Write a Java program that reads an integer between 0 and 1000 and adds all the digits in the integer.

Code:

```
import java.util.Scanner;

public class DigitSum {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      System.out.print("Enter an integer between 0 and 1000: ");
      int number = scanner.nextInt();

      if (number < 0 || number > 1000) {
         System.out.println("Invalid input! Please enter a number between 0 and 1000.");
      } else {
         int sum = 0;

         while (number > 0) {
            sum += number % 10;
            number /= 10;
         }

         System.out.println("The sum of the digits is: " + sum);
      }
      scanner.close();
   }
}
```

Write a Java program that reads an integer and outputs the number with its digits reversed.

```
import java.util.Scanner;

public class ReverseDigits {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Prompt the user to enter an integer
      System.out.print("Enter an integer: ");
      int number = scanner.nextInt();

      // Determine if the number is negative
      boolean isNegative = number < 0;

      // Make the number positive if it is negative
      if (isNegative) {
         number = -number;
      }
```

```
        int reversed = 0;

        // Reverse the digits of the number
        while (number > 0) {
            int digit = number % 10; // Extract the last digit
            reversed = reversed * 10 + digit; // Build the reversed number
            number /= 10; // Remove the last digit
        }

        // Restore the negative sign if the original number was negative
        if (isNegative) {
            reversed = -reversed;
        }

        // Display the reversed number
        System.out.println("The reversed number is: " + reversed);

        scanner.close();
    }
}
```

Write a Java program that reads an integer and counts the number of digits in the integer.

```
import java.util.Scanner;


public class CountDigits {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Prompt the user to enter an integer

        System.out.print("Enter an integer: ");

        int number = scanner.nextInt();


        // Handle negative numbers by taking the absolute value

        number = Math.abs(number);


        // Count the number of digits

        int digitCount = 0;


        if (number == 0) {
```

```
            digitCount = 1; // Special case: 0 has 1 digit
        } else {
            while (number > 0) {
                number /= 10; // Remove the last digit
                digitCount++;
            }
        }

        // Display the digit count
        System.out.println("The number of digits is: " + digitCount);

        scanner.close();
    }
}
```

Write a Java program that reads an integer and finds the largest digit in the integer.

```
import java.util.Scanner;

public class LargestDigit {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter an integer
        System.out.print("Enter an integer: ");
        int number = scanner.nextInt();

        // Handle negative numbers by taking the absolute value
        number = Math.abs(number);

        int largestDigit = 0;

        // Find the largest digit
```

```
        while (number > 0) {
            int digit = number % 10; // Extract the last digit
            if (digit > largestDigit) {
                largestDigit = digit; // Update the largest digit
            }
            number /= 10; // Remove the last digit
        }

        // Display the largest digit
        System.out.println("The largest digit is: " + largestDigit);

        scanner.close();
    }
}
```

Write a Java program that checks whether an input string is a palindrome.

```
import java.util.Scanner;

public class PalindromeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        // Remove any non-alphanumeric characters and convert to lowercase
        String cleanedInput = input.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

        // Check if the string is a palindrome
        boolean isPalindrome = true;
        int length = cleanedInput.length();
```

```
        for (int i = 0; i < length / 2; i++) {
            if (cleanedInput.charAt(i) != cleanedInput.charAt(length - 1 - i)) {
                isPalindrome = false;
                break;
            }
        }

        // Display the result
        if (isPalindrome) {
            System.out.println("The string is a palindrome.");
        } else {
            System.out.println("The string is not a palindrome.");
        }

        scanner.close();
    }
}
```

Write a Java program that reads a string and counts the number of vowels and consonants in the string

```
import java.util.Scanner;

public class VowelConsonantCounter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        // Convert the string to lowercase for easier comparison
```

```
        String lowerCaseInput = input.toLowerCase();

        // Initialize counters for vowels and consonants
        int vowelCount = 0;
        int consonantCount = 0;

        // Iterate through each character in the string
        for (int i = 0; i < lowerCaseInput.length(); i++) {
            char ch = lowerCaseInput.charAt(i);

            // Check if the character is a letter
            if (Character.isLetter(ch)) {
                // Check if the character is a vowel
                if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                    vowelCount++;
                } else {
                    consonantCount++;
                }
            }
        }

        // Display the results
        System.out.println("Number of vowels: " + vowelCount);
        System.out.println("Number of consonants: " + consonantCount);

        scanner.close();
    }
}
```

Write a Java program that reads a sentence and finds the longest word in it.

```
import java.util.Scanner;
```

```
public class LongestWordFinder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a sentence
        System.out.print("Enter a sentence: ");
        String sentence = scanner.nextLine();

        // Split the sentence into words
        String[] words = sentence.split("\\s+");

        String longestWord = "";
        int maxLength = 0;

        // Find the longest word
        for (String word : words) {
            // Remove any non-alphanumeric characters from the word
            word = word.replaceAll("[^a-zA-Z0-9]", "");

            if (word.length() > maxLength) {
                longestWord = word;
                maxLength = word.length();
            }
        }

        // Display the longest word
        System.out.println("The longest word is: " + longestWord);

        scanner.close();
    }
}
```

Write a Java program that checks whether two input strings are anagrams of each other.

```
import java.util.Scanner;
import java.util.Arrays;

public class AnagramChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter two strings
        System.out.print("Enter the first string: ");
        String firstString = scanner.nextLine();

        System.out.print("Enter the second string: ");
        String secondString = scanner.nextLine();

        // Check if the two strings are anagrams
        if (areAnagrams(firstString, secondString)) {
            System.out.println("The strings are anagrams.");
        } else {
            System.out.println("The strings are not anagrams.");
        }

        scanner.close();
    }

    public static boolean areAnagrams(String str1, String str2) {
        // Remove spaces and non-alphanumeric characters, and convert to lowercase
        str1 = str1.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();
        str2 = str2.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();

        // Check if lengths are different
```

```
        if (str1.length() != str2.length()) {
            return false;
        }

        // Convert strings to character arrays
        char[] chars1 = str1.toCharArray();
        char[] chars2 = str2.toCharArray();

        // Sort the character arrays
        Arrays.sort(chars1);
        Arrays.sort(chars2);

        // Compare sorted arrays
        return Arrays.equals(chars1, chars2);
    }
}
```

Write a Java program that reads a string and counts the number of words in the string

```
import java.util.Scanner;

public class WordCounter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        // Trim leading and trailing spaces and split the string into words
        String[] words = input.trim().split("\\s+");

        // Count the number of words
```

```
        int wordCount = (input.trim().isEmpty()) ? 0 : words.length;

        // Display the word count
        System.out.println("The number of words in the string is: " + wordCount);

        scanner.close();
    }
}
```

