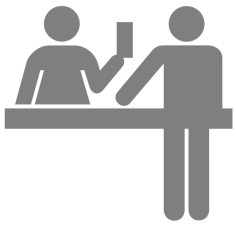
Operating Systems 2 Lab

Jean Botev (jean.botev@uni.lu)

Session VI — Concurrency & Synchronisation

Ticket Sales



Imagine a simple situation where tickets for the cinema, a concert, or similar events can only be bought before the event at the evening box-office. The visitors line up to buy a ticket at a ticket booth in order to be admitted into the venue.

As an employee of an event agency you are in charge of writing the back-end for a new digital ticketing system. The back-end design should distinguish between customers, ticket counters and tickets, for instance, as shown in the figure below.

Class TicketCounter
+ticketStock: [Ticket]
+sellTickets(requestedTickets: Int, ticketStock: &[Ticket]): [Ticket]

Struct Ticket
+nr: Int
+sold: Boolean
+used: Boolean

Class Customer
+id: Int
+boughtTickets: [Ticket]

For each event, you have a shared stock of tickets that is concurrently accessed by all counters. Each customer can request a maximum number of tickets. The seller behind the counter checks the availability of the tickets, marks them as sold, prints and hands them over to the customer. To enter the event location, the customers scan their ticket at the doors. Each ticket allows customers access to the location only once.

Write a simple, console-output Swift program that simulates the above by making use of the toolset you have been introduced to in the lecture and previous lab. In your solution design, consider the following two scenarios:

Scenario 1: Open one ticket counter to sell tickets. The customers queue at the counter and are served one by one. For each requested ticket, the counter accesses the ticket stock, marks the ticket as sold and returns it to the customer. The customer stores the received tickets.

Scenario 2: Open three more counters to sell tickets. After the customers receive their tickets, they want to enter the event location and scan their tickets at the doors. Although customers just bought their tickets, the scanner denies access to the location because some tickets have been previously used.

As your customers are honest people, you have to check your ticketing system:

- In which situation can a ticket be invalid at the doors?
- Which lines of code in your implementation can force this situation?

Find a solution and fix your implementation to make sure the access issue does not occur. You will likely find more than a single, possible solution - which option would be the best concerning the reusability of your classes and code?

Submission Information

The submission deadline for this lab is Tuesday, November 23, 2021.

Submit all the files related to your solution on Moodle in a single ZIP archive using the activity corresponding to the lab session.

For your explainer video, use Microsoft's Flipgrid (https://www.flipgrid.com/). Your usual University of Luxembourg account credentials allow you to access the site, and you will find a Join Link to the respective Flipgrid topic for your explainer video on Moodle.