Q.9 Design a system where multiple users (threads) can book movie tickets simultaneously. Implement multithreading to simulate the booking process, ensuring that:

Thread safety is maintained when accessing shared resources (e.g., the total number of available tickets).

Use synchronization to prevent multiple users from booking the same seat at the same time.

Ans.

```
Class MovieTicketBooking {
  Private int availableSeats;

  // Constructor to initialize available seats
  Public MovieTicketBooking(int seats) {
    This.availableSeats = seats;
  }

  // Synchronized method to book a ticket
  Public synchronized void bookTicket(String user, int numberOfTickets) {
    System.out.println(user + " is trying to book " + numberOfTickets + " ticket(s).");

    // Check if enough seats are available
    If (numberOfTickets <= availableSeats) {
      System.out.println("Booking successful for " + user + ". " + numberOfTickets + " ticket(s) booked.");
      availableSeats -= numberOfTickets;
    } else {
      System.out.println("Booking failed for " + user + ". Not enough seats available.");
    }
    System.out.println("Seats remaining: " + availableSeats);
  }
}

// Thread class representing a user trying to book a ticket
Class UserThread extends Thread {
  Private MovieTicketBooking bookingSystem;
```

```
    Private String userName;
    Private int ticketsToBook;

    // Constructor to initialize thread details
    Public UserThread(MovieTicketBooking bookingSystem, String userName, int ticketsToBook) {
        This.bookingSystem = bookingSystem;
        This.userName = userName;
        This.ticketsToBook = ticketsToBook;
    }

    // Run method of thread
    @Override
    Public void run() {
        bookingSystem.bookTicket(userName, ticketsToBook);
    }
}

// Main class to simulate the ticket booking process
Public class MovieTicketBookingSystem {
    Public static void main(String[] args) {
        // Initialize the booking system with 10 available seats
        MovieTicketBooking bookingSystem = new MovieTicketBooking(10);

        // Create user threads trying to book tickets
        UserThread user1 = new UserThread(bookingSystem, “User1”, 4);
        UserThread user2 = new UserThread(bookingSystem, “User2”, 5);
        UserThread user3 = new UserThread(bookingSystem, “User3”, 3);

        // Start user threads
        User1.start();
```

```
        User2.start();
        User3.start();
    }
}
```

OUTPUT:

```
User1 is trying to book 4 ticket(s).
Booking successful for User1. 4 ticket(s) booked.
Seats remaining: 6
User2 is trying to book 5 ticket(s).
Booking successful for User2. 5 ticket(s) booked.
Seats remaining: 1
User3 is trying to book 3 ticket(s).
Booking failed for User3. Not enough seats available.
Seats remaining: 1
```

Create a library management system using a List to store book objects where:

- **Each book has attributes like title, author, and genre.**
- **Implement functions to add new books, search for books by author or genre, and display all available books.**
- **Use an ArrayList to maintain the collection of books and implement sorting functionality to display books in alphabetical order.**

