Name:	Debjit Ghosal
UID:	2023300065
Experiment No.	8A

AIM:	Interfaces
Program 1	
PROBLEM STATEMENT :	Consider two interfaces, Volume and SurfaceArea with methods getVolume() and getSurfaceArea() respectively. Class 'Cylinder' implements both Volume and SurfaceArea and implements their methods. The class contains their required dimensions as data members. Write a program which inputs its dimensions and prints its volume and surface area. Create classes 'Cone'and 'Sphere' that implements both the interfaces. In main class, ask user which shape volume and area needs to be calculated. Use switch case.

PROGRAM:	import java.util.Scanner;
	
	interface Volume {
	  double getVolume();
	}
	interface SurfaceArea {
	  double getSurfaceArea();
	}
	
	class Cylinder implements Volume, SurfaceArea {
	  private double radius;
	  private double height;
	
	  public Cylinder(double radius, double height) {
	  this.radius = radius;
	  this.height = height;
	  }

```
    @Override
    public double getVolume() {
        return Math.PI * radius * radius * height;
    }

    @Override
    public double getSurfaceArea() {
        return 2 * Math.PI * radius * (radius + height);
    }
}
class Cone implements Volume, SurfaceArea {
    private double radius;
    private double height;

    public Cone(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }
@Override
    public double getVolume() {
        return (Math.PI * radius * radius * height) / 3;
    }
@Override
    public double getSurfaceArea() {
        return Math.PI * radius * (radius + Math.sqrt(radius * radius + height
* height));
    }
}
class Sphere implements Volume, SurfaceArea {
    private double radius;

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

    @Override
    public double getVolume() {
        return (4 * Math.PI * radius * radius * radius) / 3;
    }
```

```
    @Override
    public double getSurfaceArea() {
        return 4 * Math.PI * radius * radius;
    }
}

public class Cylinder1 {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.println("Choose a shape (1: Cylinder, 2: Cone, 3:
Sphere):");
        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                System.out.println("Enter radius and height of the cylinder:");
                double cylinderRadius = scanner.nextDouble();
                double cylinderHeight = scanner.nextDouble();
                Cylinder cylinder = new Cylinder(cylinderRadius,
cylinderHeight);
                System.out.println("Volume of the cylinder: " +
cylinder.getVolume());
                System.out.println("Surface area of the cylinder: " +
cylinder.getSurfaceArea());
                break;

            case 2:
                System.out.println("Enter radius and height of the cone:");
                double coneRadius = scanner.nextDouble();
                double coneHeight = scanner.nextDouble();
                Cone cone = new Cone(coneRadius, coneHeight);
                System.out.println("Volume of the cone: " + cone.getVolume());
                System.out.println("Surface area of the cone: " +
cone.getSurfaceArea());
                break;

            case 3:
                System.out.println("Enter radius of the sphere:");
                double sphereRadius = scanner.nextDouble();
```

```
            Sphere sphere = new Sphere(sphereRadius);
            System.out.println("Volume of the sphere: " +
sphere.getVolume());
            System.out.println("Surface area of the sphere: " +
sphere.getSurfaceArea());
            break;
        default:
            System.out.println("Invalid choice!");
    }
  }
  @Override
  public String toString() {
    return "Cylinder1 []";
  }
}
```

RESULT:

```
 'C:\Users\DEBJIT GHOSAL\AppData\Roaming\Code\User\workspac
Choose a shape (1: Cylinder, 2: Cone, 3: Sphere):
1
Enter radius and height of the cylinder:
5  10
Volume of the cylinder: 785.3981633974483
Surface area of the cylinder: 471.23889803846896
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

