Program 1;

This program is used to implement a polyline, and getting a polyline's length, with different points as the property.

Here is some pseudocode that will help explain the programs code:

1. Start
2. Import java's array list and list classes
3. Define a class for point, with properties x (int) and y (int)
4. Define methods for the point class:
    a. Constructor to set x and y
    b. A toString method that will return a string representation of the point
5. Define a class for polyline, with a list property, to hold points
6. Define methods for the polyline class:
    a. Constructor to initialise lists
    b. Methods to append new and existing points onto the polyline
    c. Getter for length (Double) (will calculate and return length of polyline)
    d. A toString that will return the string representation of all the points on the polyline
7. Define a class for the main, which will test the classes:
    a. Create instances of point and polyline
    b. Add points onto the polyline
    c. Print out the polyline and its length
8. End

Code:

```
import java.util.ArrayList; // import the array list and the list
import java.util.List;

class Point { // define a point class, with x and y properties
    int x;
    int y;

    public Point(int x, int y) { // constructor that initialises specific x and y coordinates
        this.x = x;
        this.y = y;
    }

    public String toString() { // method that will get the string representation of a strings coordinates
        return "(" + x + "," + y + ")";
    }
}
```

```
class Polyline { // define a polyline class with list as a property
    private List<Point> points;

    public Polyline() { // consctructor that initialises the polyline as an empty list of points
        this.points = new ArrayList<>();
    }

    public void appendPoint(int x, int y) { // method to add a point into the polyline
        Point newPoint = new Point(x, y);
        points.add(newPoint);
    }

    public void appendPoint(Point point) { // ,method that add an existing point into the polyline
        points.add(point);
    }

    public double getLength() { // method that will get (calculate and return) length
        double length = 0;
        for (int i = 0; i < points.size() - 1; i++) {
            Point start = points.get(i);
            Point end = points.get(i + 1);
            length += Math.hypot(end.x - start.x, end.y - start.y);
        }
        return length;
    }

    public String toString() { // method to get the string representation of the points on a polyline
        StringBuilder sb = new StringBuilder("{");
        for (Point point : points) {
            sb.append(point);
        }
        sb.append("}");
        return sb.toString();
    }
}
```

```
public class Main { // main method that contains the method to test the point and polyline classes
    public static void main(String[] args) {
        Polyline polyline = new Polyline();
        polyline.appendPoint(1, 1); // appends points onto the polyline
        polyline.appendPoint(2, 3);
        polyline.appendPoint(3, 0);

        System.out.println("Polyline: " + polyline);
        System.out.println("Length of Polyline: " + polyline.getLength());
    }
}
```

Answer:

```
Polyline: {(1,1)(2,3)(3,0)}
Length of Polyline: 5.398345637668l7
```

Program 2:

This program is used to implement a discount system for a beauty salon, where discounts depend on membership levels, including getting total expense, discount, and price after discount. You also have the ability to observe (set) and change (get) different properties including membership types, and service/product expenses.

Here is some pseudocode that will help explain the programs code:

1. Start
2. Define a class for customer, with properties name (String), member (Boolean) and memberType (String) – defaults for member and memberType are included
3. Define methods for the customer class:
    c. Constructor for name
    d. Getter for name, and Getters and Setters for member and memberType
    e. A toString method that will return a string describing the customer's name and membership status.
4. Define a class for visit, with properties customer (String), date (Date), serviceExpense (double) and productExpense (double)
5. Define methods for the visit class:
    a. Constructor for a customer's visit
    b. Getter for name, and Getters and Setters for serviceExpense and productExpense
    c. Getter for totalExpense (Double) (will calculate and return the total expense of the customer's visit)
    d. A toString method that will return a string that describes the visit's details
6. Define a discount rate class with static properties for product/service discounts for each membership type – serviceDiscountPremium (double), serviceDiscountGold (double), serviceDiscountSilver (double), productDiscountPremium (double),productDiscountGold (double), productDiscountSilver (double)
7. Define method for discount class:
    a. Getters for product and service discount rates
8. Define a class for the main, which will test the classes
    a. Creates instances of customer and visit
    b. Print out these details, alongside product/service expense before/after discount, alongside total expense after discount
9. End

