Introduction to Java: Lab Assignment 5

Student number: 121376141

Student name: Peace Samuel

Program1:

A polyline is a line with segments formed by points. Let's use the ArrayList (dynamically allocated array) to keep the points, but upcast to List in the instance variable. (Take note that array is of fixed-length, and you need to set the initial length).

```
//student number: 121376141
//student name: Peace Samuel

import java.util.*;
//Program 1: Polyline
public class Polyline {
    ArrayList<Point> points; //list of points

    static class Point {
    //point class, initializes point instances
    //points have an x and y value (instance variables)
    int x;
    int y;
    //constructor to create a new point
    Point(int x, int y){
        this.x = x;
        this.y = y;
    }
}
//constructor to create a new polyline
//one created with no input and the other created with an array list
    public Polyline(){
        points = new ArrayList<Point>();
    }
    public Polyline(ArrayList<Point> points){
        this.points = points;
    }

    public void appendPoint(int x, int y){
        //append or add point to the list using (x,y) values
        Point point = new Point(x, y);
        points.add(point);
    }
    public void appendPoint(Point point){
        //append or add point using point instance
        points.add(point);
    }
    public String toString(){
        //print string representation of the points in the polyline
```

```
        System.out.print("Points in polyline: ");
        String string = "";
        for(Point point : points){
            string = "(" + point.x +","+ point.y + ")";
            System.out.print(string);
        }
        System.out.println();
        return string;
    }
    public void getLength(){
        //find and return as a double the length of the polyline
        double length = 0;
        for (int i =0; i< points.size()-1; i++){
            Point p = points.get(i);
            Point p1 = points.get(i+1);
            //use distance formula for finding the distance of a line, using x
and y values
            double dx = p.x - p1.x;
            double dy = p1.y - p1.y;
            double distance = Math.sqrt(dx*dx + dy*dy);
            //add together all the distances to find the total length
            length += distance;
        }
        System.out.print("Length of the polyline: ");
        System.out.println(length);
    }

    public static void main(String[] args){
        ArrayList<Point> points = new ArrayList<Point>();
        points.add(new Point(1,2));
        points.add(new Point (3,4));
        Polyline line1 = new Polyline(points);
        line1.appendPoint(new Point(5,6));
        line1.appendPoint(8,3);
        line1.toString();
        line1.getLength();
        Polyline line2 = new Polyline();
        line2.appendPoint(8,9);
        line2.appendPoint(new Point(2,3));
        line2.toString();
        line2.getLength();
    }
}
```

OUTPUT 1:

```
PS C:\Users\peace\OneDrive\Desktop\Introduction to Java\assignment5> j
Points in polyline: (1,2)(3,4)(5,6)(8,3)
Length of the polyline: 7.0
Points in polyline: (8,9)(2,3)
Length of the polyline: 6.0
```

Program2:

