ONDOKUZ MAYIS ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

İŞLETİM SİSTEMLERİ
2023-2024 BAHAR DÖNEMİ
Proje#2 Havayolu Rezervasyon Sistemi

Mustafa ÇETİN
20060331

Kodların Açıklamaları:

1)
```
1 import java.util.concurrent.locks.ReentrantReadWriteLock;
```
Java'nın sağladığı read-write kilidini içe aktarır.

2)
```
2 import java.time.LocalTime;
```
Zaman bilgisi almak için LocalTime sınıfını içe aktarır.

3)

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

Zaman bilgisini belirli bir formatta yazdırmak için DateTimeFormatter sınıfını içe aktarır.

4)

```
5  class ReservationSystem {
6      private final ReentrantReadWriteLock lock = new ReentrantReadWriteLock();
7      private int seatsAvailable;
8
```

Okuma ve yazma kilidi oluşturur.

Mevcut koltuk sayısını tutmak için bir değişken tanımlar.

5)

```
9      public ReservationSystem(int seats) {
10         this.seatsAvailable = seats;
11     }
```

Koltuk sayısını kurucu metodla alır ve değişkene atar.

6)

```
13     private String getCurrentTime() {
14         return LocalTime.now().format(DateTimeFormatter.ofPattern("HH:mm:ss.SSS"));
15     }
```

Geçerli zamanı belirli bir formatta döndürür.

7)

```
18     public void makeReservation() {
19         lock.writeLock().lock();
20         try {
21             if (seatsAvailable > 0) {
22                 seatsAvailable--;
23                 System.out.println("Time: " + getCurrentTime() + " " + Thread.currentThread().getName() + " tries to book the seat");
24                 System.out.println("Time: " + getCurrentTime() + " " + Thread.currentThread().getName() + " booked seat number 1 successfully.");
25             } else {
26                 System.out.println("Time: " + getCurrentTime() + " " + Thread.currentThread().getName() + " tried to book the seat, but no seats are available.");
27             }
28         } finally {
29             lock.writeLock().unlock();
30         }
31     }
```

Rezervasyon yapmak için yazma kilidi kullanan metottur.
Yazma kilidini kilitler.
Eğer mevcut koltuk varsa ;
Mevcut koltuk sayısını bir azaltır.
Rezervasyon denemesini yazdırır.
Başarılı rezervasyon mesajını yazdırır.
Mevcut koltuk yoksa;
Koltuk bulunamadı mesajını yazdırır.
Yazma kilidini serbest bırakır.

8)

```
34    public void cancelReservation() {
35        lock.writeLock().lock();
36        try {
37            seatsAvailable++;
38            System.out.println("Time: " + getCurrentTime() + " " + Thread.currentThread().getName() + " cancelled a reservation. Seats left: " + seatsAvailable);
39        } finally {
40            lock.writeLock().unlock();
41        }
42    }
```

Rezervasyon iptal etmek için yazma kilidi kullanan metottur.
Yazma kilidini kilitler.
Mevcut koltuk sayısını bir arttırır.
Rezervasyon iptali mesajını yazdırır.
Yazma kilidini serbest bırakır.

9)

```
45    public void queryReservation() {
46        lock.readLock().lock();
47        try {
48            System.out.println("Time: " + getCurrentTime() + " " + Thread.currentThread().getName() + " looks for available seats. Seats available: " + seatsAvailable);
49        } finally {
50            lock.readLock().unlock();
51        }
52    }
53 }
```

Mevcut koltukları sorgulamak için okuma kilidi kullanan metottur.
Okuma kilidini kilitler.
Mevcut koltuk sayısını yazdırır.
Okuma kilidini serbest bırakır.

10)

```
56 class WriterThread implements Runnable {
57     private final ReservationSystem reservationSystem;
58     private final String operation;
59
60     public WriterThread(ReservationSystem reservationSystem, String operation) {
61         this.reservationSystem = reservationSystem;
62         this.operation = operation;
63     }
```

Rezervasyon yapma ve iptal etme işlemleri için yazma thread sınıfıdır.
Rezervasyon sistemi referansı
Yapılacak işlem (rezervasyon veya iptal)
Rezervasyon sistemi referansını atıyoruz.
Yapılacak işlemi atıyoruz.

11)

```
65      @Override
66      public void run() {
67          if (operation.equals("reserve")) {
68              reservationSystem.makeReservation();
69          } else if (operation.equals("cancel")) {
70              reservationSystem.cancelReservation();
71          }
72      }
73  }
```

Eğer işlem rezervasyon ise;
Rezervasyon yapma metodunu çağırır.
İşlem iptal ise;
Rezervasyon iptal etme metodunu çağırır..

12)

```
76  class ReaderThread implements Runnable {
77      private final ReservationSystem reservationSystem;
78
79      public ReaderThread(ReservationSystem reservationSystem) {
80          this.reservationSystem = reservationSystem;
81      }
```

Mevcut koltukları sorgulamak için okuma thread sınıfıdır.
Rezervasyon sistemi referansı
Rezervasyon sistemi referansını atıyoruz.

13)

```
83      @Override
84      public void run() {
85          reservationSystem.queryReservation();
86      }
87  }
```

Koltukları sorgulama metodunu çağırır.

14)

```
89  public class Main {
90      public static void main(String[] args) {
91          ReservationSystem reservationSystem = new ReservationSystem(10);
92
93          Thread writer1 = new Thread(new WriterThread(reservationSystem, "reserve"), "Writer1");
94          Thread writer2 = new Thread(new WriterThread(reservationSystem, "reserve"), "Writer2");
95          Thread writer3 = new Thread(new WriterThread(reservationSystem, "reserve"), "Writer3");
96
97          Thread reader1 = new Thread(new ReaderThread(reservationSystem), "Reader1");
98          Thread reader2 = new Thread(new ReaderThread(reservationSystem), "Reader2");
99          Thread reader3 = new Thread(new ReaderThread(reservationSystem), "Reader3");
100
101         writer1.start();
102         writer2.start();
103         writer3.start();
104
105         reader1.start();
106         reader2.start();
107         reader3.start();
```

10 koltuklu bir rezervasyon sistemi oluştururuz.
Yazma thread'lerini oluşturur.
Okuma thread'lerini oluşturur.
Thread'leri başlatır.

15)

```
109         try {
110             writer1.join();
111             writer2.join();
112             writer3.join();
113             reader1.join();
114             reader2.join();
115             reader3.join();
116         } catch (InterruptedException e) {
117             e.printStackTrace();
118         }
119     }
120 }
```

Thread'lerin bitmesini bekler.
Bir hata durumunda hata yığınını yazdırır.

Kodumuzun çıktısı :

```
Time: 18:05:45.122 Writer2 booked seat number 1 successfully.
Time: 18:05:45.123 Writer3 tries to book the seat
Time: 18:05:45.123 Writer3 booked seat number 1 successfully.
Time: 18:05:45.126 Reader1 looks for available seats. Seats available: 8
Time: 18:05:45.127 Reader2 looks for available seats. Seats available: 8
Time: 18:05:45.523 Writer1 tries to book the seat
Time: 18:05:45.524 Writer1 booked seat number 1 successfully.
Time: 18:05:45.524 Reader3 looks for available seats. Seats available: 7
```