

<b>Name</b>	Lekh Sanatan Nayak
<b>UID no.</b>	2023800068
<b>Experiment No.</b>	7

<b>AIM:</b>	Implement a Program to demonstrate abstraction using abstract classes
-------------	---

### Program 1

<b>PROBLEM STATEMENT :</b>	<p>Create a Shape class which is abstract with data color and abstract method area() and derive two classes Rectangle and Circle derived from shape, override area() method and complete the classes.</p> <p>Create a class called Diagram which contains an array of shape class. Each Diagram can be a composition of rectangle or circle shapes. Diagram class has method called totalArea() which computes the totalArea() of both shapes. Add setters, getters and constructors wherever required.</p> <p>Design a Tester class with main() to create different diagram objects and print the total areas and details of each diagram.</p>
<b>PROGRAM:</b>	<pre>import java.util.*;  // Class representing a store named Westside class Westside {     private int clothesAvailable; // Number of clothes available in the store     private int accessoriesAvailable; // Number of accessories available in the store     public static final int PRICE_ITEM = 5000; // Price of each item      // Constructor to initialize the availability of clothes and accessories     public Westside() {         this.clothesAvailable = 10;         this.accessoriesAvailable = 10;     }      // Method to purchase clothes     public boolean purchaseClothes() {         if (clothesAvailable &gt; 0) {             clothesAvailable--;         }     } }</pre>

```

        return true; // Clothes purchased successfully
    } else {
        return false; // Clothes out of stock
    }
}

// Method to purchase accessories
public boolean purchaseAccessories() {
    if (accessoriesAvailable > 0) {
        accessoriesAvailable--;
        return true; // Accessories purchased successfully
    } else {
        return false; // Accessories out of stock
    }
}

// Getter method to retrieve the number of clothes available
public int getClothesAvailable() {
    return clothesAvailable;
}

// Getter method to retrieve the number of accessories available
public int getAccessoriesAvailable() {
    return accessoriesAvailable;
}
}

// Abstract class representing a payment
abstract class Payment {
    protected double amount; // Total amount

    // Constructor to initialize the total amount
    public Payment() {
        amount = 25000; // Initial amount set to 25000
    }

    // Abstract method to provide payment details
    public abstract void paymentDetails(double price);
}

```

```

// Class representing cash payment
class CashPayment extends Payment {

    // Constructor to initialize cash payment
    public CashPayment() {
        super();
    }

    // Method to provide payment details for cash payment
    public void paymentDetails(double price) {
        amount -= price; // Deducting the price from the total amount
        System.out.println("Payment made in cash. Remaining balance is: " +
amount);
    }
}

// Class representing credit card payment
class CreditCardPayment extends Payment {
    private String cardNumber; // Credit card number
    private String cardExpirationDate; // Credit card expiration date
    private String cardName; // Name on the credit card

    // Constructor to initialize credit card payment details
    public CreditCardPayment(String cardName, String cardNumber, String
cardExpirationDate) {
        super();
        this.cardName = cardName;
        this.cardNumber = cardNumber;
        this.cardExpirationDate = cardExpirationDate;
    }

    // Method to provide payment details for credit card payment
    public void paymentDetails(double price) {
        amount -= price; // Deducting the price from the total amount
        System.out.println("Payment made with Credit Card.");
        System.out.println("Card Holder name is: " + cardName);
        System.out.println("Card Number is : " + cardNumber);
        System.out.println("Card Expiration Date is : " + cardExpirationDate);
        System.out.println("Remaining balance is: " + amount);
    }
}

```

```

}

// Class representing a person
class Person {
    private String personName; // Person's name
    private String personPhoneNumber; // Person's phone number

    // Constructor to initialize person's name and phone number
    public Person(String personName, String personPhoneNumber) {
        this.personName = personName;
        this.personPhoneNumber = personPhoneNumber;
    }

    // Method for a person to buy an item
    public void buyItem(Payment paymentMethod, String item, Westside
westside) {
        if (item.equals("clothes")) { // If the item to buy is clothes
            if (westside.purchaseClothes()) { // If clothes are available for
purchase
                paymentMethod.paymentDetails(Westside.PRICE_ITEM); //
Make payment
            } else {
                System.out.println("Clothes out of stock!"); // Print message if
clothes are out of stock
            }
        } else if (item.equals("accessories")) { // If the item to buy is
accessories
            if (westside.purchaseAccessories()) { // If accessories are available
for purchase
                paymentMethod.paymentDetails(Westside.PRICE_ITEM); //
Make payment
            } else {
                System.out.println("Accessories out of stock!"); // Print message
if accessories are out of stock
            }
        } else {
            System.out.println("Invalid item type!"); // Print message for invalid
item type
        }
    }
}