Write a Java program that reads a string and removes all duplicate characters, leaving only the first occurrence.

```
import java.util.Scanner;

public class RemoveDuplicates {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter a string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        // Remove duplicate characters
        String result = removeDuplicates(input);

        // Display the result
        System.out.println("String after removing duplicates: " + result);

        scanner.close();
    }

    public static String removeDuplicates(String str) {
        StringBuilder result = new StringBuilder();
```

```
        boolean[] seen = new boolean[256]; // Array to track character occurrences

        // Iterate through each character in the string
        for (char c : str.toCharArray()) {
            if (!seen[c]) { // If the character has not been seen before
                seen[c] = true; // Mark it as seen
                result.append(c); // Append it to the result
            }
        }

        return result.toString();
    }
}
```

WriteaJavaprogramtocreateaclassStudentwithdataname,cityandagealongwith methodaddDataandprintDatatoinputanddisplaythedata.Createthetwoobjectss1, s2 to declare and access the values. (CLASS AND OBJECT)

```
import java.util.Scanner;

class Student {
    // Data fields
    String name;
    String city;
    int age;

    // Method to add data
    public void addData(String name, String city, int age) {
        this.name = name;
        this.city = city;
        this.age = age;
    }

    // Method to print data
```

```
    public void printData() {
        System.out.println("Student Name: " + name);
        System.out.println("City: " + city);
        System.out.println("Age: " + age);
    }
}

public class StudentData {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create two Student objects
        Student s1 = new Student();
        Student s2 = new Student();

        // Input and add data for s1
        System.out.println("Enter details for student 1:");
        System.out.print("Enter name: ");
        String name1 = scanner.nextLine();
        System.out.print("Enter city: ");
        String city1 = scanner.nextLine();
        System.out.print("Enter age: ");
        int age1 = scanner.nextInt();
        scanner.nextLine();  // Consume the leftover newline
        s1.addData(name1, city1, age1);

        // Input and add data for s2
        System.out.println("Enter details for student 2:");
        System.out.print("Enter name: ");
        String name2 = scanner.nextLine();
        System.out.print("Enter city: ");
```

```
        String city2 = scanner.nextLine();
        System.out.print("Enter age: ");
        int age2 = scanner.nextInt();
        scanner.close();
        s2.addData(name2, city2, age2);

        // Display the data of both students
        System.out.println("\nStudent 1 details:");
        s1.printData();

        System.out.println("\nStudent 2 details:");
        s2.printData();
    }
}
```

WriteaJavaprogramtocreateaclassProductwithdatamembersproductName,category, and price. Implement methods addData to input the data and printData to display the

```
import java.util.Scanner;

class Product {
    // Data members
    String productName;
    String category;
    double price;

    // Method to input data
    public void addData(String productName, String category, double price) {
        this.productName = productName;
        this.category = category;
        this.price = price;
    }
```

```
    // Method to display data
    public void printData() {
        System.out.println("Product Name: " + productName);
        System.out.println("Category: " + category);
        System.out.println("Price: $" + price);
    }
}

public class ProductData {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create a Product object
        Product product1 = new Product();

        // Input data for product1
        System.out.println("Enter product details:");
        System.out.print("Enter product name: ");
        String productName = scanner.nextLine();
        System.out.print("Enter product category: ");
        String category = scanner.nextLine();
        System.out.print("Enter product price: ");
        double price = scanner.nextDouble();

        // Add data to product1 using addData method
        product1.addData(productName, category, price);

        // Display product details using printData method
        System.out.println("\nProduct details:");
        product1.printData();
```

```
        scanner.close();
    }
}
```

WriteaJavaprogramtocreateaclassLaptopwithdatamembersbrand,model, andprice. Implement methods addData to input the data and printData to display the data.

```
import java.util.Scanner;

class Laptop {
    // Data members
    String brand;
    String model;
    double price;

    // Method to input data
    public void addData(String brand, String model, double price) {
        this.brand = brand;
        this.model = model;
        this.price = price;
    }

    // Method to display data
    public void printData() {
        System.out.println("Laptop Brand: " + brand);
        System.out.println("Laptop Model: " + model);
        System.out.println("Laptop Price: $" + price);
    }
}

public class LaptopData {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        // Create a Laptop object
        Laptop laptop1 = new Laptop();

        // Input data for laptop1
        System.out.println("Enter laptop details:");
        System.out.print("Enter brand: ");
        String brand = scanner.nextLine();
        System.out.print("Enter model: ");
        String model = scanner.nextLine();
        System.out.print("Enter price: ");
        double price = scanner.nextDouble();

        // Add data to laptop1 using addData method
        laptop1.addData(brand, model, price);

        // Display laptop details using printData method
        System.out.println("\nLaptop details:");
        laptop1.printData();

        scanner.close();
    }
}
```

WriteaJavaprogramtocreateaclassBankAccountwithdatamembersaccountNumber, accountHolderName, and balance. Implement methods addData to input the data and printData to display the data.

```
import java.util.Scanner;

class BankAccount {
    // Data members
    String accountNumber;
    String accountHolderName;
```

```
    double balance;

    // Method to input data
    public void addData(String accountNumber, String accountHolderName, double balance) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = balance;
    }

    // Method to display data
    public void printData() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Account Holder Name: " + accountHolderName);
        System.out.println("Balance: $" + balance);
    }
}

public class BankAccountData {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Create a BankAccount object
        BankAccount account1 = new BankAccount();

        // Input data for account1
        System.out.println("Enter bank account details:");
        System.out.print("Enter account number: ");
        String accountNumber = scanner.nextLine();
        System.out.print("Enter account holder name: ");
        String accountHolderName = scanner.nextLine();
        System.out.print("Enter balance: ");
```

```
        double balance = scanner.nextDouble();

        // Add data to account1 using addData method
        account1.addData(accountNumber, accountHolderName, balance);

        // Display account details using printData method
        System.out.println("\nBank account details:");
        account1.printData();

        scanner.close();
    }
}
```

