DOCUMENTAȚIE

TEMA 2

NUME STUDENT: Iosif Adela

GRUPA: 30223

CUPRINS

1. Obiectivul temei

Se dorește proiectarea și implementarea unui simulator pentru eficientizarea cozilor folosing thread-uri și tipuri de date care asigura securitatea acestora. La intrare se primește numărul de task-uri (clienți), numărul de cozi, timpul maxim de simulare, intervalul de timp când un client trebuie să ajungă la coadă, cât și intervalul de timp în care a stat la coadă.

Rezultatele se vor afișa în fișierul text creat din IntelliJ (logFile.txt), pentru că nu am reușit să fac o interfață grafică.

2. Analiza problemei, modelare

Datele de intrare sunt hardcodate în clasa SimulationManager cu valorile din cerință, deci nu va fi cazul de mecanisme pentru tratarea erorilor (dacă ar fi fost citite din interfață, ar fi fost utile astfel de mecanisme). Evoluția cozilor este afișată în timp real în fișierul „logFile.txt”.

3. Proiectare

Am ales să utilizez 2 pachete pentru o mai bună organizare a codului:

- **businessLogic**: conține clasele Scheduler – aici se construiesc cozile și se creează thread-urile pentru fiecare; SimulationManager – de aici se pornește simularea, se face generarea aleatoare a task-urilor; TimeStrategy – strategia care adaugă clientul cozii cu cel mai mic timp de așteptare; interfața Strategy – conține metoda addTask neimplementată (ar fi fost utilă dacă aș fi implementat mai multe moduri de selecție a cozii)
- **model**: conține clasele Server – reprezentarea cozilor (BlockingQueue) și Task – reprezentarea unui client, operații de bază pe obiect (constructor, setter, getter, compareTo, toString)

4. Implementare

Proiectul actual folosește mai multe fire de execuție, mai exact, câte un fir de execuție pentru fiecare coadă. Acestea se folosesc pentru paralelism, mai exact pentru că ne dorim partajarea unor resurse comune. De aceea, cu ajutorul thread-urilor se reușește eficientizarea resurselor hardware (aplicațiile pot rula mai rapid și eficient). Totuși, între dezavantajele thread-urilor se numără: vulnerabilitatea la erori (pot avea un comportament neașteptat) sau complexitatea (gestionarea și implementarea lor este mai complicată și necesită atenție la problemele legate de concurență și sincronizare).

Astfel, am ales în proiectul meu 2 mecanisme de sincronizare: variabile atomice (AtomicInteger) și BlockingQueue pentru a asigura securitatea firelor de execuție. **AtomicInteger** oferă operații care se execută într-un singur pas, fără a fi necesar ca mai multe fire de execuție să comunice între ele. Această clasă oferă operații atomice de incrementare și decrementare, care în mod obișnuit ar fi implicat mai multe instrucțiuni executate de procesor. Deci, dacă 2 thread-uri ar fi încercat să efectueze operații în același timp, ar fi putut modifica simultan aceeași variabilă (=> rezultate neașteptate sau pierderea datelor). În aceeași ordine de idei, **BlockingQueue** este un mecanism de sincronizare thread safe pentru că permite accesul la coadă în mod concurent (mai multe fire de execuție pot face operații pe coadă, fără a interfera cu activitatea altora care accesează și ele coada). Acest comportament este datorat utilizării blocării și deblocării thread-urilor. În plus, atunci când coada blocată este goală, thread-urile care încearcă să elimine elemente din ea sunt blocate și așteaptă să fie adăugate elemente de o altă coadă; când o coadă blocată este plină, thread-urile care încearcă să adauge elemente sunt blocate și așteaptă ca alte fire de execuție să elimine elemente din coadă.

5. Rezultate

Am testat cele 3 cazuri date în laborator, iar rezultatele sunt următoarele: se observă un blocaj de câteva secunde la primul test (problemă de la sincronizare)

