

Experiment:4

1. class Cricket having data members name, age and member methods display() and setdata(). class Match inherits Cricket and has data members no_of_odi, no_of_test. Create an array of 5 objects of class Match. Provide all the required data through command line and display the information.

Solution :

```
import java.util.*;

class Cricket {
    String name;
    int age;
    public void setData(String name, int age) {
        this.name = name;
        this.age = age;
    }
    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

class Match extends Cricket {
    private int noOfOdi;
    private int noOfTest;
    public void setMatchData(String name, int age, int odi, int test) {
        setData(name, age);
        this.noOfOdi = odi;
        this.noOfTest = test;
    }
}
```

```
}

public void displayMatchData() {
    display();
    System.out.println("No. of ODIs: " + noOfOdi);
    System.out.println("No. of Test Matches: " + noOfTest);
    System.out.println();
}
}

public class CricketMatch {
    public static void main(String[] args) {
        if (args.length != 20) { // Each player requires 4 inputs: name, age, ODI, Test (5
            players * 4)
            System.out.println("Usage: java CricketMatch name1 age1 odi1 test1 ... name5
            age5 odi5 test5");
            return;
        }

        Match[] players = new Match[5];
        int index = 0;
        for (int i = 0; i < 5; i++) {
            String name = args[index];
            int age = Integer.parseInt(args[index + 1]);
            int odi = Integer.parseInt(args[index + 2]);
            int test = Integer.parseInt(args[index + 3]);
            players[i] = new Match();
            players[i].setMatchData(name, age, odi, test);
            index += 4;
        }
    }
}
```

```

    }

    System.out.println("\nDisplaying Player Information:");

    for (Match player : players) {
        player.displayMatchData();
    }
}
}

```

Output:

```

PS C:\12302130501036> javac CricketMatch.java
PS C:\12302130501036> java CricketMatch "Player1" 30 100 50 "Player2" 28 120 45 "Player3" 26 90 40 "Player4" 32 130 55 "Player5" 29 110 60

Displaying Player Information:
Name: Player1
Age: 30
No. of ODIs: 100
No. of Test Matches: 50

Name: Player2
Age: 28
No. of ODIs: 120
No. of Test Matches: 45

Name: Player3
Age: 26
No. of ODIs: 90
No. of Test Matches: 40

Name: Player4
Age: 32
No. of ODIs: 130
No. of Test Matches: 55

Name: Player5
Age: 29
No. of ODIs: 110
No. of Test Matches: 60

```

2. Define a class Cripher with following data**Field:****String plainText;****int key****Functions:****Cipher(String plaintext,int key)**

abstract String Encryption()

abstract String Decryption()

Derived two classes Substitution_Cipher and Caesar_Cipher override Encryption() and Decryption() Method. In substitution cipher every character of string is replaced with another character. For example. In this method you will replace the letters using the following scheme.

Plain Text: abcdefgh ij k l m n o p q r s t u v w x y z

Cipher Text: q a z w s x e d c r f v t g b y h n u j m i k o l p

So if string consists of letter "gcet" then encrypted string will be "ezsj" and decrypt it to get original string

In Caesar cipher encrypt the string same as program 5 of LAB 5.