WriteaJavaprogramthatcreatesaclasshierarchyforemployeesofacompany.The baseclassshouldbeEmployee,withsubclassesManager,Developer,andProgrammer. Each subclass shouldhaveproperties suchas name, address, salary, and job title. Implementmethods for calculating bonuses, generating performance reports, and managing projects.(ENCAPSULATION)

```
// Base class
class Employee {
    // Encapsulation: private data members
    private String name;
    private String address;
    private double salary;
    private String jobTitle;

    // Constructor
    public Employee(String name, String address, double salary, String jobTitle) {
        this.name = name;
        this.address = address;
        this.salary = salary;
        this.jobTitle = jobTitle;
    }
```

```
// Getter and Setter methods for encapsulation
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public String getJobTitle() {
    return jobTitle;
}

public void setJobTitle(String jobTitle) {
    this.jobTitle = jobTitle;
```

```
    }

    // Method to calculate bonus (default behavior for all employees)
    public double calculateBonus() {
        return salary * 0.05; // 5% bonus for all employees
    }

    // Method to generate a performance report
    public void generatePerformanceReport() {
        System.out.println("Performance report for " + name + ":");
        System.out.println("Job Title: " + jobTitle);
        System.out.println("Salary: $" + salary);
    }
}

// Manager subclass
class Manager extends Employee {
    private int teamSize;

    // Constructor
    public Manager(String name, String address, double salary, String jobTitle, int teamSize) {
        super(name, address, salary, jobTitle);
        this.teamSize = teamSize;
    }

    // Getter and Setter for teamSize
    public int getTeamSize() {
        return teamSize;
    }

    public void setTeamSize(int teamSize) {
```

```
        this.teamSize = teamSize;
    }

    // Overriding the calculateBonus method for Manager
    @Override
    public double calculateBonus() {
        return getSalary() * 0.10; // 10% bonus for Managers
    }

    // Method to manage projects
    public void manageProject() {
        System.out.println("Manager " + getName() + " is managing the project with a team of " +
teamSize + " members.");
    }

    // Overriding performance report for Manager
    @Override
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
        System.out.println("Team Size: " + teamSize);
    }
}

// Developer subclass
class Developer extends Employee {
    private String programmingLanguage;

    // Constructor
    public Developer(String name, String address, double salary, String jobTitle, String
programmingLanguage) {
        super(name, address, salary, jobTitle);
        this.programmingLanguage = programmingLanguage;
```

```
    }

    // Getter and Setter for programmingLanguage
    public String getProgrammingLanguage() {
        return programmingLanguage;
    }

    public void setProgrammingLanguage(String programmingLanguage) {
        this.programmingLanguage = programmingLanguage;
    }

    // Overriding the calculateBonus method for Developer
    @Override
    public double calculateBonus() {
        return getSalary() * 0.07; // 7% bonus for Developers
    }

    // Method to work on a project
    public void workOnProject() {
        System.out.println("Developer " + getName() + " is working on the project using " +
programmingLanguage + ".");
    }

    // Overriding performance report for Developer
    @Override
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
        System.out.println("Programming Language: " + programmingLanguage);
    }
}
```

```
// Programmer subclass
class Programmer extends Employee {
    private String expertiseArea;

    // Constructor
    public Programmer(String name, String address, double salary, String jobTitle, String expertiseArea) {
        super(name, address, salary, jobTitle);
        this.expertiseArea = expertiseArea;
    }

    // Getter and Setter for expertiseArea
    public String getExpertiseArea() {
        return expertiseArea;
    }

    public void setExpertiseArea(String expertiseArea) {
        this.expertiseArea = expertiseArea;
    }

    // Overriding the calculateBonus method for Programmer
    @Override
    public double calculateBonus() {
        return getSalary() * 0.08; // 8% bonus for Programmers
    }

    // Method to write code
    public void writeCode() {
        System.out.println("Programmer " + getName() + " is writing code in the area of " + expertiseArea + ".");
    }
```

```
    // Overriding performance report for Programmer
    @Override
    public void generatePerformanceReport() {
        super.generatePerformanceReport();
        System.out.println("Expertise Area: " + expertiseArea);
    }
}

public class Company {
    public static void main(String[] args) {
        // Creating objects for each type of employee
        Manager manager = new Manager("Alice", "123 Main St", 80000, "Manager", 5);
        Developer developer = new Developer("Bob", "456 Oak St", 70000, "Developer", "Java");
        Programmer programmer = new Programmer("Charlie", "789 Pine St", 65000, "Programmer",
"AI");

        // Display performance reports and calculate bonuses for each employee
        manager.generatePerformanceReport();
        System.out.println("Bonus: $" + manager.calculateBonus());
        manager.manageProject();

        System.out.println("\n");

        developer.generatePerformanceReport();
        System.out.println("Bonus: $" + developer.calculateBonus());
        developer.workOnProject();

        System.out.println("\n");

        programmer.generatePerformanceReport();
        System.out.println("Bonus: $" + programmer.calculateBonus());
```

```
        programmer.writeCode();
    }
}
```