Ans.

```
Import java.util.ArrayList;
Import java.util.Scanner;
// Book class to store attributes of each book
Class Book {
    String title;
    String author;
    String genre;
    // Constructor
    Public Book(String title, String author, String genre) {
```

```
    This.title = title;
    This.author = author;
    This.genre = genre;
  }
  // Method to display book details
  @Override
  Public String toString() {
    Return “Title: “ + title + “, Author: “ + author + “, Genre: “ + genre;
  }
}

// Library class to manage the list of books
Class Library {
  ArrayList<Book> books = new ArrayList<>();

  // Method to add a new book to the library
  Public void addBook(String title, String author, String genre) {
    Book newBook = new Book(title, author, genre);
    books.add(newBook);
    System.out.println(“Book added successfully!”);
  }

  // Method to search for books by author
  Public void searchByAuthor(String author) {
    Boolean found = false;
    System.out.println(“\nBooks by author ‘” + author + “’:”);
    For (Book book : books) {
      If (book.author.equalsIgnoreCase(author)) {
        System.out.println(book);
        Found = true;
```

```
      }
    }
    If (!found) {
      System.out.println(“No books found by this author.”);
    }
  }

// Method to search for books by genre
Public void searchByGenre(String genre) {
    Boolean found = false;
    System.out.println(“\nBooks in genre ‘” + genre + “’:”);
    For (Book book : books) {
      If (book.genre.equalsIgnoreCase(genre)) {
        System.out.println(book);
        Found = true;
      }
    }
    If (!found) {
      System.out.println(“No books found in this genre.”);
    }
}

// Method to display all books sorted alphabetically by title using Bubble Sort
Public void displayAllBooks() {
    If (books.isEmpty()) {
      System.out.println(“No books available in the library.”);
      Return;
    }

    // Bubble Sort to sort books by title
```

```
    For (int I = 0; I < books.size() – 1; i++) {
      For (int j = 0; j < books.size() – I – 1; j++) {
        If (books.get(j).title.compareToIgnoreCase(books.get(j + 1).title) > 0) {
          // Swap books[j] and books[j + 1]
          Book temp = books.get(j);
          Books.set(j, books.get(j + 1));
          Books.set(j + 1, temp);
        }
      }
    }

    System.out.println("\nList of all available books (sorted by title):");
    For (Book book : books) {
      System.out.println(book);
    }
  }
}

// Main class
Public class LibraryManagementSystem {
  Public static void main(String[] args) {
    Library library = new Library();
    Scanner scanner = new Scanner(System.in);

    While (true) {
      System.out.println("\nLibrary Management System");
      System.out.println("1. Add New Book");
      System.out.println("2. Search Book by Author");
      System.out.println("3. Search Book by Genre");
      System.out.println("4. Display All Books");
```

```
System.out.println("5. Exit");
System.out.print("Enter your choice: ");

Int choice = scanner.nextInt();
Scanner.nextLine(); // Consume newline

Switch (choice) {
  Case 1:
    System.out.print("Enter book title: ");
    String title = scanner.nextLine();
    System.out.print("Enter book author: ");
    String author = scanner.nextLine();
    System.out.print("Enter book genre: ");
    String genre = scanner.nextLine();
    Library.addBook(title, author, genre);
    Break;

  Case 2:
    System.out.print("Enter author name to search: ");
    String searchAuthor = scanner.nextLine();
    Library.searchByAuthor(searchAuthor);
    Break;

  Case 3:
    System.out.print("Enter genre to search: ");
    String searchGenre = scanner.nextLine();
    Library.searchByGenre(searchGenre);
    Break;

  Case 4:
```

```
            Library.displayAllBooks();
            Break;

        Case 5:
            System.out.println("Exiting the system. Goodbye!");
            Scanner.close();
            Return;

        Default:
            System.out.println("Invalid choice. Please try again.");
      }
    }
  }
}
```

OUTPUT:

Library Management System

1. Add New Book

2. Search Book by Author

3. Search Book by Genre

4. Display All Books

5. Exit

Enter your choice: 1

Enter book title: JAVA

Enter book author: BHOOMI MAM

Enter book genre: PROGRAMMING

Book added successfully!

Library Management System

1. Add New Book

2. Search Book by Author

3. Search Book by Genre

4. Display All Books

5. Exit

Enter your choice: 4

List of all available books (sorted by title):

Title: JAVA, Author: BHOOMI MAM, Genre: PROGRAMMING

Que.11 Develop a system to manage student grades using a Map where:

- **The student name (or ID) is the key, and their grade is the value.**
- **Implement features to add, remove, update, and retrieve student grades.**
- **Use the HashMap collection to store student data and implement operations like searching for students with the highest grade, average grade, etc.**