You are asked to write a discount system for a beauty salon, which provides services and sells beauty products. It offers 3 types of memberships: Premium, Gold and Silver. Premium, gold and silver members receive a discount of 20%, 15%, and 10%, respectively, for all services provided. Customers without a membership receive no discount. All members receives a flat 10% discount on products purchased (this might change in future). Your system shall consist of three classes: Customer, Discount and Visit, as shown in the class diagram. It shall compute the total bill if a customer purchases $x of products and $y of services, for a visit. Also write a test program to exercise all the classes.

```java
//student number: 121376141
//student name: Peace Samuel
//Program2: Beauty Salon

import java.util.*;

public class Salon{
    public static void main(String[] args){
        ///test the classes///
        Customer customer1 = new Customer("Peace");
        customer1.setMember(true);
        customer1.setMemberType("Premium");
        System.out.println(customer1.toString());
        Date date = new Date();
        Visit v1 = new Visit(customer1, date);
        v1.setProductExpense(500);
        v1.setServiceExpense(500);
        System.out.println("Service expense(with discount incl.):
"+v1.getServiceExpense());
        System.out.println("Product expense(with discount incl.):
"+v1.getProductExpense());
        System.out.println("Total expense(with discount incl.):
"+v1.getTotalExpense());
        System.out.println(v1.toString());
    }
}

public class Customer {
```

```
    //(instance variables)
    String name; //name of customer
    Boolean member; //member or not
    String memberType; //membership type
//constructor
    public Customer(String name){
        this.name = name;
        this.member = false;
        this.memberType = "";
    }
//getters and setters
    public String getName(){
        return name;
    }

    public boolean isMember(){
        return member;
    }

    public void setMember(boolean member){
        this.member = member;
    }

    public String getMemberType(){
        return memberType;
    }

    public void setMemberType(String type){
        this.memberType = type;
    }

    public String toString(){
        String string = "";
        string = "Name: "+name+", Member: "+member+", Membership: "+
memberType;
        return string;
    }
}

public class Visit{
    //(instance variables)
    Customer customer; //name,mem,mem_type
    Date date;
    double serviceExpense;
    double productExpense;

//constructor
    public Visit(Customer customer, Date date){
```

```
        this.customer = customer;
        this.date = date;
        this.serviceExpense = 0;
        this.productExpense = 0;
        //service expense and product expense set later in methods
        //initialize them to zero
    }

    public String getName(){
        return customer.getName();
    }

    public double getServiceExpense(){
        //create DiscountRate instance to find service expense with discount
included
        DiscountRate discount = new DiscountRate();
        double rate =
discount.getServiceDiscountRate(customer.getMemberType());
        double discount2, total;
        discount2 = rate * serviceExpense;
        total = serviceExpense - discount2;
        return total;
    }

    public void setServiceExpense(double ex){
        this.serviceExpense = ex;
    }

    public double getProductExpense(){
        //create DiscountRate instance to find product expense with the
discount included
        DiscountRate discount = new DiscountRate();
        double rate =
discount.getProductDiscountRate(customer.getMemberType());
        double discount1, total;
        discount1 = rate * productExpense;
        total = productExpense - discount1;
        return total;
    }

    public void setProductExpense(double ex){
        this.productExpense = ex;
    }

    public double getTotalExpense(){
        //use previous methods to find total expense
        double total_expense;
        total_expense = this.getProductExpense() + this.getServiceExpense();
```

```
        return total_expense;
    }

    public String toString(){
        String string = "";
        string = "Customer:" + customer.getName() + ", Total expense:"+
this.getTotalExpense()+ ", Date: "+date;
        return string;
    }
}

public class DiscountRate{
    //(instance variables)
    double serviceDiscountPremium;
    double serviceDiscountGold;
    double serviceDiscountSilver;
    double productDiscountPremium;
    double productDiscountGold;
    double productDiscountSilver;
//constructor
    public DiscountRate(){
        this.serviceDiscountPremium = 0.2;
        this.serviceDiscountGold = 0.15;
        this.serviceDiscountSilver = 0.1;
        this.productDiscountPremium = 0.1;
        this.productDiscountGold = 0.1;
        this.productDiscountSilver = 0.1;
    }

    public double getServiceDiscountRate(String type){
        double s_discount = 0;
        if (type == "Premium"){
            s_discount = serviceDiscountPremium;
        }
        if (type == "Gold"){
            s_discount = serviceDiscountGold;
        }
        if (type == "Silver"){
            s_discount = serviceDiscountSilver;
        }
        return s_discount;
    }

    public double getProductDiscountRate(String type){
        double p_discount= 0;
        if (type == "Premium"){
            p_discount = productDiscountPremium;
        }
```

```
        if (type == "Gold"){
            p_discount = productDiscountGold;
        }
        if (type == "Silver"){
            p_discount = productDiscountSilver;
        }
        return p_discount;
    }

}
```

OUTPUT 2:

```
PS C:\Users\peace\OneDrive\Desktop\Introduction to Java\assignment5> java
Name: Peace, Member: true, Membership: Premium
Service expense(with discount incl.): 400.0
Product expense(with discount incl.): 450.0
Total expense(with discount incl.): 850.0
Customer:Peace, Total expense:850.0, Date: Fri Mar 17 19:17:12 GMT 2023
PS C:\Users\peace\OneDrive\Desktop\Introduction to Java\assignment5>
```

Program3:

