Cinema

Tutorial

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

Overview

`Customer` and `SuperWorker` are `Threads`. `Room` is the shared object.

Objects

Customer

First I wanted to represent the customers path like that

```java
@Override
public void run() {
    while (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;
        }
    }
}
```

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.

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 everyon has left in this configuration:

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

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

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

Surprisingly, when I uncomment the debug message in the while, 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 waiting by a simple combination `if` / `notify`

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