Code:

```
class Customer { // define a define a customer, with name, member, and member-type properties
    private String name; // private class members used to store name, member, and member-type (property values are the defaults)
    private boolean member = false;
    private String memberType;

    public Customer(String name) { // constuctor to create a customer with a name
        this.name = name;
    }

    public String getName() { // getter for name
        return name;
    }

    public boolean isMember() { // getter to see if ctomer is a member
        return member;
    }

    public void setMember(boolean member) { // setter for member status
        this.member = member;
    }

    public String getMemberType() { // getter for member-type
        return memberType;
    }

    public void setMemberType(String type) { // setter for member-type
        this.memberType = type;
    }

    @Override
    public String toString() { // method that will get string representation of a customer status
        String memberStr = isMember() ? ("Member Type: " + getMemberType()) : "Not a Member";
        return "Customer Name: " + getName() + ", " + memberStr;
    }
}
```

```
class Visit { // define a visit class, with customer, date and service/product expense as properties
    private Customer customer;  // private class members used to store customer, date, and service/expense
    private java.util.Date date;
    private double serviceExpense;
    private double productExpense;

    public Visit(String name, java.util.Date date) { // constrcutor to create a visit for a customer on a particular date
        this.customer = new Customer(name);
        this.date = date;
    }

    public String getName() { // getter for name
        return customer.getName();
    }

    public double getServiceExpense() { // getter for service expense
        return serviceExpense;
    }

    public void setServiceExpense(double ex) { // setter for service expense
        serviceExpense = ex;
    }

    public double getProductExpense() { // getter for product expense
        return productExpense;
    }

    public void setProductExpense(double ex) { // setter for product expense
        productExpense = ex;
    }

    public double getTotalExpense() { // getter that will get (caculate and return) toal expense
        return serviceExpense + productExpense;
    }

    @Override
    public String toString() { // this will get the string representation of the customer's visit
        return "Visit for " + customer.getName() +
               "\nDate of Service: " + date +
               "\nService Expense Before Discount: " + serviceExpense +
               "\nProduct Expense Before Discount: " + productExpense;
    }
}
```

```
class DiscountRate { // class to the hold the different discount rates for the different types of member
    static double serviceDiscountPremium = 0.2; // static variables to hold discount for different member types
    static double serviceDiscountGold = 0.15;
    static double serviceDiscountSilver = 0.1;
    static double productDiscountPremium = 0.1;
    static double productDiscountGold = 0.1;
    static double productDiscountSilver = 0.1;

    public static double getServiceDiscountRate(String type) { // method to retrieve the service discount, based on the member
type
        switch (type) {
            case "Premium":
                return serviceDiscountPremium;
            case "Gold":
                return serviceDiscountGold;
            case "Silver":
                return serviceDiscountSilver;
            default:
                return 0;
        }
    }

    public static double getProductDiscountRate(String type) { // method to retrieve the product discount based on the member
type
        switch (type) {
            case "Premium":
                return productDiscountPremium;
            case "Gold":
                return productDiscountGold;
            case "Silver":
                return productDiscountSilver;
            default:
                return 0;
        }
```

```
public class Main { // main class that contains the method to test out the customer, visit, and discount classes
    public static void main(String[] args) {
        Customer customer = new Customer("John Doe"); // create a customer to use as a test
        customer.setMember(true);
        customer.setMemberType("Gold");

        Visit visit = new Visit(customer.getName(), new java.util.Date()); // create a visit for the customer to use a test
        visit.setProductExpense(200.0);
        visit.setServiceExpense(50.0);

        double totalServiceExpense = visit.getServiceExpense() - (visit.getServiceExpense() *
DiscountRate.getServiceDiscountRate(customer.getMemberType())); // apply discounts to the visit
        double totalProductExpense = visit.getProductExpense() - (visit.getProductExpense() *
DiscountRate.getProductDiscountRate(customer.getMemberType()));

        System.out.println("Invoice for the Visit:"); // print before/after discount expenses
        System.out.println(visit);
        System.out.println("Service Expense After Discount: " + String.format("%.2f", totalServiceExpense));
        System.out.println("Product Expense After Discount: " + String.format("%.2f", totalProductExpense));
        System.out.println("Total Expense After Discount: " + String.format("%.2f", (totalServiceExpense +
totalProductExpense)));
    }
}
```