Pentru testul 1

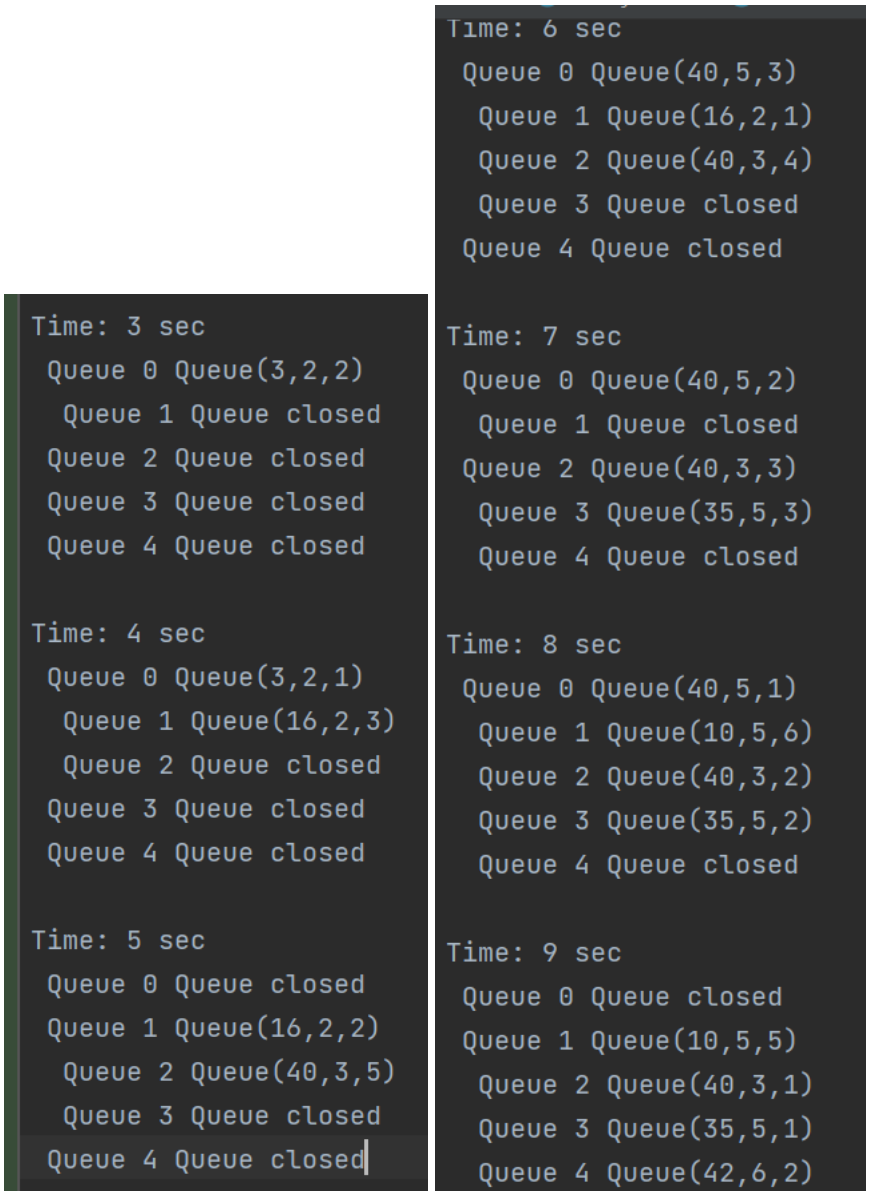
```
Time: 1 sec
 Queue 0 Queue closed
 Queue 1 Queue closed

Time: 2 sec
 Queue 0 Queue closed
 Queue 1 Queue closed

Time: 3 sec
 Queue 0 Queue closed
 Queue 1 Queue closed

Time: 4 sec
 Queue 0 Queue closed
 Queue 1 Queue closed

Time: 5 sec
 Queue 0 Queue closed
 Queue 1 Queue closed
```

```
Time: 20 sec
 Queue 0 Queue(4,19,2)
  Queue 1 Queue closed

Time: 21 sec
 Queue 0 Queue(4,19,2)
  Queue 1 Queue closed

Time: 22 sec
 Queue 0 Queue(4,19,2)
  Queue 1 Queue closed

Time: 23 sec
 Queue 0 Queue(4,19,1)
  Queue 1 Queue(2,22,2)

Time: 24 sec
 Queue 0 Queue(4,19,1)
  Queue 1 Queue(2,22,2)
```

```
Time: 26 sec
 Queue 0 Queue(4,26,3)
  Queue 1 Queue(2,22,1)

Time: 27 sec
 Queue 0 Queue(4,26,2)
  Queue 1 Queue closed

Time: 28 sec
 Queue 0 Queue(4,26,1)
  Queue 1 Queue closed

Time: 29 sec
 Queue 0 Queue closed
 Queue 1 Queue closed
```