WriteaJavaprogramtocreateaclasshierarchyforane-commercesystem.Thebaseclass shouldbeProduct,withsubclassesElectronics,Clothing, andFood.Eachsubclass should have properties such as price, brand, expiryDate, andwarranty. Implementmethods for calculating discounts, managing inventory, and processing orders.

```
import java.util.Date;

// Base class
class Product {
    private double price;
    private String brand;
    private String expiryDate;  // For Food items
    private int warranty;       // In months (for Electronics and Clothing)

    // Constructor for Product
    public Product(double price, String brand, String expiryDate, int warranty) {
        this.price = price;
        this.brand = brand;
        this.expiryDate = expiryDate;
        this.warranty = warranty;
    }

    // Getter and Setter methods (Encapsulation)
    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
```

```
public String getBrand() {
    return brand;
}

public void setBrand(String brand) {
    this.brand = brand;
}

public String getExpiryDate() {
    return expiryDate;
}

public void setExpiryDate(String expiryDate) {
    this.expiryDate = expiryDate;
}

public int getWarranty() {
    return warranty;
}

public void setWarranty(int warranty) {
    this.warranty = warranty;
}

// Method to calculate discount (Base method, to be overridden)
public double calculateDiscount() {
    return 0;  // No discount by default for generic product
}

// Method to manage inventory (Base method, to be overridden)
```

```
    public void manageInventory(int quantity) {
        System.out.println("Managing inventory for " + quantity + " items of " + brand + ".");
    }

    // Method to process an order (Base method, to be overridden)
    public void processOrder(int quantity) {
        System.out.println("Processing order for " + quantity + " " + brand + " product(s).");
    }
}

// Electronics subclass
class Electronics extends Product {
    private String model;
    private String specifications;

    // Constructor for Electronics
    public Electronics(double price, String brand, String expiryDate, int warranty, String model, String
specifications) {
        super(price, brand, expiryDate, warranty);
        this.model = model;
        this.specifications = specifications;
    }

    // Getter and Setter methods for Electronics-specific properties
    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }
```

```
    public String getSpecifications() {
        return specifications;
    }

    public void setSpecifications(String specifications) {
        this.specifications = specifications;
    }

    // Overriding the calculateDiscount method for Electronics
    @Override
    public double calculateDiscount() {
        return getPrice() * 0.10;  // 10% discount for Electronics
    }

    // Overriding manageInventory method for Electronics
    @Override
    public void manageInventory(int quantity) {
        System.out.println("Managing inventory for " + quantity + " units of " + getBrand() + " " + model
+ " electronics.");
    }

    // Overriding processOrder method for Electronics
    @Override
    public void processOrder(int quantity) {
        System.out.println("Processing order for " + quantity + " unit(s) of " + getBrand() + " " + model +
" electronics.");
    }
}

// Clothing subclass
class Clothing extends Product {
```

```
    private String size;
    private String material;

    // Constructor for Clothing
    public Clothing(double price, String brand, String expiryDate, int warranty, String size, String material) {
        super(price, brand, expiryDate, warranty);
        this.size = size;
        this.material = material;
    }

    // Getter and Setter methods for Clothing-specific properties
    public String getSize() {
        return size;
    }

    public void setSize(String size) {
        this.size = size;
    }

    public String getMaterial() {
        return material;
    }

    public void setMaterial(String material) {
        this.material = material;
    }

    // Overriding the calculateDiscount method for Clothing
    @Override
    public double calculateDiscount() {
```

```
        return getPrice() * 0.15;  // 15% discount for Clothing
    }

    // Overriding manageInventory method for Clothing
    @Override
    public void manageInventory(int quantity) {
        System.out.println("Managing inventory for " + quantity + " items of " + getBrand() + " clothing, size " + size + ".");
    }

    // Overriding processOrder method for Clothing
    @Override
    public void processOrder(int quantity) {
        System.out.println("Processing order for " + quantity + " unit(s) of " + getBrand() + " clothing, size " + size + ".");
    }
}

// Food subclass
class Food extends Product {
    private Date manufactureDate;
    private Date expiryDate;

    // Constructor for Food
    public Food(double price, String brand, String expiryDate, int warranty, Date manufactureDate, Date expiryDate) {
        super(price, brand, expiryDate, warranty);
        this.manufactureDate = manufactureDate;
        this.expiryDate = expiryDate;
    }

    // Getter and Setter methods for Food-specific properties
```

```
public Date getManufactureDate() {
    return manufactureDate;
}

public void setManufactureDate(Date manufactureDate) {
    this.manufactureDate = manufactureDate;
}

public Date getExpiryDate() {
    return expiryDate;
}

public void setExpiryDate(Date expiryDate) {
    this.expiryDate = expiryDate;
}

// Overriding the calculateDiscount method for Food
@Override
public double calculateDiscount() {
    return getPrice() * 0.05;  // 5% discount for Food
}

// Overriding manageInventory method for Food
@Override
public void manageInventory(int quantity) {
    System.out.println("Managing inventory for " + quantity + " items of " + getBrand() + " food.");
}

// Overriding processOrder method for Food
@Override
public void processOrder(int quantity) {
```

```
        System.out.println("Processing order for " + quantity + " unit(s) of " + getBrand() + " food.");
    }
}

public class ECommerceSystem {
    public static void main(String[] args) {
        // Create sample products for each category
        Electronics laptop = new Electronics(1500, "Dell", "2025-12-31", 24, "XPS 15", "Intel i7, 16GB RAM, 512GB SSD");
        Clothing tshirt = new Clothing(25, "Nike", "2026-01-01", 12, "M", "Cotton");
        Food apple = new Food(2, "Fresh Farms", "2024-11-30", 0, new Date(), new Date());

        // Display product details, calculate discount, manage inventory, and process orders
        System.out.println("Electronics Discount: $" + laptop.calculateDiscount());
        laptop.manageInventory(50);
        laptop.processOrder(5);

        System.out.println("\nClothing Discount: $" + tshirt.calculateDiscount());
        tshirt.manageInventory(200);
        tshirt.processOrder(10);

        System.out.println("\nFood Discount: $" + apple.calculateDiscount());
        apple.manageInventory(100);
        apple.processOrder(30);
    }
}
```

WriteaJavaprogramtocreateaclasshierarchyforitemsinalibrary.Thebaseclassshould beLibraryItem,with subclassesBook,Magazine, andDVD. Each subclass shouldhave propertiessuchastitle,author,publicationYear,andgenre.Implementmethodsforchecking out items, returning items, and calculating late fees.

```
// Base class
class LibraryItem {
```

```
private String title;
private String author;
private int publicationYear;
private String genre;

// Constructor
public LibraryItem(String title, String author, int publicationYear, String genre) {
    this.title = title;
    this.author = author;
    this.publicationYear = publicationYear;
    this.genre = genre;
}

// Getter and Setter methods
public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public int getPublicationYear() {
```

```
        return publicationYear;
    }

    public void setPublicationYear(int publicationYear) {
        this.publicationYear = publicationYear;
    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    // Method to check out an item
    public void checkOut() {
        System.out.println(title + " has been checked out.");
    }

    // Method to return an item
    public void returnItem() {
        System.out.println(title + " has been returned.");
    }

    // Method to calculate late fees (default, overridden in subclasses)
    public double calculateLateFee(int daysLate) {
        return 0.0; // No default late fee
    }
}
```

```
// Book subclass
class Book extends LibraryItem {
    private int pageCount;

    // Constructor
    public Book(String title, String author, int publicationYear, String genre, int pageCount) {
        super(title, author, publicationYear, genre);
        this.pageCount = pageCount;
    }

    // Getter and Setter for pageCount
    public int getPageCount() {
        return pageCount;
    }

    public void setPageCount(int pageCount) {
        this.pageCount = pageCount;
    }

    // Overriding calculateLateFee
    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 0.50; // $0.50 per day for books
    }
}

// Magazine subclass
class Magazine extends LibraryItem {
    private int issueNumber;

    // Constructor
```

```
    public Magazine(String title, String author, int publicationYear, String genre, int issueNumber) {
        super(title, author, publicationYear, genre);
        this.issueNumber = issueNumber;
    }

    // Getter and Setter for issueNumber
    public int getIssueNumber() {
        return issueNumber;
    }

    public void setIssueNumber(int issueNumber) {
        this.issueNumber = issueNumber;
    }

    // Overriding calculateLateFee
    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 0.25; // $0.25 per day for magazines
    }
}

// DVD subclass
class DVD extends LibraryItem {
    private int duration; // Duration in minutes

    // Constructor
    public DVD(String title, String author, int publicationYear, String genre, int duration) {
        super(title, author, publicationYear, genre);
        this.duration = duration;
    }
```

```
    // Getter and Setter for duration
    public int getDuration() {
        return duration;
    }

    public void setDuration(int duration) {
        this.duration = duration;
    }

    // Overriding calculateLateFee
    @Override
    public double calculateLateFee(int daysLate) {
        return daysLate * 1.00; // $1.00 per day for DVDs
    }
}

// Main class
public class LibrarySystem {
    public static void main(String[] args) {
        // Create objects for each type of library item
        Book book = new Book("1984", "George Orwell", 1949, "Dystopian", 328);
        Magazine magazine = new Magazine("Time", "Various Authors", 2023, "News", 52);
        DVD dvd = new DVD("Inception", "Christopher Nolan", 2010, "Science Fiction", 148);

        // Check out items
        book.checkOut();
        magazine.checkOut();
        dvd.checkOut();

        System.out.println();
```

```
        // Calculate late fees
        System.out.println("Late fee for the book: $" + book.calculateLateFee(5)); // 5 days late
        System.out.println("Late fee for the magazine: $" + magazine.calculateLateFee(3)); // 3 days late
        System.out.println("Late fee for the DVD: $" + dvd.calculateLateFee(7)); // 7 days late

        System.out.println();

        // Return items
        book.returnItem();
        magazine.returnItem();
        dvd.returnItem();
    }
}
```

