider class extends the Animal class.
   - Define a default constructor that calls the superclass constructor to specify that all spiders have eight legs.
   - Implement the eat method.

3. Create the Pet interface specified by the UML diagram.

4. Create the Cat class that extends Animal and implements Pet.

   - This class must include a String attribute to store the name of the pet.
   - Define a constructor that takes one String parameter that specifies the cat's name. This constructor must also call the superclass constructor to specify that all cats have four legs.
   - Define another constructor that takes no parameters. Have this constructor call the previous constructor (using the this keyword) and pass an empty string as the argument.
   - Implement the Pet interface methods.
   - Implement the eat method.

5. Create the Fish class. Override the Animal methods to specify that fish can't walk and don't have legs.

6. Create an TestAnimals program. Have the main method create and manipulate instances of the classes you created above. Start with:

   ```
   Fish d = new Fish();
   Cat c = new Cat("Fluffy");
   Animal a = new Fish();
   Animal e = new Spider();
   Pet p = new Cat();
   ```

7. Experiment by: a) calling the methods in each object, b) casting objects, c) using polymorphism, and d) using super to call super class methods.

CODE:

```
import java.util.Scanner;

abstract class Animal
{
   protected int legs;

   protected Animal(int legs)
   {
      this.legs=legs;
   }

   void walk()
   {
      if(legs>0)
      System.out.println("Animal with "+legs+" legs is walking!");
      else
      System.out.println("Animal with "+legs+" legs can't walk!");
   }

   abstract void eat();
}

class Spider extends Animal
{
   Spider(int legs)
   {
      super(8);
   }

   public void eat()
   {
      System.out.println("Spider is eating insects!");
   }
}

interface Pet
{
```

```
    String getName();
    void setName();
    void play();
}

class Cat extends Animal implements Pet
{
    String name;

    Cat(String name)
    {
        super(4);
        this.name=name;
    }

    Cat()
    {
        this(" ");
    }

    public String getName()
    {
        return name;
    }

    public void setName()
    {
        Scanner in = new Scanner(System.in);
        System.out.print("\nEnter name of cat: ");
        name= in.next();
    }

    public void play()
    {
        System.out.println("Cat is playing with ball!");
    }

    public void eat()
    {
        System.out.println("Cat is eating fish!");
    }
}

class Fish extends Animal
{
    String name;

    Fish()
    {
        super(0);
    }

    public String getName()
    {
        return name;
```

```
    }

    public void setName()
    {
        Scanner in = new Scanner(System.in);
        System.out.print("\nEnter name of fish: ");
        name= in.next();
    }

    public void eat()
    {
        System.out.println("Fish is eating something!");
    }
}

public class TestAnimals
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        Fish d = new Fish();
        Cat c = new Cat();
        Animal a = new Fish();
        Animal e = new Spider(8);
        Pet p = new Cat();

        //Fish
        d.setName();
        System.out.println("\nFish");
        System.out.println("Cat Name: "+d.getName());
        d.walk();
        d.eat();

        //Cat
        c.setName();
        System.out.println("\nCat:");
        System.out.println("No: of legs = "+c.legs);
        System.out.println("Cat Name: "+c.getName());
        c.walk();
        c.eat();
        c.play();

        //Animal Fish
        System.out.println("\nFish:");
        System.out.println("No: of legs = "+a.legs);
        a.walk();
        a.eat();

        //Animal Spider
        System.out.println("\nSpider:");
        System.out.println("No: of legs = "+e.legs);
        e.walk();
        e.eat();

        //Pet Cat
```

```
        p.setName();
        System.out.println("\nCat:");
        System.out.println("Cat Name: "+c.getName());
        p.play();
    }
}
```

OUTPUT:

```
C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>javac TestAnimals.java

C:\Gokul\VIT\SEM-4\CSE1007 - Java\Lab\Lab7>java TestAnimals

Enter name of fish: Limo

Fish
Cat Name: Limo
Animal with 0 legs can't walk!
Fish is eating something!

Enter name of cat: Ollie

Cat:
No: of legs = 4
Cat Name: Ollie
Animal with 4 legs is walking!
Cat is eating fish!
Cat is playing with ball!

Fish:
No: of legs = 0
Animal with 0 legs can't walk!
Fish is eating something!

Spider:
No: of legs = 8
Animal with 8 legs is walking!
Spider is eating insects!

Enter name of cat: Ruby

Cat:
Cat Name: Ollie
Cat is playing with ball!
```