Ln 63, Col 31 | 5,429 characters | 60% | Windows (CRLF) | UTF-8

Answer:

```
Invoice for the Visit:
Visit for John Doe
Date of Service: Sun Mar 10 17:58:55 GMT 2024
Service Expense Before Discount: 50.0
Product Expense Before Discount: 200.0
Service Expense After Discount: 42.50
Product Expense After Discount: 180.00
Total Expense After Discount: 222.50
```

Program 3:

This program is used to perform certain operations on different shapes, including getting a shapes area/volume, and observing (getting) and changing (setting) a shapes properties, including radius, length, colour etc. properties.

Here is some pseudocode that will help explain the programs code:

1. Start
2. Define a class for circle, with properties colour (String) and radius (Double) – defaults for these properties will be included
3. Define methods for the circle class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameters (it will set colour and radius)
    c. Getters and Setters for colour and radius
    d. Getter for area (Double) (will calculate and return the area of the circle)
    e. A toString method that will return a string describing the circle
4. Define a class for cylinder, that will extend (inherit) the previous circle class, but with the additional height property (Double) – defaults for height will be included
5. Define methods for the cylinder class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameters (one with inherited radius, one with that will set height and also inherits radius, and finally one that's sets radius, and inherits height and colour)
    c. Getter and Setter for hieght
    d. Getter for Volume (Double) (will calculate and return the volume of the cylinder)
    e. A toString that will return a string describing the cylinder.
6. Define a class for the main, which will test the classes
    a. Creates instances of circle, cylinder
    b. Print out these details, alongside area and volume calculations
7. End

Code:

```
class Circle { // 9define circle class, with radius and colour properties
    private double radius = 1.0; // private class members used to store colour and radius (property values are
the defaults)
    private String colour = "red";

    public Circle() { // a no-arguement constructor, where the properties of the class are set to the defaults

       }

    public Circle(double radius) { // constructor that sets radius and height
        this.radius = radius;
    }

    public Circle(double radius, String colour) { // constructor that sets radius and colour
        this.radius = radius;
        this.colour = colour;
    }

    public double getRadius() { // getter for radius
        return radius;
    }

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

    public String getColour() { // getter for colour
        return colour;
    }

    public void setColour(String colour) { // setter for colour
        this.colour = colour;
    }

    public double getArea() { // getter that will get (calculate and return) area of circle
        return Math.PI * radius * radius;
    }

    public String toString() { // method that will get string representation of a circle
        return "Circle[radius=" + radius + ", colour=" + colour + "]";
    }
}
```

```
class Cylinder extends Circle { // defines the class cylinder that extends (inherits off of) circle class
    private double height = 1.0; // private class members used to store height (property values are the defaults)


    public Cylinder() { // a no-arguement constructor, where the properties of the class are set to the defaults
    }

    public Cylinder(double radius) { // constructor that the radius (inherited from super)
        super(radius);
    }

    public Cylinder(double radius, double height) { // constructor that sets height and the radius (inherited
from super)
        super(radius);
        this.height = height;
    }

    public Cylinder(double radius, double height,  String colour) { // constructor that sets height, and both
colour and radius (both inherited from super class)
        super(radius, colour);
        this.height = height;
    }

    public double getHeight() { // getter for height
        return height;
    }

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

    @Override // this will override supers getArea() method
    public double getArea() {
        // Surface area of the cylinder (2πrh + 2πr^2)
        return (2 * Math.PI * getRadius() * height) + (2 * super.getArea());
    }

    public double getVolume() { // getter that will get(calculate and return) volume
        // Volume of the cylinder (πr^2h)
        return super.getArea() * height;
    }

    @Override // this will override supers string representation, and use cylinder's
    public String toString() {
        return "Cylinder[" + super.toString() + " and height=" + height;
    }
```

```
public class Main { // main class that will contain the method that will be used to test the different classes
    public static void main(String[] args) {
        Circle circle = new Circle(5, "blue");
        Cylinder cylinder = new Cylinder(5, 10, "blue");

        System.out.println(circle); // test circle class
        System.out.println("Area of the circle: " + circle.getArea());

        System.out.println(cylinder); // test cylinder class
        System.out.println("Surface area of the cylinder: " + cylinder.getArea());
        System.out.println("Volume of the cylinder: " + cylinder.getVolume());

        }
}
```