```

```

}

// Main class to test the functionality
public class MainWestside {
    public static void main(String[] args) {
        Westside westside = new Westside(); // Creating an instance of the
        Westside store

        Person[] persons = { // Creating an array of Person objects
            new Person("Alice", "123-456-7890"),
            new Person("Bob", "987-654-3210"),
            new Person("Charlie", "111-222-3333"),
            new Person("David", "444-555-6666"),
            new Person("Eve", "777-888-9999")
        };

        // Person 1 buys clothes with cash
        CashPayment cashPayment1 = new CashPayment();
        persons[0].buyItem(cashPayment1, "clothes", westside);

        // Person 2 buys accessories with cash
        CashPayment cashPayment2 = new CashPayment();
        persons[1].buyItem(cashPayment2, "accessories", westside);

        // Person 3 buys clothes with credit card
        CreditCardPayment creditCardPayment1 = new
        CreditCardPayment("John Doe", "12/26", "1234567890123456");
        persons[2].buyItem(creditCardPayment1, "clothes", westside);

        // Person 4 buys accessories with credit card
        CreditCardPayment creditCardPayment2 = new
        CreditCardPayment("Jane Smith", "11/25", "9876543210987654");
        persons[3].buyItem(creditCardPayment2, "accessories", westside);

        // Person 5 tries to buy clothes, but they are out of stock
        CashPayment cashPayment3 = new CashPayment();
        persons[4].buyItem(cashPayment3, "clothes", westside);
    }
}

```

**RESULT:**

```
lekh@lekh-lenovo: ~/Desktop/Lekh Nayak/exp 7
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7$ javac MainWestside.java
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7$ java MainWestside
Payment made in cash. Remaining balance is: 20000.0
Payment made in cash. Remaining balance is: 20000.0
Payment made with Credit Card.
Card Holder name is: John Doe
Card Number is : 12/26
Card Expiration Date is : 1234567890123456
Remaining balance is: 20000.0
Payment made with Credit Card.
Card Holder name is: Jane Smith
Card Number is : 11/25
Card Expiration Date is : 9876543210987654
Remaining balance is: 20000.0
Payment made in cash. Remaining balance is: 20000.0
lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7$
```

**Program 2****PROBLEM STATEMENT :**

Define a Westside class that has sales in clothes and accessories. Let us say 10 clothes and 10 accessories each cost 5000. Clothes and accessories are limited and updated as soon as purchase is done. Define a class named Payment (abstract class) that contains an instance variable of type double that stores the amount of the payment. Amount is initialized 25,000 and updated with each purchase. Also create a method named (abstract) paymentDetails that updates the amount of the payment. Next, define a class named CashPayment that is derived from Payment. This class should redefine the paymentDetails method to indicate that the payment is in cash. Include appropriate constructor(s)/methods. Define a class named CreditCardPayment that is derived from Payment. This class should contain instance variables for the name on the card, expiration date, and credit card number. Include appropriate constructor(s)/methods. Finally, redefine the paymentDetails method to include all credit card information in the printout. Define a class Person that contains person\_name and P\_phone\_no. Create a main method that creates at least five persons who will be given random chances for buying using any payment method CashPayment /CreditCardPayment. Once a person buys clothes/ accessories, the amount gets debited. Note that both CashPayment and CreditCardPayment will be derived from the Payment class.