Ans.

```
Import java.util.HashMap;

Import java.util.Map;

Import java.util.Scanner;


Public class StudentGradesSystem {
  // HashMap to store student name (or ID) as key and grade as value
  Private HashMap<String, Double> studentGrades;


  // Constructor
  Public StudentGradesSystem() {
    studentGrades = new HashMap<>();
  }


  // Method to add or update a student's grade
  Public void addOrUpdateStudent(String name, double grade) {
```

```
    studentGrades.put(name, grade);
    System.out.println("Student grade added/updated successfully!");
}

// Method to remove a student
Public void removeStudent(String name) {
    If (studentGrades.containsKey(name)) {
        studentGrades.remove(name);
        System.out.println("Student removed successfully!");
    } else {
        System.out.println("Student not found.");
    }
}

// Method to retrieve a student's grade
Public void getStudentGrade(String name) {
    If (studentGrades.containsKey(name)) {
        System.out.println("Grade of " + name + ": " + studentGrades.get(name));
    } else {
        System.out.println("Student not found.");
    }
}

// Method to find the student(s) with the highest grade
Public void getHighestGrade() {
    If (studentGrades.isEmpty()) {
        System.out.println("No students in the system.");
        Return;
    }
```

```
    Double highestGrade = Double.MIN_VALUE;
    For (double grade : studentGrades.values()) {
      If (grade > highestGrade) {
        highestGrade = grade;
      }
    }

    System.out.println("Students with the highest grade (" + highestGrade + "):");
    For (Map.Entry<String, Double> entry : studentGrades.entrySet()) {
      If (entry.getValue() == highestGrade) {
        System.out.println(entry.getKey());
      }
    }
}

// Method to calculate the average grade
Public void getAverageGrade() {
    If (studentGrades.isEmpty()) {
      System.out.println("No students in the system.");
      Return;
    }

    Double sum = 0;
    For (double grade : studentGrades.values()) {
      Sum += grade;
    }

    Double averageGrade = sum / studentGrades.size();
    System.out.println("Average grade of all students: " + averageGrade);
}
```

```
// Method to display all students and their grades
Public void displayAllStudents() {
  If (studentGrades.isEmpty()) {
    System.out.println("No students in the system.");
    Return;
  }

  System.out.println("List of all students and their grades:");
  For (Map.Entry<String, Double> entry : studentGrades.entrySet()) {
    System.out.println("Student: " + entry.getKey() + ", Grade: " + entry.getValue());
  }
}

// Main method
Public static void main(String[] args) {
  StudentGradesSystem gradesSystem = new StudentGradesSystem();
  Scanner scanner = new Scanner(System.in);
  System.out.println("\nStudent Grades Management System");
  System.out.println("1. Add/Update Student Grade");
  System.out.println("2. Remove Student");
  System.out.println("3. Retrieve Student Grade");
  System.out.println("4. Display All Students");
  System.out.println("5. Find Student(s) with Highest Grade");
  System.out.println("6. Calculate Average Grade");
  System.out.println("7. Exit");

  While (true) {

    System.out.print("Enter your choice: ");
```

```
Int choice = scanner.nextInt();
Scanner.nextLine(); // Consume newline

Switch (choice) {
  Case 1:
    System.out.print("Enter student name (or ID): ");
    String name = scanner.nextLine();
    System.out.print("Enter student grade: ");
    Double grade = scanner.nextDouble();
    gradesSystem.addOrUpdateStudent(name, grade);
    break;

  case 2:
    System.out.print("Enter student name (or ID) to remove: ");
    String removeName = scanner.nextLine();
    gradesSystem.removeStudent(removeName);
    break;

  case 3:
    System.out.print("Enter student name (or ID) to retrieve grade: ");
    String retrieveName = scanner.nextLine();
    gradesSystem.getStudentGrade(retrieveName);
    break;

  case 4:
    gradesSystem.displayAllStudents();
    break;

  case 5:
```

```
            gradesSystem.getHighestGrade();
            break;

        case 6:
            gradesSystem.getAverageGrade();
            break;

        case 7:
            System.out.println("Exiting the system. Goodbye!");
            Scanner.close();
            Return;

        Default:
            System.out.println("Invalid choice. Please try again.");
      }
    }
  }
}
```

OUTPUT:

```
Student Grades Management System
1. Add/Update Student Grade
2. Remove Student
3. Retrieve Student Grade
4. Display All Students
5. Find Student(s) with Highest Grade
6. Calculate Average Grade
7. Exit
Enter your choice: 1
```

Enter student name (or ID): Student1

Enter student grade: 85.5

Student grade added/updated successfully!

Enter your choice: 1

Enter student name (or ID): Student2

Enter student grade: 92.0

Student grade added/updated successfully!

Enter your choice: 5

Students with the highest grade (92.0):

Student2

Enter your choice: 6

Average grade of all students: 88.75