Create a subclass called Cylinder, which is derived from the superclass Circle as shown in the class diagram (where an an arrow pointing up from the subclass to its superclass). Study how the subclass Cylinder invokes the superclass' constructors (via super() and super(radius)) and inherits the variables and methods from the superclass Circle.
The subclass Cylinder inherits getArea() method from its superclass Circle.
Try overriding the getArea() method in the subclass Cylinder to compute the surface area (=2π×radius×height + 2×base-area) of the cylinder instead of base area. That is, if getArea() is called by a Circle instance, it returns the area. If getArea() is called by a Cylinder instance, it returns the surface area of the cylinder.

```
//student number: 121376141
//student name: Peace Samuel
//Program 3: Circle & Cylinder

import java.util.*;
public class MainCircle{
    public static void main(String args[]){
        Circle circle1 = new Circle();
        System.out.println(circle1.toString());
        System.out.println("Area of the circle: "+ circle1.getArea());
        circle1.setColor("blue");
        circle1.toString();
        Cylinder c1 = new Cylinder(0.1,1.0,"blue");
        System.out.println(c1.toString());
        System.out.println("Area of the cylinder: "+ c1.getArea());
    }
}
```

```java
public class Circle {
    //SUPERCLASS//
    //(instance variables)
    double radius; //radius of circle
    String color; //color of circle

//constructor to create a new circle
    public Circle(){
        //set default construct values for radius and color
        this.color = "red";
        this.radius = 0.1;
    }

    public Circle(double radius){
        this.color = "red";
        this.radius = radius;
    }

    public Circle(double radius, String color){
        this.radius = radius;
        this.color = color;
    }
//getters and setters
    public double getRadius(){
        return radius;
    }

    public void setRadius(double radius){
        this.radius = radius;
    }

    public String getColor(){
        return color;
    }

    public void setColor(String color){
        this.color = color;
    }
//other methods
    public double getArea(){
        //use area of a circle formula, pi(r-squared)
        double area = Math.PI *radius * radius;
        return area;
    }
    public String toString(){
        //print string representation of the circle
        String string ="";
```

```
        string = "Circle"+ "[radius:" + radius +", color:" + color +  "]";
        return string;
    }
}


public class Cylinder extends Circle{
    //SUBCLASS//
    //(instance variables)
    double height; //height of the cylinder
//constructor
    public Cylinder(double radius, double height, String color){
        super(radius, color);
        this.height = height;
    }

    public Cylinder(double radius){
        super(radius);
        this.height = 1.0;
    }

    public Cylinder(double radius, double height){
        super(radius);
        this.height = height;
    }

    public Cylinder(){
        super(); //uses radius and color default value from  circle superclass
        this.height = 1.0;
    }
//getters and setters
    public double getHeight(){
        return height;
    }

    public void setHeight(double height){
        this.height = height;
    }

    public double getVolume(){
        //volume formula =base_area * height
        double base_area = Math.PI *radius * radius;
        double volume = base_area*height;
        System.out.println("the volume of this cylinder: "+ volume);
        return volume;
    }

    @Override
    public double getArea(){
```

```
        //override Circle area method//
        //use area formula: 2pi *radius *height+ (2base-area)
        //base area: pi(radius-squared)
        double base_area = Math.PI *radius * radius;
        double area = (2*(Math.PI)*radius*height) + base_area;
        return area;
    }
//other methods
    @Override
    public String toString() {
        return "Cylinder[radius: " + getRadius() + ", height: " + height + ",
color: " + color+ "]";
    }
}
```

OUTPUT 3:

```
Circle[radius:0.1, color:red]
Area of the circle: 0.031415926535897934
Cylinder[radius: 0.1, height: 1.0, color: blue]
Area of the cylinder: 0.6597344572538566
PS C:\Users\peace\OneDrive\Desktop\Introduction to Java\assignment
```

Program4:

Write a superclass called Shape (as shown in the class diagram), which contains:
• Two instance variables color (String) and filled (boolean).
• Two constructors: a no-arg (no-argument) constructor that initializes the color to "green" and filled to true, and a constructor that initializes the color and filled to the given values.
• Getter and setter for all the instance variables. By convention, the getter for a boolean variable xxx is called isXXX() (instead of getXxx() for all the other types).
• A toString() method that returns "A Shape with color of xxx and filled/Not filled".
Write a test program to test all the methods defined in Shape.
Write two subclasses of Shape called Circle and Rectangle, as shown in the class diagram.
The Circle class contains:
• An instance variable radius (double).
• Three constructors as shown. The no-arg constructor initializes the radius to 1.0.
• Getter and setter for the instance variable radius.
• Methods getArea() and getPerimeter().

Override the toString() method inherited, to return "A Circle with radius=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.
The Rectangle class contains:
• Two instance variables width (double) and length (double).
• Three constructors as shown. The no-arg constructor initializes the width and length to 1.0.
• Getter and setter for all the instance variables.
• Methods getArea() and getPerimeter().
• Override the toString() method inherited, to return "A Rectangle with width=xxx and

length=zzz, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

Write a class called Square, as a subclass of Rectangle. Convince yourself that Square can be modeled as a subclass of Rectangle. Square has no instance variable, but inherits the instance variables width and length from its superclass Rectangle.

• Provide the appropriate constructors (as shown in the class diagram).

• Override the toString() method to return "A Square with side=xxx, which is a subclass of yyy", where yyy is the output of the toString() method from the superclass.

• Do you need to override the getArea() and getPerimeter()? Try them out.