Answer:

```
Circle[radius=5.0, colour=blue]
Area of the circle: 78.53981633974483
Cylinder[Circle[radius=5.0, colour=blue] and height=10.0
Surface area of the cylinder: 471.23889803846896
Volume of the cylinder: 785.3981633974483
```

Program 4

This program is used to perform certain operations on different shapes, including getting a shapes area/perimeter, and observing (getting) and changing (setting) a shapes properties, including radius, length, colour etc. properties.

Here is some pseudocode that will help explain the programs code:

8. Start
9. Define a class for shape, with properties colour (a string colour) and filled (Boolean) – defaults for these properties will be included
10. Define methods for the shape class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameters (it will set colour and filled)
    c. Getters and Setters for colour and filled
    d. A toString method that will return a string describing the shape
11. Define a class for circle, that will extend (inherit) the previous shape class, but with the additional radius property (Double) – defaults for radius will be included
12. Define methods for the circle class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameters (one with set radius, and another set radius, and inherited parameters)
    c. Getter and Setter for radius
    d. Getter for area (Double) and parameter (Double) (will calculate and return the area and perimeter of the circle)
    e. A toString that will return a string describing the circle.
13. Define a class for rectangle, that will extend (inherit) the previous shape class, but with the additional length (Double) and width (Double) properties – defaults for length and width will be included
14. Define methods for the rectangle class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameters (one with set width and length, and another width and length, and inherited parameters)
    c. Getters and Setters for length and width
    d. Getter for area (Double) and parameter (Double) (will calculate and return the area and perimeter of the rectangle)
    e. A toString that will return a string describing the rectangle.
15. Define a class for square, that will extend (inherit) the previous rectangle class, but with the length and width being equal (known as sides (Double)) – defaults for side will be included
16. Define methods for the square class:
    a. Constructor without parameters (it will use the defaults)
    b. Constructor with parameter side and inherited parameters
    c. Getter and Setter for side (this will override length and width)
    d. A toString that will return a string describing the square.
17. Define a class for the main, which will test the classes
    a. Creates instances of shape, circle, square, rectangle
    b. Print out these details, alongside area and perimeter calculations

