1. Ronak wants to develop an app for his movie theatre that manages seat reservations to help his villagers book tickets. The theatre has a limited number of seats, and customers can choose and reserve seats based on availability. The program should display a seating chart, allow seat reservations, and provide an option to quit the application once the customer is done.
   Customers should be able to choose the menu provided in the system and proceed.

2. A weather station collects temperature data every hour throughout the day. A scientist wants to analyse this temperature data to determine the hottest, and coldest hours, the average temperatures of the day and the number of hours where the temperature exceeded the threshold of 30 degrees Celsius. Analyse this scenario and implement the same using Java and implement the same using Java

3. Imagine an e-commerce platform like Amazon, where customers can shop for a wide range of items, including electronics, clothing, groceries, and cosmetics. Each of these categories has specific details that need to be captured, such as brand and warranty for electronics, size and material for clothing, weight and expiration date for groceries, and brand and type for cosmetics.
   To efficiently manage the addition of items to the shopping cart, the platform's backend system uses method overloading to handle each product category individually, ensuring that the right information is captured for each type of item.

4. Develop a class "SortingClass" that has three overloaded methods to sort data of int, float, and String types. Demonstrate its working by passing an array of all the above three types. Use any sorting technique.

5. Demonstrate the concept of class and its members by writing a Java program to define a class Lamp. It can be in an on or off-state. You can turn on and turn off the lamp. Display whether the Lamp is in on or off-state

6. Smart Home Lighting System- Imagine you are designing a smart home lighting system where each room in the house is equipped with smart lamps that can be controlled programmatically. The system allows users to turn lamps on or off and check their status via a smart home app or a central control panel.

Ronak wants to develop an app for his movie theatre that manages seat reservations to help his villagers book tickets. The theatre has a limited number of seats, and customers can choose and reserve seats based on availability. The program should display a seating chart, allow seat reservations, and provide an option to quit the application once the customer is done.

```
import java.util.Scanner;
public class MovieTheatreReservation {
    // 2D array to represent the seating chart (0 = available, 1 = reserved)
    private static int[][] seats = new int[5][5]; // 5x5 seating chart for simplicity

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice;

        // Main menu loop
        do {
            System.out.println("\nWelcome to Ronak's Movie Theatre");
            System.out.println("1. View Seats");
            System.out.println("2. Reserve a Seat");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    displaySeats();
                    break;
```

```
            case 2:
                reserveSeat(sc);
                break;
            case 3:
                System.out.println("Thank you for using the reservation system!");
                break;
            default:
                System.out.println("Invalid choice. Please try again.");
        }
    } while (choice != 3);

    sc.close();
}

// Function to display the seating chart
public static void displaySeats() {
    System.out.println("\nSeating Chart:");
    System.out.println("0 = Available, 1 = Reserved\n");
    for (int i = 0; i < seats.length; i++) {
        for (int j = 0; j < seats[i].length; j++) {
            System.out.print(seats[i][j] + " ");
        }
        System.out.println();
    }
}

// Function to reserve a seat
public static void reserveSeat(Scanner sc) {
    System.out.println("\nEnter the row (0-4): ");
    int row = sc.nextInt();
    System.out.println("Enter the column (0-4): ");
```

```
        int col = sc.nextInt();

        if (row >= 0 && row < seats.length && col >= 0 && col < seats[row].length) {
            if (seats[row][col] == 0) {
                seats[row][col] = 1; // Mark the seat as reserved
                System.out.println("Seat reserved successfully.");
            } else {
                System.out.println("Sorry, that seat is already reserved.");
            }
        } else {
            System.out.println("Invalid seat number. Please try again.");
        }
    }
}
```