```
 'C:\Users\DEBJIT GHOSAL\AppData\Roaming\Code\User\workspac
Choose a shape (1: Cylinder, 2: Cone, 3: Sphere):
2
Enter radius and height of the cone:
3     6
Volume of the cone: 56.548667764616276
Surface area of the cone: 91.49766646167468
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

```
 'C:\Users\DEBJIT GHOSAL\AppData\Roaming\Code\User\workspac
Choose a shape (1: Cylinder, 2: Cone, 3: Sphere):
3
Enter radius of the sphere:
5
Volume of the sphere: 523.5987755982989
Surface area of the sphere: 314.1592653589793
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

Program 2	
PROBLEM STATEMENT :	A banking system has two interfaces SavingAccount and CurrentAccount. The SavingAccount account has method getSimpleInterest() and CurrentAccont has method getCompoudInterest(). Both these interfaces are implemented by class Customer. Customer have data members: account type, interest rate and balance. The class then calculates interest on balance and prints it.

PROGRAM:	import java.util.Scanner;
	
	interface SavingAccount {
	  double getSimpleInterest();
	}
	
	interface CurrentAccount {
	  double getCompoundInterest();
	}
	
	class Customer implements SavingAccount, CurrentAccount {
	  private String accountType;
	  private double interestRate;
	  private double balance;
	
	  public Customer(String accountType, double interestRate, double balance) {
	  this.accountType = accountType;
	  this.interestRate = interestRate;
	  this.balance = balance;
	  }

```
    @Override
    public double getSimpleInterest() {
        // Calculate simple interest
        double simpleInterest = (balance * interestRate) / 100;
        return simpleInterest;
    }

    @Override
    public double getCompoundInterest() {
        // Assuming compound interest is calculated annually
        double compoundInterest = balance * Math.pow(1 + (interestRate /
100), 1) - balance;
        return compoundInterest;
    }

    public void printInterest() {
        if (accountType.equalsIgnoreCase("Saving")) {
            System.out.println("Simple interest on saving account: " +
getSimpleInterest());
        } else if (accountType.equalsIgnoreCase("Current")) {
            System.out.println("Compound interest on current account: " +
getCompoundInterest());
        } else {
            System.out.println("Invalid account type!");
        }
    }
}

public class BankingSystem {
    public static void main(String[] args) {
        Customer savingCustomer = new Customer("Saving", 5.0, 1000);
        Customer currentCustomer = new Customer("Current", 6.0, 1500);

        savingCustomer.printInterest();
        currentCustomer.printInterest();
    }
}
```

Write a class that implements the CharSequence interface. Your implementation should return the string backwards. Select one of the sentences to use as the data. Write a small main method to test your class.

RESULT:

```
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
er\workspaceStorage\9e74b28faa244b7149de54e209cd65d5\redhat
Simple interest on saving account: 50.0
Compound interest on current account: 90.0
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

Program 3

PROBLEM STATEMENT:

Write a class that implements the CharSequence interface. Your implementation should return the string backwards. Select one of the sentences to use as the data. Write a small main method to test your class.

PROGRAM:

```
import java.util.Scanner;

public class ReverseCharSequence implements CharSequence {
    private String str;

    public ReverseCharSequence(String str) {
        this.str = str;
    }
    @Override
    public int length() {
        return str.length();
    }
    @Override
    public char charAt(int index) {
        return str.charAt(str.length() - 1 - index);
    }
    @Override
    public CharSequence subSequence(int start, int end) {
        return new ReverseCharSequence(str.substring(start, end));
    }
    @Override
    public String toString() {
        StringBuilder reversed = new StringBuilder();
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed.append(str.charAt(i));
        }
        return reversed.toString();
    }
```

```
    // Small main method to test the class
    public static void main(String[] args) {
        String sentence = "Write a class that implements the CharSequence
interface.";
        ReverseCharSequence reversedSequence = new
ReverseCharSequence(sentence);
        System.out.println("Original: " + sentence);
        System.out.println("Reversed: " + reversedSequence);
    }
}
```

RESULT:

```
n\java.exe' '-cp' 'C:\Users\DEBJIT GHOSAL\AppData\Roaming\Code\User\wo
straction.Interfaces.ReverseCharSequence'
Original: Write a class that implements the CharSequence interface.
Reversed: .ecafretni ecneuqeSrahC eht stnemelpmi taht ssalc a etirW
PS C:\Users\DEBJIT GHOSAL\Desktop\SPIT_CODING\PSOOP\PSOOP>
```

CONCLUSION: I have learnt about abstraction,abstract class and interface.