**PROGRAM:**

```
import java.util.*;
```

```
// Abstract class representing a geometric shape
abstract class Shape {
    String color;

    // Constructor to initialize the color of the shape
    public Shape(String color) {
        this.color = color;
    }

    // Getter method to retrieve the color of the shape
    public String getColor() {
        return color;
    }

    // Setter method to set the color of the shape
    public void setColor(String color) {
        this.color = color;
    }

    // Abstract method to calculate the area of the shape
    public abstract double area();
}

// Class representing a rectangle, a subclass of Shape
class Rectangle extends Shape {
    private double length;
    private double width;

    // Constructor to initialize the color, length, and width of the rectangle
    public Rectangle(String color, double length, double width) {
        super(color);
        this.length = length;
        this.width = width;
    }

    // Getter method to retrieve the length of the rectangle
    public double getLength() {
        return length;
    }
}
```

```
// Setter method to set the length of the rectangle
public void setLength(double length) {
    this.length = length;
}

// Getter method to retrieve the width of the rectangle
public double getWidth() {
    return width;
}

// Setter method to set the width of the rectangle
public void setWidth(double width) {
    this.width = width;
}

// Method to calculate the area of the rectangle
public double area() {
    return length * width;
}
}

// Class representing a circle, a subclass of Shape
class Circle extends Shape {
    private double radius;

    // Constructor to initialize the color and radius of the circle
    public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
    }

    // Getter method to retrieve the radius of the circle
    public double getRadius() {
        return radius;
    }

    // Setter method to set the radius of the circle
    public void setRadius(double radius) {
        this.radius = radius;
    }
}
```



```

    }

    // Method to calculate the area of the circle
    public double area() {
        return Math.PI * radius * radius;
    }
}

// Class representing a diagram containing multiple shapes
class Diagram {
    private Shape[] shapes;

    // Constructor to initialize the diagram with a specified number of shapes
    public Diagram(int num) {
        shapes = new Shape[num]; // Create an array to store the shapes
        Scanner sc = new Scanner(System.in);
        for (int i = 0 ; i < shapes.length; i++) { // Iterate through each shape
            System.out.println("Enter 1 for Rectangle , Enter 2 for Circle , Enter
0 to exit program");
            int choice = sc.nextInt(); // Prompt user for choice
            if (choice == 1) { // If choice is to create a rectangle
                System.out.print("Enter color, length, and width for Rectangle " +
(i+1) + ": ");
                String color = sc.next(); // Read color input
                double length = sc.nextDouble(); // Read length input
                double width = sc.nextDouble(); // Read width input
                shapes[i] = new Rectangle(color, length, width); // Create a new
Rectangle object and add to the array
            }
            else if (choice == 2) { // If choice is to create a circle
                System.out.print("Enter color and radius for Circle " + (i+1) + ":
");
                String color = sc.next(); // Read color input
                double radius = sc.nextDouble(); // Read radius input
                shapes[i] = new Circle(color, radius); // Create a new Circle
object and add to the array
            }
            else if (choice == 0) { // If choice is to exit the program
                System.exit(0);
            }
        }
    }
}

```

```

        else { // If an invalid choice is made
            System.out.println("Invalid choice. Please enter 1 for Rectangle
or 2 for Circle.");
            i--; // Decrement i to repeat the loop for the current shape.
        }
    }
}

// Method to calculate the total area of all shapes in the diagram
public double totalArea() {
    double totalArea = 0;
    for (Shape shape : shapes) { // Iterate through each shape in the array
        totalArea += shape.area(); // Calculate the area of the shape and add
to totalArea
    }
    return totalArea; // Return the total area
}

// Main class to test the functionality of the Shape hierarchy and Diagram
public class TestShapes {
    public static void main(String[] args) {
        Diagram d1 = new Diagram(3); // Create a diagram with 3 shapes
        System.out.println(d1.totalArea()); // Print the total area of all shapes
in the diagram
    }
}

```

**RESULT:**

<pre> lekh@lekh-lenovo: ~/Desktop/Lekh Nayak/exp 7 lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7\$ javac TestShapes.java lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7\$ java TestShapes Enter 1 for Rectangle , Enter 2 for Circle , Enter 0 to exit program 1 Enter color, length, and width for Rectangle 1: Orange 5 6 Enter 1 for Rectangle , Enter 2 for Circle , Enter 0 to exit program 2 Enter color and radius for Circle 2: Blue 5 Enter 1 for Rectangle , Enter 2 for Circle , Enter 0 to exit program 1 Enter color, length, and width for Rectangle 3: green 7 8 164.53981633974485 lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7\$ java TestShapes Enter 1 for Rectangle , Enter 2 for Circle , Enter 0 to exit program 0 lekh@lekh-lenovo:~/Desktop/Lekh Nayak/exp 7\$ </pre>	
<b>CONCLUSION:</b>	In this experiment I learnt about using abstract classes to implement abstraction