WriteaJavaprogramtocreateaclasshierarchyforbankaccounts.Thebaseclassshouldbe Account, with subclasses SavingsAccount, CheckingAccount, andFixedDepositAccount. Each subclass shouldhaveproperties suchas accountNumber, balance, interestRate, and accountHolderName.Implementmethodsforcalculatinginterest,managingtransactions,and generating account statements.

```
// Base class
class Account {
    protected String accountNumber;
    protected String accountHolderName;
    protected double balance;
    protected double interestRate;

    // Constructor
    public Account(String accountNumber, String accountHolderName, double balance, double
interestRate) {
        this.accountNumber = accountNumber;
        this.accountHolderName = accountHolderName;
        this.balance = balance;
        this.interestRate = interestRate;
    }
```

```
// Getter and Setter methods
public String getAccountNumber() {
    return accountNumber;
}

public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

public String getAccountHolderName() {
    return accountHolderName;
}

public void setAccountHolderName(String accountHolderName) {
    this.accountHolderName = accountHolderName;
}

public double getBalance() {
    return balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}

public double getInterestRate() {
    return interestRate;
}

public void setInterestRate(double interestRate) {
```

```
    this.interestRate = interestRate;
}

// Method to calculate interest (default, overridden in subclasses)
public double calculateInterest() {
    return balance * (interestRate / 100);
}

// Method to deposit money
public void deposit(double amount) {
    if (amount > 0) {
        balance += amount;
        System.out.println("Deposited: $" + amount + " | New Balance: $" + balance);
    } else {
        System.out.println("Invalid deposit amount.");
    }
}

// Method to withdraw money (default, overridden in CheckingAccount)
public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
        System.out.println("Withdrew: $" + amount + " | New Balance: $" + balance);
    } else {
        System.out.println("Invalid withdrawal amount or insufficient balance.");
    }
}

// Method to generate account statement
public void generateAccountStatement() {
    System.out.println("Account Statement for " + accountHolderName);
```

```
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: $" + balance);
    }
}

// SavingsAccount subclass
class SavingsAccount extends Account {
    private final double minimumBalance;

    // Constructor
    public SavingsAccount(String accountNumber, String accountHolderName, double balance, double interestRate, double minimumBalance) {
        super(accountNumber, accountHolderName, balance, interestRate);
        this.minimumBalance = minimumBalance;
    }

    // Method to calculate interest (specific to savings account)
    @Override
    public double calculateInterest() {
        return super.calculateInterest();
    }

    // Method to withdraw money (with minimum balance restriction)
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= minimumBalance) {
            super.withdraw(amount);
        } else {
            System.out.println("Cannot withdraw. Minimum balance of $" + minimumBalance + " must be maintained.");
        }
    }
```

```
}

// CheckingAccount subclass
class CheckingAccount extends Account {
    private final double overdraftLimit;

    // Constructor
    public CheckingAccount(String accountNumber, String accountHolderName, double balance,
double interestRate, double overdraftLimit) {
        super(accountNumber, accountHolderName, balance, interestRate);
        this.overdraftLimit = overdraftLimit;
    }

    // Method to withdraw money (with overdraft limit)
    @Override
    public void withdraw(double amount) {
        if (balance - amount >= -overdraftLimit) {
            balance -= amount;
            System.out.println("Withdrew: $" + amount + " | New Balance: $" + balance);
        } else {
            System.out.println("Cannot withdraw. Overdraft limit of $" + overdraftLimit + " exceeded.");
        }
    }

    // Checking accounts typically do not have interest
    @Override
    public double calculateInterest() {
        return 0.0;
    }
}
```

```
// FixedDepositAccount subclass
class FixedDepositAccount extends Account {
    private final int depositTerm; // in months

    // Constructor
    public FixedDepositAccount(String accountNumber, String accountHolderName, double balance, double interestRate, int depositTerm) {
        super(accountNumber, accountHolderName, balance, interestRate);
        this.depositTerm = depositTerm;
    }

    // Method to calculate interest (specific to fixed deposit account)
    @Override
    public double calculateInterest() {
        return balance * (interestRate / 100) * (depositTerm / 12.0);
    }

    // Fixed deposits usually do not allow withdrawals
    @Override
    public void withdraw(double amount) {
        System.out.println("Withdrawals are not allowed for Fixed Deposit Accounts.");
    }
}

// Main class
public class BankSystem {
    public static void main(String[] args) {
        // Create accounts
        SavingsAccount savings = new SavingsAccount("SA123", "Alice", 5000, 3.5, 1000);
        CheckingAccount checking = new CheckingAccount("CA456", "Bob", 2000, 0, 500);
        FixedDepositAccount fixedDeposit = new FixedDepositAccount("FD789", "Charlie", 10000, 5.0, 12);
```

```
        // Perform operations on Savings Account
        System.out.println("Savings Account:");
        savings.deposit(1000);
        savings.withdraw(4500);
        savings.withdraw(200);
        System.out.println("Interest: $" + savings.calculateInterest());
        savings.generateAccountStatement();

        System.out.println("\nChecking Account:");
        checking.deposit(500);
        checking.withdraw(2500);
        checking.withdraw(1000);
        System.out.println("Interest: $" + checking.calculateInterest());
        checking.generateAccountStatement();

        System.out.println("\nFixed Deposit Account:");
        fixedDeposit.deposit(2000); // Should work as it adds to principal
        fixedDeposit.withdraw(500); // Should not allow withdrawal
        System.out.println("Interest: $" + fixedDeposit.calculateInterest());
        fixedDeposit.generateAccountStatement();
    }
}
```

