DAY-3 TASK

1)Implements runnable to print a message from multiple threads

```
class MyTask implements Runnable {
    private String message;
     MyTask(String message) {
        this.message = message;
    }
      public void run() {
        System.out.println("Thread Message: " + message);
    }
}

public class MultiThreadDemo {
    public static void main(String[] args) {
        MyTask task1 = new MyTask("Hello from Thread 1");
        MyTask task2 = new MyTask("Hello from Thread 2");
        MyTask task3 = new MyTask("Hello from Thread 3");
        Thread t1 = new Thread(task1);
        Thread t2 = new Thread(task2);
        Thread t3 = new Thread(task3);
        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output:

```
Thread Message: Hello from Thread 1
Thread Message: Hello from Thread 2
Thread Message: Hello from Thread 3
```

2) Demonstrate sleep() and join() using thread coordination

```
class MyThread extends Thread {
    public void run() {
        try {
            for (int i = 1; i <= 3; i++) {
                System.out.println(getName() + " is running... Count: " + i);
                Thread.sleep(1000); // Pauses for 1 second
```

```
            }
        } catch (InterruptedException e) {
            System.out.println(getName() + " interrupted.");
        }
    }
}

public class SleepJoinDemo {
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();

        t1.setName("Thread-1");
        t2.setName("Thread-2");

        t1.start();

        try {
            t1.join(); // Wait for t1 to finish
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        t2.start();

        try {
            t2.join(); // Wait for t2 to finish
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }

        System.out.println("Main thread finished.");
    }
}
```

Output:

```
Thread-1 is running... Count: 1
Thread-1 is running... Count: 2
Thread-1 is running... Count: 3
Thread-2 is running... Count: 1
Thread-2 is running... Count: 2
Thread-2 is running... Count: 3
Main thread finished.
```

3)MAIN PROGRAM:

- Multipe users(threads) are trying to book tickets
- use a synchronized method/block to avoid race conditions
- include sleep() to simulate delays
- display booking status and remaining seats after each booking

```
class TicketBooking {
    private int availableSeats = 5;

    public void bookTicket(String user, int seatsToBook) {
        synchronized (this) {
            System.out.println(user + " is trying to book " + seatsToBook + " seat(s).");

            if (seatsToBook <= availableSeats) {
                try {

                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    System.out.println("Thread interrupted.");
                }

                availableSeats -= seatsToBook;
                System.out.println(user + " successfully booked " + seatsToBook + " seat(s).");
                System.out.println("Remaining seats: " + availableSeats);
            } else {
                System.out.println("Sorry " + user + ", not enough seats available.");
                System.out.println("Remaining seats: " + availableSeats);
            }

            System.out.println("-------------------------------");
        }
    }
}

class User extends Thread {
    private TicketBooking bookingSystem;
    private int seatsToBook;

    public User(TicketBooking bookingSystem, String name, int seatsToBook) {
        super(name);
        this.bookingSystem = bookingSystem;
        this.seatsToBook = seatsToBook;
```

```
    }

    public void run() {
        bookingSystem.bookTicket(getName(), seatsToBook);
    }
}

public class TicketBookingApp {
    public static void main(String[] args) {
        TicketBooking bookingSystem = new TicketBooking();

        User user1 = new User(bookingSystem, "User-1", 2);
        User user2 = new User(bookingSystem, "User-2", 1);
        User user3 = new User(bookingSystem, "User-3", 3);
        User user4 = new User(bookingSystem, "User-4", 1);

        user1.start();
        user2.start();
        user3.start();
        user4.start();
    }
}
```

Output:

```
User-1 is trying to book 2 seat(s).
User-2 is trying to book 1 seat(s).
User-3 is trying to book 3 seat(s).
User-4 is trying to book 1 seat(s).
User-1 successfully booked 2 seat(s).
Remaining seats: 3
--------------------------------
User-2 successfully booked 1 seat(s).
Remaining seats: 2
--------------------------------
User-3 successfully booked 3 seat(s).
Remaining seats: -1
--------------------------------
User-4 successfully booked 1 seat(s).
Remaining seats: -2
--------------------------------
```