Cinema by Florian EPAIN

Tutorial

To run the project, run the function `main` in the `Cinema.java` file with "animation" as argument.

Overview

I decided to concept this solution using the java monitor. `Customer` and `SuperWorker` are `Thread`s. `Room` is the shared object.

Objects

Cinema

I've paved the way for several cinema's rooms, but in the test setup (`main` function) only one is used.

Customer

First I wanted to represent the customers path like that

```
@Override
public void run() {
    while (this.movieSeen) {
        /* ----------------- Waiting the room to open ----------------- */
        while (room.getRoomState() != RoomState.OPEN) {
        }

        if (this.potentialSeat.isEmpty() && room.getRoomState() ==
RoomState.OPEN)
            this.potentialSeat = this.room.stand(this);
        /* ----------------- Waiting the flim to start ----------------- */
        while (room.getRoomState() != RoomState.PROJECTING) {
        }

        if (room.getRoomState() == RoomState.PROJECTING) {
            // Appreciate the flim
        }

        /* -------------- Waiting the flim to finish Sadge ------------- */
        while (room.getRoomState() != RoomState.EXITING) {
        }

        if (this.potentialSeat.isPresent() && room.getRoomState() ==
RoomState.EXITING) {
            this.room.freeSeat(potentialSeat.get());
            this.potentialSeat = Optional.empty();
            this.movieSeen = true;
        }
```

```
    }
} // in `Customer.java`
```

But the customers now are stubborn and will try endlessly to take a seat if they have a ticket, and will try to leave instantly. But they will be stopped by a room's state check. They will be woken up by the super-worker in the changing room's state by a `notifyAll()`. We ensure that their wake-up is valid with a while loop.

Super-Worker

The super-worker is in an infinite loop (`while (true)`) and will be interrupted when all other threads (except the main thread) are stopped.

I didn't represent the super-worker entering the room, but we can imagine it.

Room's life

I thought like giving life to the room, which will switch state itself but for now we give fullpower to the `SuperWorker`.

I put `nextRoomState()` on `synchronized` to ensure that everyone has the good room's state.

Problems

- The super-worker won't clean the room, even if everyone has left, in this configuration:

```
@Override
public void run() {
    // ...

    /* ---------------------- Exiting Phase ---------------------- */
    this.rooms[0].nextRoomState();

    while (!this.rooms[0].isRoomEmpty()) {
        // System.out.println("still waiting...");
    }

    /* ---------------------- Cleaning Phase ---------------------- */
    this.rooms[0].nextRoomState();

    // ...
} // In `SuperWorker.java`
```

  Surprisingly, when I uncomment the debug message in the while loop, the blocking disappears.

  I solved this problem by creating a room method called `clean()`, the `SuperWorker` will be put to sleep and should be woken by the departure of the last client. We can limit the complexity by a simple combination `if` / `notify`, as the only one waiting will be the super-worker.

```

```

```
public synchronized void freeSeat(Customer customer, Pair<Integer,
Integer> seat) {
    // ...

    // if last make sure to wake the super worker :)
    if (this.isRoomEmpty())
        notify();

    // ...
}

/* ------------------------ Super Employee Methods ------------------
------- */

public synchronized void clean(SuperWorker superWorker) {
    // as all the customers left, the only one waiting is the super-
worker
    if (!this.isRoomEmpty()) {
        try {
            wait();
        } catch (InterruptedException e) {
            Logger.getGlobal()
                    .warning("Super Worker got interruped in their
sleep (waiting for cleaing).\n" + e.toString());
            superWorker.interrupt();
        }
    }
}// In `Room.java`
```

All the other problems were conception or bad implementation problems.

Notes

We also checked that the code was free of deadlocks after removing the simulation sleep (when the clients moved around the cinema).