Write a javaprogramto create anabstract class namedShape that contains two integers andan emptymethodnamedprintArea (). Provide three classes named Rectangle,TriangleandCirclesuchthateachoneoftheclassesextendstheclassShape. Eachoneof theclassescontainsonlythemethodprintArea() thatprintstheareaof the given shape (ABSTRACT CLASS)

```
// Abstract class Shape
abstract class Shape {
    protected int dimension1;
    protected int dimension2;
```

```
    // Constructor
    public Shape(int dimension1, int dimension2) {
        this.dimension1 = dimension1;
        this.dimension2 = dimension2;
    }

    // Abstract method
    public abstract void printArea();
}

// Rectangle class extends Shape
class Rectangle extends Shape {
    // Constructor
    public Rectangle(int length, int width) {
        super(length, width);
    }

    // Implementing printArea for Rectangle
    @Override
    public void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Rectangle Area: " + area);
    }
}

// Triangle class extends Shape
class Triangle extends Shape {
    // Constructor
    public Triangle(int base, int height) {
        super(base, height);
```

```
    }

    // Implementing printArea for Triangle
    @Override
    public void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Triangle Area: " + area);
    }
}

// Circle class extends Shape
class Circle extends Shape {
    // Constructor
    public Circle(int radius) {
        super(radius, 0); // Second dimension is unused for Circle
    }

    // Implementing printArea for Circle
    @Override
    public void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Circle Area: " + area);
    }
}

// Main class
public class ShapeTest {
    public static void main(String[] args) {
        // Create instances of each shape
        Shape rectangle = new Rectangle(10, 5);
        Shape triangle = new Triangle(10, 8);
```

```
        Shape circle = new Circle(7);

        // Print areas
        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
    }
}
```

WriteaJavaprogramtocreateanabstractclassVehiclethatcontainsproperties likespeed and fuelCapacity, and an abstract method calculateRange(). Provide subclasses Car, Motorcycle, andTruck, each implementing the calculateRange()method tocalculate the range based on their specific fuel efficiency.

```
// Abstract class Vehicle
abstract class Vehicle {
    protected double speed; // Speed in km/h
    protected double fuelCapacity; // Fuel capacity in liters

    // Constructor
    public Vehicle(double speed, double fuelCapacity) {
        this.speed = speed;
        this.fuelCapacity = fuelCapacity;
    }

    // Abstract method to calculate range
    public abstract double calculateRange();

    // Method to display vehicle details
    public void displayDetails() {
        System.out.println("Speed: " + speed + " km/h");
        System.out.println("Fuel Capacity: " + fuelCapacity + " liters");
    }
}
```

```
// Car class extends Vehicle
class Car extends Vehicle {
    private double fuelEfficiency; // Fuel efficiency in km/l

    // Constructor
    public Car(double speed, double fuelCapacity, double fuelEfficiency) {
        super(speed, fuelCapacity);
        this.fuelEfficiency = fuelEfficiency;
    }

    // Implementing calculateRange
    @Override
    public double calculateRange() {
        return fuelCapacity * fuelEfficiency;
    }
}

// Motorcycle class extends Vehicle
class Motorcycle extends Vehicle {
    private double fuelEfficiency; // Fuel efficiency in km/l

    // Constructor
    public Motorcycle(double speed, double fuelCapacity, double fuelEfficiency) {
        super(speed, fuelCapacity);
        this.fuelEfficiency = fuelEfficiency;
    }

    // Implementing calculateRange
    @Override
    public double calculateRange() {
```

```
        return fuelCapacity * fuelEfficiency;
    }
}

// Truck class extends Vehicle
class Truck extends Vehicle {
    private double fuelEfficiency; // Fuel efficiency in km/l

    // Constructor
    public Truck(double speed, double fuelCapacity, double fuelEfficiency) {
        super(speed, fuelCapacity);
        this.fuelEfficiency = fuelEfficiency;
    }

    // Implementing calculateRange
    @Override
    public double calculateRange() {
        return fuelCapacity * fuelEfficiency;
    }
}

// Main class
public class VehicleTest {
    public static void main(String[] args) {
        // Create instances of each vehicle type
        Vehicle car = new Car(120, 50, 15); // speed, fuelCapacity, fuelEfficiency
        Vehicle motorcycle = new Motorcycle(100, 15, 40);
        Vehicle truck = new Truck(80, 150, 5);

        // Display details and range for each vehicle
        System.out.println("Car:");
```

```
        car.displayDetails();
        System.out.println("Range: " + car.calculateRange() + " km\n");

        System.out.println("Motorcycle:");
        motorcycle.displayDetails();
        System.out.println("Range: " + motorcycle.calculateRange() + " km\n");

        System.out.println("Truck:");
        truck.displayDetails();
        System.out.println("Range: " + truck.calculateRange() + " km\n");
    }
}
```

Write a Java program to create an abstract class Appliance with properties like powerConsumption and brand, and an abstractmethod calculateEnergyUsage(). Provide subclasses WashingMachine, Refrigerator, and Microwave, each implementing the calculateEnergyUsage()method to calculate energyusage based on their specific usage patterns.

```
// Abstract class Appliance
abstract class Appliance {
    protected double powerConsumption; // Power consumption in watts
    protected String brand; // Brand of the appliance

    // Constructor
    public Appliance(double powerConsumption, String brand) {
        this.powerConsumption = powerConsumption;
        this.brand = brand;
    }

    // Abstract method to calculate energy usage
    public abstract double calculateEnergyUsage(int hours);

    // Method to display appliance details
    public void displayDetails() {
```

```
        System.out.println("Brand: " + brand);
        System.out.println("Power Consumption: " + powerConsumption + " watts");
    }
}

// WashingMachine subclass
class WashingMachine extends Appliance {
    private double loadFactor; // Efficiency factor for energy calculation

    // Constructor
    public WashingMachine(double powerConsumption, String brand, double loadFactor) {
        super(powerConsumption, brand);
        this.loadFactor = loadFactor;
    }

    // Implementing calculateEnergyUsage
    @Override
    public double calculateEnergyUsage(int hours) {
        return powerConsumption * loadFactor * hours / 1000; // Energy in kWh
    }
}

// Refrigerator subclass
class Refrigerator extends Appliance {
    // Constructor
    public Refrigerator(double powerConsumption, String brand) {
        super(powerConsumption, brand);
    }

    // Implementing calculateEnergyUsage
    @Override
```

```
    public double calculateEnergyUsage(int hours) {
        return powerConsumption * hours / 1000; // Energy in kWh
    }
}

// Microwave subclass
class Microwave extends Appliance {
    private double efficiency; // Efficiency of the microwave

    // Constructor
    public Microwave(double powerConsumption, String brand, double efficiency) {
        super(powerConsumption, brand);
        this.efficiency = efficiency;
    }

    // Implementing calculateEnergyUsage
    @Override
    public double calculateEnergyUsage(int hours) {
        return powerConsumption * efficiency * hours / 1000; // Energy in kWh
    }
}

// Main class
public class ApplianceTest {
    public static void main(String[] args) {
        // Create instances of each appliance type
        Appliance washingMachine = new WashingMachine(500, "Samsung", 0.75);
        Appliance refrigerator = new Refrigerator(150, "LG");
        Appliance microwave = new Microwave(1200, "Panasonic", 0.8);

        // Display details and calculate energy usage
```

```
        System.out.println("Washing Machine:");
        washingMachine.displayDetails();
        System.out.println("Energy Usage for 3 hours: " + washingMachine.calculateEnergyUsage(3) + "
kWh\n");

        System.out.println("Refrigerator:");
        refrigerator.displayDetails();
        System.out.println("Energy Usage for 24 hours: " + refrigerator.calculateEnergyUsage(24) + "
kWh\n");

        System.out.println("Microwave:");
        microwave.displayDetails();
        System.out.println("Energy Usage for 2 hours: " + microwave.calculateEnergyUsage(2) + "
kWh\n");
    }
}
```

