DOCUMENTATIE

TEMA 2

NUME STUDENT: ……Sorin Turda……
GRUPA: ……30222.……

CUPRINS

1. Obiectivul temei

Obiectiv principal: Proiectarea și implementarea unei aplicatii care are ca scop analiza sistemelor bazate pe cozile de așteptare prin:

- simularea unei serii de N clienți care sosesc pentru a fi serviți, intră în Q cozi, așteaptă, sunt serviți și, în final, părăsesc coada de așteptare
- calcularea timpului mediu de așteptare, a timpului mediu de servire și a orei de vârf

Obiectiv secundar:

- Analiza problemei si realizarea cerintelor
- Proiectarea aplicatiei de simulare
- Implementarea aplicatiei de simulare
- Testarea aplicatiei de simulare

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Cerinte functionale:

- Configurarea simulării: utilizatorul ar trebui să poată defini parametrii simulării.
- Pornirea simulării: utilizatorul ar trebui să poată porni simularea.
- Vizualizarea cozilor în timp real: Aplicația ar trebui să afișeze cozile de așteptare în timp real.

Cerinte non-functionale:

- Aplicatia de simulare ar trebui sa fie intuitiva si usor de inteles de catre utilizator
- Afisare grafica a cozilor

Cazuri de utilizare: configurarea simularii

Actor: utilizator

Scenariu de succes:

- Utilizatorul introduce valorile pentru: numărul de clienți, numărul de cozi, intervalul de simulare, timpul minim si maxim de sosire și timpul minim și maxim de servire
- Utilizatorul face clic pe butonul de validare a datelor introduse
- Aplicația validează datele și afișează un mesaj prin care se informează utilizatorul să înceapă simularea

Scenariu alternativ:

- Utlizatorul nu introduce corect datele (datele introduse nu sunt numere intregi)
- O eroare poate fi afisata

