Tehnici de programare

Tema 1

Operatii pe polinoame

Alexandru Madalina-Elena

Grupa: 30228

Anul 2

Cuprins:

1. Obiectivul temei

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

3. Proiectare

4. Implementare

5. Rezultate

6. Concluzii

7. Bibliografie

# Obiectivul temei

***Cerinta:*** Design and implement a simulation application aiming to analyze queuing based systems for determining and minimizing clients’ waiting time.

***Description:*** Queues are commonly used to model real world domains. The main objective of a queue isto provide a place for a "client" to wait before receiving a "service". The management ofqueue based systems is interested in minimizing the time amount their "clients" are waitingin queues before they are served. One way to minimize the waiting time is to add moreservers, i.e. more queues in the system (each queue is considered as having an associatedprocessor) but this approach increases the costs of the service supplier. When a new serveris added the waiting customers will be evenly distributed to all current available queues.

The application should simulate a series of clients arriving for service, entering queues, waiting, being served and finally leaving the queue. It tracks the time the customers spend waiting in queues and outputs the average waiting time. To calculate waiting time we need to know the arrival time, finish time and service time. The arrival time and the service time depend on the individual clients – when they show up and how much service they need. The finish time depends on the number of queues, the number of clients in the queue and their service needs.

***Input data***:

- Minimum and maximum interval of arriving time between customers;

- Minimum and maximum service time;

- Number of queues;

- Simulation interval;

- Other information you may consider necessary;

***Minimal output***:

- The average of waiting time, service time and empty queue time for 1, 2 and 3 queues for

the simulation interval and for a specified interval (other useful information may be also

considered);

- Log of events and main system data;

- Queue evolution;

- Peak hour for the simulation interval;

Conceptul de thread (fir de execuție) definește cea mai mică unitate de procesare ce poate fi programată spre execuție de către sistemul de operare. Este folosit în programare pentru a eficientiza execuția programelor, executând porțiuni distincte de cod în paralel în interiorul aceluiași proces.

Câteodata însă, aceste portiuni de cod care constituie corpul threadurilor, nu sunt complet independente și în anumite momente ale execuției, se poate întampla ca un thread să trebuiască să aștepte execuția unor instructiuni din alt thread, pentru a putea continua execuția propriilor instrucțiuni. Această tehnică, prin care un thread asteaptă execuția altor threaduri înainte de a continua propria execuție, se numește sincronizarea threadurilor.

În limbajul Java, threadurile sunt un element esențial pentru execuția paralelă, întrucât o aplicație ce rulează în mașina virtuală Java reprezintă un singur proces. Clasa *java.lang.Thread* poate fi utilizată pentru implementarea unui fir de execuție separat. Acțiunile ce se execută pe thread se pot defini în metoda *run()* a acestei clase (sau în cea a unui obiect ce implementează *java.lang.Runnable* și care este pasat ca argument constructorului clasei *java.lang.Thread*). Firul de execuție se pornește prin apelul metodei *start()* a acestuia.

***Abordarea problemei***: Am proiectat o interfata grafica pentru o aplicatie care simuleaza cozile de la un magazin si primeste ca valori de intrare numarul de cozi, numarul total de client din magazine, timpul minim de servire al unui client si timpul maxim de servire al unui client. Folosesc o zona de text in care este afisat in timp real statusul fiecarei cozi. La sfarsitul simularii va fi afisat si timpul mediu de servire a tuturor clientilor.

# Analiza

***Scenariu de utilizare***:

* Se introduc optiunile dorite pentru simulare si se apasa «Ok»
* Se apasa «Start» pentru inceperea simularii
* Se repeta pasii de cate ori se doreste

***Diagrama UML***:

# Proiectare

# Implementare si testare

# Rezultate obtinute

# Concluzii

# Bibliografie

1. Wikipedia
2. Diferite referate despre polinoame
3. Tutoriale de Java
4. Laboratoare de POO din semestrul 1
5. Diagrama a fost desenata folosind starUML