Writea Javaprogramtocreateanabstract classDocumentwithproperties like titleand author,andanabstractmethodprintContent().ProvidesubclassesBook,Article,andReport, eachimplementingtheprintContent()methodtodisplaythecontentspecifictoeachtypeof document.

```
// Abstract class Document
abstract class Document {
    protected String title;
    protected String author;

    // Constructor
    public Document(String title, String author) {
        this.title = title;
        this.author = author;
    }

    // Abstract method to print content
    public abstract void printContent();
```

```
    // Method to display document details
    public void displayDetails() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
    }
}

// Book subclass
class Book extends Document {
    private String genre;

    // Constructor
    public Book(String title, String author, String genre) {
        super(title, author);
        this.genre = genre;
    }

    // Implementing printContent
    @Override
    public void printContent() {
        System.out.println("This book is a " + genre + " novel with captivating storytelling.");
    }
}

// Article subclass
class Article extends Document {
    private String publication;

    // Constructor
    public Article(String title, String author, String publication) {
        super(title, author);
```

```
        this.publication = publication;
    }

    // Implementing printContent
    @Override
    public void printContent() {
        System.out.println("This article was published in " + publication + " and discusses current trends.");
    }
}

// Report subclass
class Report extends Document {
    private String summary;

    // Constructor
    public Report(String title, String author, String summary) {
        super(title, author);
        this.summary = summary;
    }

    // Implementing printContent
    @Override
    public void printContent() {
        System.out.println("This report contains an in-depth analysis: " + summary);
    }
}

// Main class
public class DocumentTest {
    public static void main(String[] args) {
```

```
        // Create instances of each document type
        Document book = new Book("The Great Adventure", "John Smith", "Adventure");
        Document article = new Article("Tech Innovations 2024", "Jane Doe", "Tech Monthly");
        Document report = new Report("Quarterly Financial Overview", "Alice Johnson", "Highlights of
Q2 performance.");

        // Display details and content for each document
        System.out.println("Book:");
        book.displayDetails();
        book.printContent();
        System.out.println();

        System.out.println("Article:");
        article.displayDetails();
        article.printContent();
        System.out.println();

        System.out.println("Report:");
        report.displayDetails();
        report.printContent();
    }
}
```

Writeaprogramtoprint thenamesofstudentsbycreatingaStudentclass. Ifnonameis passedwhilecreatinganobjectof theStudentclass, thenthenameshouldbe"Unknown", otherwisethenameshouldbeequaltotheStringvaluepassedwhilecreatinganobjectofthe Student class.

```
class Student {
    private String name;

    // Default constructor: sets name to "Unknown"
    public Student() {
        this.name = "Unknown";
```

```
    }

    // Parameterized constructor: sets name to the provided value
    public Student(String name) {
        this.name = name;
    }

    // Method to print the student's name
    public void printName() {
        System.out.println("Student Name: " + name);
    }
}

// Main class
public class StudentTest {
    public static void main(String[] args) {
        // Create Student objects
        Student student1 = new Student(); // No name provided
        Student student2 = new Student("Alice"); // Name provided

        // Print names
        student1.printName(); // Should print "Unknown"
        student2.printName(); // Should print "Alice"
    }
}
```

Design a class Time With data members for hr and min sec. Provide default and Parameterized constructors. Write a program to print date and time java.time.format. Date Time Formatter. (CONSTRUCTOR)

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

// Time class
```

```
class Time {
    private int hr;
    private int min;
    private int sec;

    // Default constructor
    public Time() {
        this.hr = 0;
        this.min = 0;
        this.sec = 0;
    }

    // Parameterized constructor
    public Time(int hr, int min, int sec) {
        this.hr = hr;
        this.min = min;
        this.sec = sec;
    }

    // Method to display time
    public void displayTime() {
        System.out.printf("Time: %02d:%02d:%02d\n", hr, min, sec);
    }
}

public class TimeTest {
    public static void main(String[] args) {
        // Creating objects of Time class
        Time defaultTime = new Time(); // Using default constructor
        Time customTime = new Time(14, 30, 45); // Using parameterized constructor
```

```
        // Displaying times
        System.out.println("Default Time:");
        defaultTime.displayTime();

        System.out.println("Custom Time:");
        customTime.displayTime();

        // Printing current date and time
        LocalDateTime now = LocalDateTime.now(); // Current date and time
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        String formattedDateTime = now.format(formatter);
        System.out.println("Current Date and Time: " + formattedDateTime);
    }
}
```

WriteaJavaprogramtocreateaclasscalledCirclewithaprivateinstancevariableradius. Providepublicgetterandsettermethodstoaccessandmodifytheradiusvariable.However, provide two methods called calculateArea() and calculatePerimeter() that return the calculated area and perimeter based on the current radius value.(INHERITANCE)

```
// Base class Shape
class Shape {
    // Common properties and methods for shapes can be added here
    public void displayShapeType() {
        System.out.println("This is a shape.");
    }
}

// Circle class extending Shape
class Circle extends Shape {
    private double radius;

