Übung11: Threads (freiwillig)

Abgabetermin: 14.06.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe28	Java Programm, Testausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe28: Simulation eines Wahllokales (12 Punkte)

Implementieren Sie, analog zum gezeigten Musterbeispiel in der Übungsstunde, ein Java Programm, in dem der Betrieb eines Wahllokals simuliert wird.

Ein Wahllokal besteht aus drei Bereichen. Jeder Bereich kann von Wählern nach gewissen Kapazitäten betreten und verlassen werden, wobei Wähler von eigenen Threads gesteuert werden. Somit stellen die Bereiche gemeinsame Ressourcen dar und der Zugriff durch die Wähler muss durch gegebene Kapazitäten wie Anzahl an Wahlhelfern und Wahlkabinen synchronisiert werden.

Ablauf: Zu Beginn trifft jeder Wähler im Eingangsbereich ein. Dort muss er eventuell warten, bis im Hauptbereich Kapazitäten der Wahlhelfer frei werden, um an die Reihe zu kommen. Im Hauptbereich angekommen vergeht eine gewisse Zeit, da die Helfer für den Wähler Unterlagen, sowie Stimmzettel und Kuvert vorbereiten müssen. Nach dem Vorbereiten kann es auch hier eventuell zu Verzögerungen kommen und der Wähler muss warten, wenn keine der gegebenen Wahlkabinen frei ist. Wird eine Wahlkabine frei, betritt der Wähler diese, füllt seinen Stimmzettel aus, und gibt das Kuvert in der Urne ab, was wieder eine gewisse Zeit lang dauern kann.

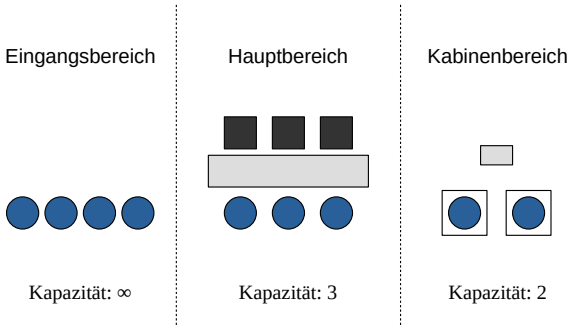

Abbildung 1: Simulation eines Wahllokals mit Beispielkapazitäten

Implementieren Sie eine Klasse `Area`, in der Sie Bereichsnamen, maximale Kapazität, und aktuelle Anzahl an Wählern speichern. Implementieren Sie weiters Methoden zum Betreten und Verlassen eines Bereichs, in denen Sie den Zugriff auf den Bereich mittels `wait()` und `notify()` synchronisieren und den Zähler beim Aufrufen der Methoden entsprechend setzen.

Implementieren Sie die Klasse `Voter` als Ableitungen von `Thread` oder `Runnable`, welche eine ID für den Wähler und die drei `Area` Objekte übergeben bekommt. Der Thread soll einmalig durchlaufen und der Reihe nach die Bereiche betreten und wieder verlassen. Achten Sie bei der Implementierung darauf, dass die Threads von außen richtig mittels `Thread.interrupt` zur Beendung aufgefordert werden können. Für die Dauer der Vorgänge im Haupt- und Kabinenbereich kann `Thread.sleep` und `Math.random()`, skaliert mit einer gewissen Größe, verwendet werden.

Achtung: Stellen Sie sicher, dass Wähler beim Wechseln zweier Bereiche nicht in einem inkonsistenten Zustand landen, wenn z.B. der Zielbereich voll ist, und implementieren Sie den Übergang zwischen zwei Bereichen als „atomare“ Operation.

Implementieren Sie zum Schluss eine Klasse `PollSite`, in der die Bereiche und die Threads für die Wähler erzeugt, gestartet und nach einer gewissen Simulationsdauer wieder sauber beendet werden. Die aktuelle Zeit kann mit `System.currentTimeMillis()` regelmäßig abfragt werden um zu prüfen, ob die Simulationsdauer erreicht worden ist. Wenn nein, kann mit `Thread.yield()` eine kurze Zeit lang die Ausführung des Hauptthreads abgeben werden.

Nachdem die Simulationsdauer erreicht worden ist, können mit `Thread.interrupt()` alle vorher gestarteten Threads zur Beendigung auffordern werden, sofern dies richtig in der Klasse `Voter` implementiert wurde. Mit `Thread.join()` kann dann auf das Ende der Threads gewartet werden.

Parametrisieren Sie die Kapazitäten der Bereiche und die Anzahl der Wähler, sowie die Simulationsdauer, damit beim Programmstart verschiedenene Werte zum Ausprobieren übergeben werden können.

Beispielausgabe:

```
# java PollSite
usage: java PollSite nOfHelpers nOfCabines nOfVoters durationMillis

# java PollSite 2 2 4 10000
Entry Hall entered by Voter 0 (voters: 1)
Voter 0 moved from Entry Hall to Main Hall (Entry Hall: 0, Main Hall: 1)
Entry Hall entered by Voter 3 (voters: 1)
Voter 3 moved from Entry Hall to Main Hall (Entry Hall: 0, Main Hall: 2)
Entry Hall entered by Voter 2 (voters: 1)
Main Hall blocked, Voter 2 has to wait (voters: 2)
Entry Hall entered by Voter 1 (voters: 2)
Main Hall blocked, Voter 1 has to wait (voters: 2)
Voter 0 moved from Main Hall to Cabines (Main Hall: 1, Cabines: 1)
Voter 2 moved from Entry Hall to Main Hall (Entry Hall: 1, Main Hall: 2)
Voter 3 moved from Main Hall to Cabines (Main Hall: 1, Cabines: 2)
Voter 1 moved from Entry Hall to Main Hall (Entry Hall: 0, Main Hall: 2)
Cabines blocked, Voter 1 has to wait (voters: 2)
Cabines left by Voter 3 (voters: 1)
Voter 1 moved from Main Hall to Cabines (Main Hall: 1, Cabines: 2)
Cabines blocked, Voter 2 has to wait (voters: 2)
Cabines left by Voter 1 (voters: 1)
Voter 2 moved from Main Hall to Cabines (Main Hall: 0, Cabines: 2)
Cabines left by Voter 0 (voters: 1)
Cabines left by Voter 2 (voters: 0)
```

Abzugeben:

- Java Programm
- Testausgaben