2. A weather station collects temperature data every hour throughout the day. A scientist wants to analyse this temperature data to determine the hottest, and coldest hours, the average temperatures of the day and the number of hours where the temperature exceeded the threshold of 30 degrees Celsius. Analyse this scenario and implement the same using Java and implement the same using Java

```
import java.util.Scanner;

public class WeatherStation {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        double[] temperatures = new double[24]; // Array to store temperature data for 24 hours
        double totalTemperature = 0;
        double maxTemperature = Double.MIN_VALUE;
        double minTemperature = Double.MAX_VALUE;
        int hottestHour = 0;
        int coldestHour = 0;
        int hoursAboveThreshold = 0;
        final double THRESHOLD = 30.0;

        // Collect temperature data
        System.out.println("Enter the temperature data for 24 hours:");
        for (int hour = 0; hour < 24; hour++) {
            System.out.print("Hour " + hour + ": ");
            temperatures[hour] = sc.nextDouble();

            // Calculate total for average
```

```
            totalTemperature += temperatures[hour];

            // Determine hottest hour
            if (temperatures[hour] > maxTemperature) {
                maxTemperature = temperatures[hour];
                hottestHour = hour;
            }

            // Determine coldest hour
            if (temperatures[hour] < minTemperature) {
                minTemperature = temperatures[hour];
                coldestHour = hour;
            }

            // Count hours above threshold
            if (temperatures[hour] > THRESHOLD) {
                hoursAboveThreshold++;
            }
        }

        // Calculate the average temperature
        double averageTemperature = totalTemperature / 24;

        // Display results
        System.out.println("\n--- Temperature Analysis ---");
            System.out.println("Hottest Hour: " + hottestHour + " with temperature " + maxTemperature + "°C");
            System.out.println("Coldest Hour: " + coldestHour + " with temperature " + minTemperature + "°C");
        System.out.println("Average Temperature: " + averageTemperature + "°C");
          System.out.println("Number of hours where temperature exceeded " + THRESHOLD + "°C: " + hoursAboveThreshold);
```

```
        sc.close();
    }
}
```

3. Imagine an e-commerce platform like Amazon, where customers can shop for a wide range of items, including electronics, clothing, groceries, and cosmetics. Each of these categories has specific details that need to be captured, such as brand and warranty for electronics, size and material for clothing, weight and expiration date for groceries, and brand and type for cosmetics.

To efficiently manage the addition of items to the shopping cart, the platform's backend system uses method overloading to handle each product category individually, ensuring that the right information is captured for each type of item.

```
class ShoppingCart {
    // Fixed-size array to store cart items
    private String[] cartItems = new String[10];
    private int itemCount = 0;  // Keeps track of how many items are added


    // Method to add Electronics (Overloaded method)
    public void addItem(String category, String brand, int warranty) {
        if (itemCount < cartItems.length) {
            String item = "Category: " + category + ", Brand: " + brand + ", Warranty: " + warranty
+ " years";
            cartItems[itemCount] = item;
```

```
            itemCount++;
        } else {
            System.out.println("Cart is full. Cannot add more items.");
        }
    }

    // Method to add Clothing (Overloaded method)
    public void addItem(String category, String size, String material) {
        if (itemCount < cartItems.length) {
            String item = "Category: " + category + ", Size: " + size + ", Material: " + material;
            cartItems[itemCount] = item;
            itemCount++;
        } else {
            System.out.println("Cart is full. Cannot add more items.");
        }
    }

    // Method to add Groceries (Overloaded method)
    public void addItem(String category, double weight, String expirationDate) {
        if (itemCount < cartItems.length) {
            String item = "Category: " + category + ", Weight: " + weight + " kg, Expiration Date: "
+ expirationDate;
            cartItems[itemCount] = item;
            itemCount++;
        } else {
            System.out.println("Cart is full. Cannot add more items.");
        }
    }

    // Method to add Cosmetics (Overloaded method)
    public void addItem(String category, String brand, String type) {
```

```
        if (itemCount < cartItems.length) {
            String item = "Category: " + category + ", Brand: " + brand + ", Type: " + type;
            cartItems[itemCount] = item;
            itemCount++;
        } else {
            System.out.println("Cart is full. Cannot add more items.");
        }
    }

    // Method to display cart items
    public void displayCart() {
        if (itemCount == 0) {
            System.out.println("Your shopping cart is empty.");
        } else {
            System.out.println("\n--- Items in your Shopping Cart ---");
            for (int i = 0; i < itemCount; i++) {
                System.out.println(cartItems[i]);
            }
        }
    }
}

public class EcommercePlatform {
    public static void main(String[] args) {
        ShoppingCart cart = new ShoppingCart();

        // Adding various items to the cart
        cart.addItem("Electronics", "Samsung", 2);          // Electronics item
        cart.addItem("Clothing", "L", "Cotton");            // Clothing item
        cart.addItem("Groceries", 2.5, "2024-12-01");       // Groceries item
        cart.addItem("Cosmetics", "L'Oreal", "Lipstick");   // Cosmetics item
```

```
        // Display cart items
        cart.displayCart();
    }
}
```

```
public class GetCharsExample {
    public static void main(String[] args) {
        // Create a string
        String str = "Hello, World!";

        // Create a destination character array
        char[] dst = new char[5];
        str.getChars(7, 12, dst, 0);

        // Print the contents of the destination array
        System.out.println("Characters copied to destination array:");
        System.out.println(dst); // Output will be "World"
    }
}
// public void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)
```