    // Default constructor
    public Circle() {
```

```
    this.radius = 0.0;
}

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

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

// Setter for radius
public void setRadius(double radius) {
    if (radius < 0) {
        System.out.println("Radius cannot be negative. Setting radius to 0.");
        this.radius = 0;
    } else {
        this.radius = radius;
    }
}

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

// Method to calculate perimeter
public double calculatePerimeter() {
    return 2 * Math.PI * radius;
```

```
    }
}

// Main class to test Circle class
public class CircleTest {
    public static void main(String[] args) {
        // Create a Circle object using the default constructor
        Circle circle1 = new Circle();
        circle1.setRadius(5.0);

        System.out.println("Circle 1:");
        System.out.println("Radius: " + circle1.getRadius());
        System.out.println("Area: " + circle1.calculateArea());
        System.out.println("Perimeter: " + circle1.calculatePerimeter());
        System.out.println();

        // Create a Circle object using the parameterized constructor
        Circle circle2 = new Circle(7.0);

        System.out.println("Circle 2:");
        System.out.println("Radius: " + circle2.getRadius());
        System.out.println("Area: " + circle2.calculateArea());
        System.out.println("Perimeter: " + circle2.calculatePerimeter());
        System.out.println();
    }
}
```

WriteaJavaprogramtocreateabaseclassAnimalwithmethodsmove()andmakeSound(). Create twosubclassesBirdandPanthera.Override themove()methodineachsubclass to describehoweachanimalmoves.Also,overridethemakeSound()methodineachsubclass to make a specific sound for each animal. (POLYMORPHISM)

```
// Base class Animal
class Animal {
```

```
    // Method to describe how the animal moves
    public void move() {
        System.out.println("This animal moves in some way.");
    }

    // Method to describe the sound the animal makes
    public void makeSound() {
        System.out.println("This animal makes a sound.");
    }
}

// Subclass Bird that extends Animal
class Bird extends Animal {
    // Overriding the move method to describe how a bird moves
    @Override
    public void move() {
        System.out.println("The bird flies in the sky.");
    }

    // Overriding the makeSound method to describe the sound a bird makes
    @Override
    public void makeSound() {
        System.out.println("The bird chirps.");
    }
}

// Subclass Panthera that extends Animal
class Panthera extends Animal {
    // Overriding the move method to describe how a panthera moves
    @Override
    public void move() {
```

```
        System.out.println("The panthera runs swiftly on the ground.");
    }

    // Overriding the makeSound method to describe the sound a panthera makes
    @Override
    public void makeSound() {
        System.out.println("The panthera roars.");
    }
}

// Main class to test polymorphism
public class AnimalTest {
    public static void main(String[] args) {
        // Create objects of Bird and Panthera
        Animal myBird = new Bird(); // Using Animal reference, Bird object
        Animal myPanthera = new Panthera(); // Using Animal reference, Panthera object

        // Call move and makeSound on Bird
        System.out.println("Bird:");
        myBird.move(); // Calls the overridden move() method in Bird
        myBird.makeSound(); // Calls the overridden makeSound() method in Bird
        System.out.println();

        // Call move and makeSound on Panthera
        System.out.println("Panthera:");
        myPanthera.move(); // Calls the overridden move() method in Panthera
        myPanthera.makeSound(); // Calls the overridden makeSound() method in Panthera
    }
}
```

Write a Java program that reads a list of numbers from a file and throws an exception if any of the numbers are positive. (EXCEPTION HANDLING)

```
// Custom exception for positive numbers
class PositiveNumberException extends Exception {
    public PositiveNumberException(String message) {
        super(message);
    }
}
import java.io.*;
import java.util.Scanner;

public class NumberFileReader {

    // Method to read numbers from the file and check for positive numbers
    public static void readAndCheckNumbers(String filename) throws PositiveNumberException {
        try (Scanner scanner = new Scanner(new File(filename))) {
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();
                try {
                    int number = Integer.parseInt(line.trim());

                    // Check if the number is positive
                    if (number > 0) {
                        throw new PositiveNumberException("Positive number found: " + number);
                    }
                } catch (NumberFormatException e) {
                    System.out.println("Skipping invalid input: " + line);
                }
            }
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + filename);
        }
    }
```

```
    public static void main(String[] args) {
        // Path to the file containing the numbers
        String filename = "numbers.txt";

        try {
            // Call method to read the file and check the numbers
            readAndCheckNumbers(filename);
            System.out.println("No positive numbers found in the file.");
        } catch (PositiveNumberException e) {
            // Handle exception when a positive number is found
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Write a Java program to remove the third element from an array list.

```
import java.util.ArrayList;

public class RemoveThirdElement {
    public static void main(String[] args) {
        // Create and populate the ArrayList with some elements
        ArrayList<String> list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");
        list.add("Date");
        list.add("Elderberry");

        System.out.println("Original list: " + list);

        // Check if the list has at least 3 elements to remove the third element
```

```
        if (list.size() >= 3) {
            // Remove the third element (index 2 since indexing starts from 0)
            list.remove(2);
            System.out.println("List after removing the third element: " + list);
        } else {
            System.out.println("The list does not have enough elements to remove the third one.");
        }
    }
}
```

Write a Java program to insert the specified element at the specified position in the linked list.

```
import java.util.LinkedList;

public class InsertElementInLinkedList {
    public static void main(String[] args) {
        // Create a LinkedList of Strings
        LinkedList<String> list = new LinkedList<>();

        // Add some elements to the LinkedList
        list.add("Apple");
        list.add("Banana");
        list.add("Orange");
        list.add("Mango");

        System.out.println("Original LinkedList: " + list);

        // Specify the element to insert and the position
        String elementToInsert = "Grapes";
        int position = 2; // Insert the element at index 2 (third position)

        // Check if the position is valid
        if (position >= 0 && position <= list.size()) {
```

```
            // Insert the element at the specified position
            list.add(position, elementToInsert);
            System.out.println("LinkedList after insertion: " + list);
        } else {
            System.out.println("Invalid position! The list size is: " + list.size());
        }
    }
}
```

Write a Java program to get the number of elements in a hash set.(COLLECTION OF FRAMEWORK)

```
import java.util.HashSet;

public class HashSetSizeExample {
    public static void main(String[] args) {
        // Create a HashSet of Strings
        HashSet<String> set = new HashSet<>();

        // Add some elements to the HashSet
        set.add("Apple");
        set.add("Banana");
        set.add("Orange");
        set.add("Mango");

        // Print the HashSet
        System.out.println("HashSet: " + set);

        // Get the number of elements in the HashSet using size() method
        int size = set.size();

        // Print the size of the HashSet
        System.out.println("Number of elements in the HashSet: " + size);
    }
```

```
}
```