• Override the setLength() and setWidth() to change both the width and length, so as to maintain the square geometry.

```
//student number: 121376141
//student name: Peace Samuel
//Program 4: Shapes

import java.util.*;
public class MainShape{
    //test out the classes
    public static void main(String[] args){
        Shape s1 = new Shape();
        System.out.println(s1.toString());
        s1.setColor("blue");
        System.out.println(s1.toString());
        Circle c1 = new Circle();
        System.out.println(c1.toString());
        Rectangle r1 = new Rectangle();
        r1.setLength(2.0);
        System.out.println(r1.toString());
        System.out.println("Area of rectangle:"+r1.getArea());
        Square s = new Square();
        System.out.println(s.toString());
        System.out.println(s.isFilled());
        System.out.println("Area of square:"+s.getArea());
    }
}

public class Shape{
    //SUPERCLASS//
    //(instance variables)
    String color; //color of shape
    boolean filled; //states whether shape is filled or not

//constructor
    public Shape(){
        //default values for the shape
        this.color = "green";
        this.filled = true;
    }
```

```
    public Shape(String color, boolean filled){
        this.color = color;
        this.filled = filled;
    }
//getters and setters
    public String getColor(){
        return color;
    }

    public void setColor(String color){
        this.color = color;
    }

    public boolean isFilled(){
        return filled;
    }

    public void setFilled(boolean filled){
        this.filled = filled;
    }
//other methods
    public String toString(){
        return "Shape[color:"+color+", filled:"+filled+"]";
    }
}

public class Circle extends Shape{
    //SUBCLASS//
    //(instance variables)
    double radius; //radius of circle
//constructor
    public Circle(){
        super(); //uses constructor from superclass
        this.radius = 1.0;
    }

    public Circle(double radius){
        super();
        this.radius = radius;
    }

    public Circle(double radius, String color, boolean filled){
        super(color, filled);
        this.radius = radius;
    }

    public double getRadius(){
```

```
        return radius;
    }

    public void setRadius(double radius){
        this.radius = radius;
    }

    public double getArea(){
        //area of circle formula: pi(r-squared)
        double area = Math.PI * radius* radius;
        return area;
    }

    public double getPerimeter(){
        //perimeter formula: 2pi(radius)
        double perimeter =2*Math.PI*radius;
        return perimeter;
    }

    public String toString(){
        String string = "";
        string = "Circle[radius:" +radius+"], "+ "Subclass of: "+
super.toString();
        return string;
    }
}

public class Rectangle extends Shape{
    //SUBCLASS// +//SUPERCLASS//[SQUARE]
    //(instance variables)
    double width; //width of rectangle
    double length; //length of rectangle
//constructor
    public Rectangle(){
        super();
        this.width = 1.0;
        this.length = 1.0;
    }

    public Rectangle(double width, double length){
        super();
        this.width = width;
        this.length = length;
    }

    public Rectangle(double width, double length, String color, boolean
filled){
        super(color, filled);
```

```
        this.width = width;
        this.length = length;
    }
//getter and setters
    public double getWidth(){
        return width;
    }

    public void setWidth(double width){
        this.width = width;
    }

    public double getLength(){
        return length;
    }

    public void setLength(double length){
        this.length = length;
    }

    public double getArea(){
        //area of rectangle formula: l*w
        double area = length*width;
        return area;
    }

    public double getPerimeter(){
        //perimeter of rectangle: 2*l+2*w
        double perimeter = (2*length) + (2*width);
        return perimeter;
    }

    public String toString(){
        String string ="";
        string = "Rectangle[width:"+width+", length:"+length+"], Subclass
of:"+ super.toString();
        return string;
    }
}

public class Square extends Rectangle{
    //SUBCLASS//
    //(no instance variables, inherits from superclass)
//constructor
    public Square(){
        super();
    }
```

```java
    public Square(double side){
       super(side, side);
    }

    public Square(double side, String color, boolean filled){
        super(side, side, color, filled);
    }

    public double getSide(){
        return length;
    }

    public void setSide(double side){
        this.length = side;
        this.width = side;
    }

    public void setWidth(double side){
        this.width = side;
        this.length = side;
    }

    public void setLength(double side){
        this.length = side;
        this.width = side;
    }

    public String toString(){
        String string = "";
        string = "Square["+ super.toString()+"]";
        return string;
    }
    //Don't need to override getArea() and getPerimeter()//

}
```

OUTPUT 4:

```
Shape[color:green, filled:true]
Shape[color:blue, filled:true]
Circle[radius:1.0], Subclass of: Shape[color:green, filled:true]
Rectangle[width:1.0, length:2.0], Subclass of:Shape[color:green, filled:true]
Area of rectangle:2.0
Square[Rectangle[width:1.0, length:1.0], Subclass of:Shape[color:green, filled:true]]
true
Area of square:1.0
```