Solution :

```
abstract class Cipher {
    protected String plainText;
    protected int key;
    public Cipher(String plainText, int key) {
        this.plainText = plainText;
        this.key = key;
    }
    abstract String Encryption();
    abstract String Decryption();
}
```

```
class Substitution_Cipher extends Cipher {
    private static final String PLAIN_ALPHABET = "abcdefghijklmnopqrstuvwxyz";
    private static final String CIPHER_ALPHABET = "qazwsxedcrfvtgbyhnujmikolp";
```

```
public Substitution_Cipher(String plainText, int key) {
    super(plainText, key);
}

@Override
public String Encryption() {
    char[] plainArray = plainText.toCharArray();
    for (int i = 0; i < plainArray.length; i++) {
        int index = PLAIN_ALPHABET.indexOf(plainArray[i]);
        if (index != -1) {
            plainArray[i] = CIPHER_ALPHABET.charAt(index);
        }
    }
    plainText = new String(plainArray);
    return plainText;
}

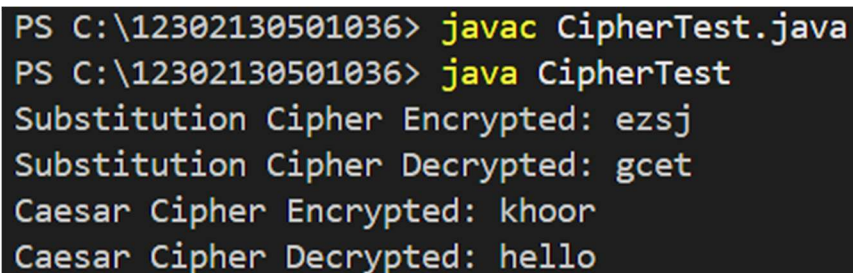
@Override
public String Decryption() {
    char[] cipherArray = plainText.toCharArray();
    for (int i = 0; i < cipherArray.length; i++) {
        int index = CIPHER_ALPHABET.indexOf(cipherArray[i]);
        if (index != -1) {
            cipherArray[i] = PLAIN_ALPHABET.charAt(index);
        }
    }
    plainText = new String(cipherArray);
}
```

```
        return plainText;
    }
}

class Caesar_Cipher extends Cipher {
    public Caesar_Cipher(String plainText, int key) {
        super(plainText, key);
    }
    @Override
    public String Encryption() {
        char[] plainArray = plainText.toCharArray();
        for (int i = 0; i < plainArray.length; i++) {
            plainArray[i] = (char) (plainArray[i] + this.key);
        }
        plainText = new String(plainArray);
        return plainText;
    }
    @Override
    public String Decryption() {
        char[] cipherArray = plainText.toCharArray();
        for (int i = 0; i < cipherArray.length; i++) {
            cipherArray[i] = (char) (cipherArray[i] - this.key);
        }
        plainText = new String(cipherArray);
        return plainText;
    }
}
```

```
}  
  
public class CipherTest {  
    public static void main(String[] args) {  
        Substitution_Cipher subCipher = new Substitution_Cipher("gcet", 0);  
        String encryptedSub = subCipher.Encryption();  
        System.out.println("Substitution Cipher Encrypted: " + encryptedSub);  
        subCipher = new Substitution_Cipher(encryptedSub, 0);  
        System.out.println("Substitution Cipher Decrypted: " + subCipher.Decryption());  
        Caesar_Cipher caesarCipher = new Caesar_Cipher("hello", 3);  
        String encryptedCaesar = caesarCipher.Encryption();  
        System.out.println("Caesar Cipher Encrypted: " + encryptedCaesar);  
        caesarCipher = new Caesar_Cipher(encryptedCaesar, 3);  
        System.out.println("Caesar Cipher Decrypted: " + caesarCipher.Decryption());  
    }  
}
```

Output:



```
PS C:\12302130501036> javac CipherTest.java  
PS C:\12302130501036> java CipherTest  
Substitution Cipher Encrypted: ezsj  
Substitution Cipher Decrypted: gcet  
Caesar Cipher Encrypted: koor  
Caesar Cipher Decrypted: hello
```

3. Declare an interface called Property containing a method computePrice to compute and return the price. The interface is to be implemented by following two classes i) Bungalow and ii) Flat. Both the classes have following data members

- **name**
- **constructionArea**

The class Bungalow has an additional data member called landArea. Define computePrice for both classes for computing total price. Use following rules for computing total price by summing up sub-costs:

Construction cost(for both classes):Rs.500/- per sq.feet

Additional cost (for Flat) : Rs. 200000/-

(for Bungalow): Rs. 200/- per sq. feet for landArea

Land cost (only for Bungalow): Rs. 400/- per sq. feet

Define method main to show usage of method computePrice.