Code:

```
class Shape { //define shape class with colour and filled properties
    private String colour = "red"; //private class members used to store colour and filled status
    private boolean filled = true;

    public Shape() { //a no-arguement constructor, where the properties of the class are set to the defaults
    }

    public Shape(String colour, boolean filled) { //constructor that sets the colour and filled
        this.colour = colour;
        this.filled = filled;
    }

    public String getColour() { //getter for colour
        return colour;
    }

    public void setColour(String colour) { //setter for colour
        this.colour = colour;
    }

    public boolean isFilled() { //getter for filled
        return filled;
    }

    public void setFilled(boolean filled) { //setter for filled
        this.filled = filled;
    }

    public String toString() { //method that will get the string representation of the shape
        return "Shape[color=" + color + ",filled=" + filled + "]";
    }
}
```

```
class Circle extends Shape { //defines the class circle that extends (inherits of) shape class
    private double radius = 1.0; //private class member

    public Circle() { //a no-arguement constructor, where the properties of the class are set to the defaults
    }

    public Circle(double radius) { //constructor that sets radius
        this.radius = radius;
    }

    public Circle(double radius, String colour, boolean filled) { //constrcutor that sets radius, and the colour
and filled (both inherited from super class)
        super(colour, filled);
        this.radius = radius;
    }

    public double getRadius() { //getter for radius
        return radius;
    }

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

    public double getArea() { //getter that will get (calculate and return) area of circle
        return Math.PI * radius * radius;
    }

    public double getPerimeter() { //getter that will get (calculate and return) perimeter of circle
        return 2 * Math.PI * radius;
    }

    @Override //this will override and use string representation of the super class and use circle class's
instead
    public String toString() { //method that will get a string representation of the circle
        return "Circle[" + super.toString() + ",radius=" + radius + "]";
    }
}
```

```
class Rectangle extends Shape { //defines rectangle class, which extends (inherits) shape class
    private double width = 1.0; //private class members for width and length of rectangle
    private double length = 1.0;

    public Rectangle() { //a no-arguement constructor, where the properties of the class are set to the defaults
    }

    public Rectangle(double width, double length) { //constructor that will set length and width properties
        this.width = width;
        this.length = length;
    }

    public Rectangle(double width, double length, String colour, boolean filled) { //constructor that will set
length and width properties, alongside colour and filled properties, inherited from super class
        super(colour, filled);
        this.width = width;
        this.length = length;
    }

    public double getWidth() { //getter for width
        return width;
    }

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

    public double getLength() { //getter for length
        return length;
    }

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

    public double getArea() { //getter that will get (calculate and return) area
        return width * length;
    }

    public double getPerimeter() { //getter that will get(calculate and return) perimetre
        return 2 * (width + length);
    }

    @Override //will override string representation of super class, and will instead use rectangle class's
    public String toString() {
        return "Rectangle[" + super.toString() + ",width=" + width + ",length=" + length + "]";
    }
```

```
class Square extends Rectangle { //define the square class, that inherits from the rectangle class
    public Square() {
    }

    public Square(double side) { //a no-arguement constructor, where the properties of the class are set to the
defaults (in this case length=width=side)
        super(side, side);
    }

    public Square(double side, String colour, boolean filled) { //constructor that will set the side properties,
alongside the colour and filled properties that will be inherited from the super's super
        super(side, side, colour, filled);
    }

    public double getSide() { //getter for side
        return getWidth();
    }

    public void setSide(double side) { //setter for side
        super.setWidth(side);
        super.setLength(side);
    }

    @Override //this will override supers width setter
    public void setWidth(double width) { //setter for width
        setSide(width);
    }

    @Override //this will override supers length setter
    public void setLength(double length) { //setter for length
        setSide(length);
    }

    @Override //will override supers string representation and will use square's instead
    public String toString() {
        return "Square[" + super.toString().replace("Rectangle", "Shape") + "]";
    }
}
```

```
class Main { //main class that will contain the method that will be used to test the different classes
    public static void main(String[] args) {

        Shape shape = new Shape("purple", false); //test ths shape class
        System.out.println(shape);

        Circle circle = new Circle(5, "blue", false); //test the circle class
        System.out.println(circle);
        System.out.println("Area of circle: " + circle.getArea());
        System.out.println("Perimeter of circle: " + circle.getPerimeter());

        Rectangle rectangle = new Rectangle(4, 5, "red", true); //test the rectangle class
        System.out.println(rectangle);
        System.out.println("Area of rectangle: " + rectangle.getArea());
        System.out.println("Perimeter of rectangle: " + rectangle.getPerimeter());

        Square square = new Square(4, "yellow", true); //test the square class--
        System.out.println(square);
        System.out.println("Area of square: " + square.getArea());
        System.out.println("Perimeter of square: " + square.getPerimeter());
    }
}
```

Answer:

```
Shape[colour=purple,filled=false]
Circle[Shape[colour=blue,filled=false],radius=5.0]
Area of circle: 78.53981633974483
Perimeter of circle: 31.41592653589793
Rectangle[Shape[colour=red,filled=true],width=4.0,length=5.0]
Area of rectangle: 20.0
Perimeter of rectangle: 18.0
Square[Shape[Shape[colour=yellow,filled=true],width=4.0,length=4
.0]]
Area of square: 16.0
Perimeter of square: 16.0
```