3. Proiectare

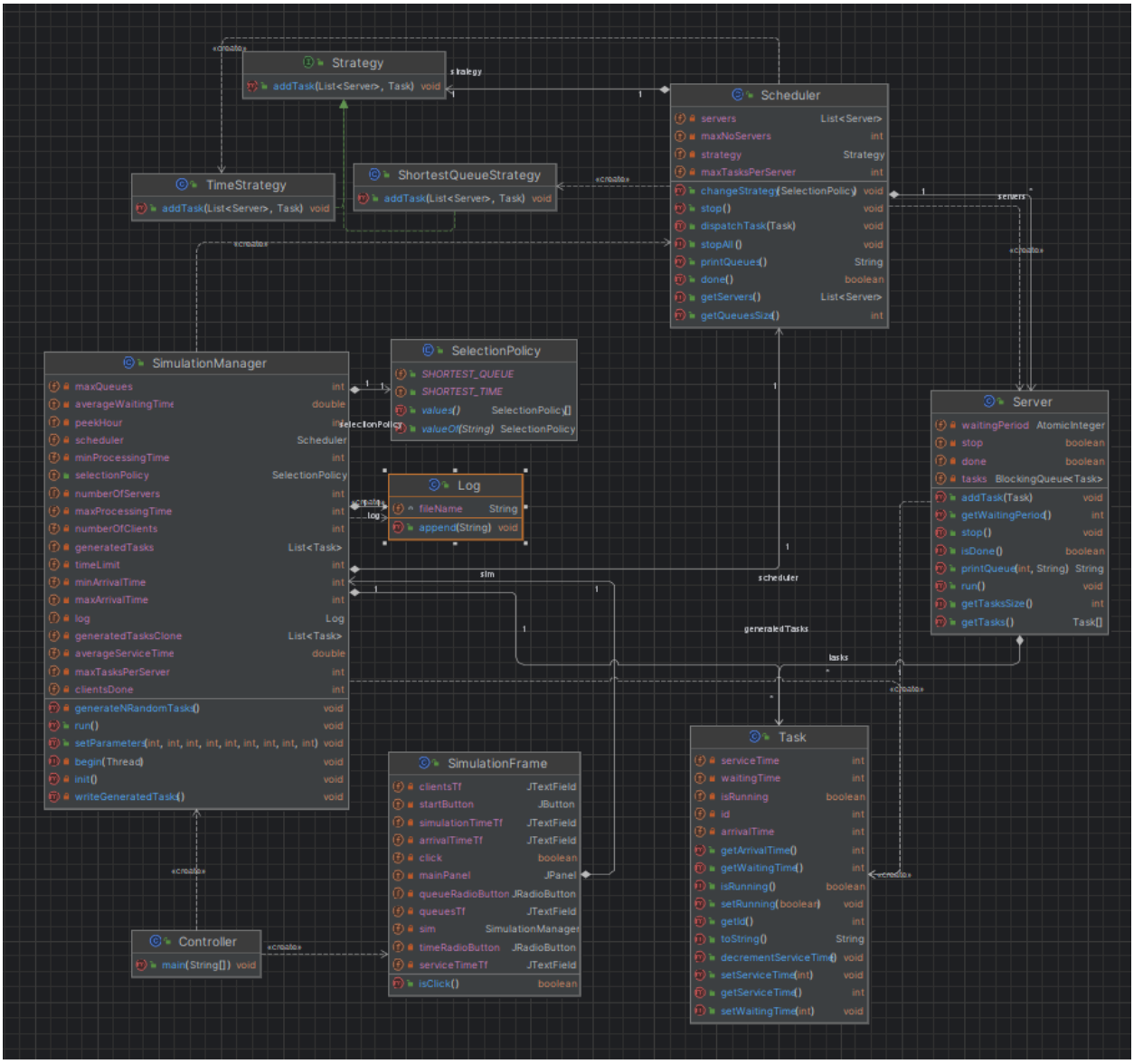

Diagrama de clase

4. Implementare

Clasa SimulationManager implementeaza Runnable

- Contine datele introduse de catre utilizator
- genereaza cei N clienti cu timpii de sosire si de servire generati aleator intr-un ArrayList
- este thread-ul principal

Medota importanta:

- run() care afiseaza si simuleaza evolutia cozilor pe parcursul timpului de asteptare

Clasa Scheduler:

- contine lista cu servere, practic cozile de asteptare + alte informatii
- repartizeaza clientii in functie de timpul de sosire si strategia aleasa

Clasa Server implementeaza Runnable:

- coada este implementata cu BlockingQueue care faciliteaza adaugarea si stergerea clientilor (daca nu este un client in coada, asteapta pana sa fie introdus, iar daca se doreste adaugarea unui nou client si coada este plina, se va adauga cand memoria va fi eliberata)

Clasa Log:

- se ocupa de scrierea rezultatelor in fisier

5. Rezultate

Au fost facute 3 simulari pe urmatoarele date:

Simularea 1:

- Numar clienti: 4
- Numar cozi: 2
- Timpul de simulare 60s
- Timpul de sosire min = 2, max = 30
- Timpul de servire min = 2, max = 4

Rezultate:
Secunda de varf: 13
Timp mediu de asteptare: 0.0048s
Timp mediu de servire: 0.334s

Simularea 2:

- Numar clienti: 50
- Numar cozi: 5
- Timpul de simulare 60s
- Timpul de sosire min = 2, max = 40
- Timpul de servire min = 1, max = 7

Rezultate:
Secunda de varf: 13
Timp mediu de asteptare: 0.26s
Timp mediu de servire: 4.06s

Simularea 3:
- Numar clienti: 1000
- Numar cozi: 20
- Timpul de simulare: 200s
- Timpul de sosire min = 1, max = 100
- Timpul de servire min = 3, max = 9

Rezultate:
Secunda de varf: 100
Timp mediu de asteptare: 0.67s
Timp mediu de servire: 5.997s

6. Concluzii

Dezvoltari ulterioare:

- O interfata grafica completa

- Adaugarea unor exceptii pentru a trata cazurile speciale

7. Bibliografie

1. https://docs.oracle.com/javase/8/docs/api/
2. https://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-and-threadpoolexecutor.html