Solution :

```
interface Property {  
    double computePrice();  
}  
  
class Bungalow implements Property {  
    String name;  
    double constructionArea;  
    double landArea;  
    public Bungalow(String name, double constructionArea, double landArea) {  
        this.name = name;  
        this.constructionArea = constructionArea;  
        this.landArea = landArea;  
    }  
    @Override  
    public double computePrice() {  
        double constructionCost = constructionArea * 500;  
        double additionalCost = landArea * 200;
```



```
        double landCost = landArea * 400;
        return constructionCost + additionalCost + landCost;
    }
}
```

```
class Flat implements Property {
    String name;
    double constructionArea;

    public Flat(String name, double constructionArea) {
        this.name = name;
        this.constructionArea = constructionArea;
    }

    @Override
    public double computePrice() {
        double constructionCost = constructionArea * 500;
        double additionalCost = 200000;
        return constructionCost + additionalCost;
    }
}
```

```
public class PropertyTest {
    public static void main(String[] args) {
        Bungalow bungalow = new Bungalow("Luxury Bungalow", 2500, 500);
        Flat flat = new Flat("City Apartment", 1200);
    }
}
```

```
        System.out.println("Price of " + bungalow.name + ": Rs. " +  
bungalow.computePrice());  
  
        System.out.println("Price of " + flat.name + ": Rs. " + flat.computePrice());  
    }  
}
```

Output:

```
PS C:\12302130501036> javac PropertyTest.java  
PS C:\12302130501036> java PropertyTest  
Price of Luxury Bungalow: Rs. 1550000.0  
Price of City Apartment: Rs. 800000.0
```

4. Define following classes and interfaces.

```
public interface GeometricShape {  
    public void describe();  
}  
  
public interface TwoDShape extends GeometricShape {  
    public double area();  
}  
  
public interface ThreeDShape extends GeometricShape {  
    public double volume();  
}  
  
public class Cone implements ThreeDShape {  
    private double radius;  
    private double height;  
    public Cone (double radius, double height)  
    public double volume()  
    public void describe()  
}
```

```
public class Rectangle implements TwoDShape {  
    private double width, height;  
    public Rectangle (double width, double height)  
    public double area()  
    public double perimeter()  
    public void describe()  
}
```

```
public class Sphere implements ThreeDShape {  
    private double radius;  
    public Sphere (double radius)  
    public double volume()  
    public void describe()  
}
```

Define test class to call various methods of Geometric Shape

Solution :

```
interface GeometricShape {  
    void describe();  
}  
  
interface TwoDShape extends GeometricShape {  
    double area();  
}  
  
interface ThreeDShape extends GeometricShape {  
    double volume();  
}  
  
class Cone implements ThreeDShape {
```

```
private double radius;
private double height;
public Cone(double radius, double height) {
    this.radius = radius;
    this.height = height;
}
@Override
public double volume() {
    return (1.0 / 3) * Math.PI * radius * radius * height;
}
@Override
public void describe() {
    System.out.println("This is a cone with radius " + radius + " and height " + height);
}
}

class Rectangle implements TwoDShape {
    private double width, height;
    public Rectangle(double width, double height) {
        this.width = width;
        this.height = height;
    }
    @Override
    public double area() {
        return width * height;
    }
    public double perimeter() {
```

```
        return 2 * (width + height);
    }

    @Override
    public void describe() {
        System.out.println("This is a rectangle with width " + width + " and height " +
height);
    }
}

class Sphere implements ThreeDShape {
    private double radius;

    public Sphere(double radius) {
        this.radius = radius;
    }

    @Override
    public double volume() {
        return (4.0 / 3) * Math.PI * Math.pow(radius, 3);
    }

    @Override
    public void describe() {
        System.out.println("This is a sphere with radius " + radius);
    }
}

public class GeometricTest {
    public static void main(String[] args) {
        Cone cone = new Cone(3, 5);
        Rectangle rectangle = new Rectangle(4, 6);
    }
}
```

```
Sphere sphere = new Sphere(3);
cone.describe();
System.out.println("Volume: " + cone.volume());
rectangle.describe();
System.out.println("Area: " + rectangle.area());
System.out.println("Perimeter: " + rectangle.perimeter());
sphere.describe();
System.out.println("Volume: " + sphere.volume());
}
}
```

Output:

```
PS C:\12302130501036> javac GeometricTest.java
PS C:\12302130501036> java GeometricTest
This is a cone with radius 3.0 and height 5.0
Volume: 47.12388980384689
This is a rectangle with width 4.0 and height 6.0
Area: 24.0
Perimeter: 20.0
This is a sphere with radius 3.0
Volume: 113.09733552923254
```