Pentru testul 2

- Acest test pare să aibă un comportament corect



```
Time: 18 sec
 Queue 0 Queue(15,11,4)
  Queue 1 Queue closed
 Queue 2 Queue closed
 Queue 3 Queue(37,11,1)
  Queue 4 Queue closed

Time: 19 sec
 Queue 0 Queue(15,11,3)
  Queue 1 Queue(33,11,5)
  Queue 2 Queue closed
 Queue 3 Queue closed
 Queue 4 Queue closed

Time: 20 sec
 Queue 0 Queue(15,11,2)
  Queue 1 Queue(33,11,4)
  Queue 2 Queue(50,11,5)
  Queue 3 Queue closed
 Queue 4 Queue closed

Time: 21 sec
 Queue 0 Queue(15,11,1)
  Queue 1 Queue(33,11,3)
  Queue 2 Queue(50,11,4)
  Queue 3 Queue closed
 Queue 4 Queue closed
```

```
Time: 49 sec
 Queue 0 Queue(33,29,1)
  Queue 1 Queue(5,30,2)
  Queue 2 Queue(25,34,4)
  Queue 3 Queue closed
 Queue 4 Queue closed

Time: 50 sec
 Queue 0 Queue closed
 Queue 1 Queue(5,30,2)
  Queue 2 Queue(25,34,3)
  Queue 3 Queue(36,36,2)
  Queue 4 Queue closed

Time: 51 sec
 Queue 0 Queue(17,38,2)
  Queue 1 Queue closed
 Queue 2 Queue(25,34,2)
  Queue 3 Queue closed
 Queue 4 Queue closed

Time: 51 sec
 Queue 0 Queue(17,38,1)
  Queue 1 Queue closed
 Queue 2 Queue(25,34,1)
  Queue 3 Queue closed
 Queue 4 Queue closed
```

Pentru testul 3

- Acest test pare să aibă, în mare, un comportament corect (decrementează ocazional cu 2 timpul de servire)

```
Time: 15 sec
 Queue 0 Queue(580,10,1)
  Queue 1 Queue(871,10,3)
  Queue 2 Queue(785,10,3)
  Queue 3 Queue closed
 Queue 4 Queue closed
 Queue 5 Queue closed
 Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 16 sec
 Queue 0 Queue closed
 Queue 1 Queue(871,10,2)
  Queue 2 Queue(785,10,2)
  Queue 3 Queue(296,10,6)
  Queue 4 Queue closed
 Queue 5 Queue closed
 Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 20 sec
 Queue 0 Queue(544,10,3)
  Queue 1 Queue(683,10,6)
  Queue 2 Queue(329,10,8)
  Queue 3 Queue(296,10,1)
  Queue 4 Queue(454,10,4)
  Queue 5 Queue closed
 Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 21 sec
 Queue 0 Queue(544,10,2)
  Queue 1 Queue(683,10,5)
  Queue 2 Queue(329,10,6)
  Queue 3 Queue closed
 Queue 4 Queue(454,10,3)
  Queue 5 Queue(248,11,1)
  Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 22 sec
 Queue 0 Queue(544,10,1)
  Queue 1 Queue(683,10,4)
  Queue 2 Queue(329,10,5)
  Queue 3 Queue(204,11,3)
  Queue 4 Queue(454,10,2)
  Queue 5 Queue closed
 Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 99 sec
 Queue 0 Queue(709,17,3)
  Queue 1 Queue(315,17,1)
  Queue 2 Queue(844,17,2)
  Queue 3 Queue(680,17,6)
  Queue 4 Queue(167,17,4)
  Queue 5 Queue(987,17,7)
  Queue 6 Queue closed
 Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```

```
Time: 100 sec
 Queue 0 Queue(709,17,2)
  Queue 1 Queue closed
 Queue 2 Queue closed
 Queue 3 Queue(680,17,5)
  Queue 4 Queue(167,17,2)
  Queue 5 Queue(987,17,6)
  Queue 6 Queue(713,17,7)
  Queue 7 Queue closed
 Queue 8 Queue closed
 Queue 9 Queue closed
 Queue 10 Queue closed
 Queue 11 Queue closed
 Queue 12 Queue closed
 Queue 13 Queue closed
 Queue 14 Queue closed
 Queue 15 Queue closed
 Queue 16 Queue closed
 Queue 17 Queue closed
 Queue 18 Queue closed
 Queue 19 Queue closed
```