Bridge Crossing Problem

Dinca Gabriel
Group 3.S1
Year: III
Section:CR

December 2017

1 Problem statement

Se considera o multime de masini care incearca in mod repetat sa treaca un pod in lucru ce are o singura banda de traversare. Fiecare masina se deplaseaza intr-un singur sens, fie de la stanga la dreapta, fie de la dreapta la stanga. Masinile care se deplaseaza de la stanga la dreapta formeaza convoiul din stanga , iar celelalte masini care se deplaseaza de la dreapta la stanga formeaza convoiul din dreapta . Capacitatea de traversare a podului este limitata, in sensul ca la orice moment de timp pot sa existe pe pod maximum masini. In plus, masinile se pot deplasa pe pod doar intr-o singura directie, nefiind posibil ca doua masini ce merg in directii opuse sa se intalneasca pe pod. Se cunoaste ca exista masini in convoiul din stanga si masini in convoiul din dreapta.Sa se dezvolte un program concurent care simuleaza modul in care masinile traverseaza podul in mod repetat. Fiecare masina intra pe pod, traverseaza podul, si iese de pe pod, iar acest proces se repeta pentru fiecare masina in parte de un numar prestabilit de ori. Fiecare masina se va implementa printr-un fir de executie separat. Fiecare dintre activitatile de deplasare a unei masini in afara podului, respectiv pe pod, dureaza un interval de timp nenul. Masinile nu se pot depasi pe pod, in schimb exista posibilitatea depasirii in afara podului. Se cere:
a. Sa se schiteze o solutie a problemei si sa se explice modul de functionare, independent de o implementare particulara. Se va avea in vedere corectitudinea solutiei. Acest lucru presupune formularea explicita a conditiilor de siguranta si vivacitate si asigurarea respectarii acestora.
b. Sa se proiecteze si sa se implementeze un program concurent care simuleaza modul in care masinile din cele doua convoaie traverseaza in mod repetat podul. Se vor implementa doua metode diferite de sincronizare si coordonare folosind: i) semafoare; ii) zavoare.
c. Sa se propuna o metoda de testare a corectitudinii functionarii programului dezvoltat. Se va argumenta de ce aceasta metoda ne asigura ca se respecta proprietatile de siguranta si vivacitate.

2 Pseudocode

run())

```
1.   while (leftVehicles.isEmpty() || !rightVehicles.isEmpty())
2.     if (cossingDirection == 0)
3.       if (bridgeCars.size() < capacity)
4.         if (leftCars.size()! = 0)
5.           _mutex.lock()
6.           bridgeCars.add(leftCars.poll())
7.           bridgeCars.getLast().start()
8.           _mutex.unlock()
9.     if (crossingDirection == 1)
10.      if (bridegCars.size() < capacity)
11.        if (rightCars.size()! = 0)
12.          _mutex.lock()
13.          bridgeCars.add(rightCars.poll())
14.          bridgeCars.getLast().start()
15.          _mutex.unlock()
16.    if (Timer.timer % 10 == 0)
17.      ChangeDirection()
18.   print "All cars have crossed the bridge")
```

public void ChangeDirection()

```
1.   lastCrossingDirection = crossingDirection
2.   crossingDirection = 2
3.   while (!bridgeClear)
4.     if (bridgeCars.size() == 0)
5.       bridgeClear = true
6.     try
7.       Thread.sleep(1000)
8.     catch (InterruptedException e)
9.       e.printStackTrace();
```

public void run()

```
1.   Bridge.bridegClear = false
2.   timeCrossed = Timer.timer + crossingTime
3.   while (!bridgeTraveled)
4.     if (Bridge.bridgeCars.getFirst()! =null && Bridge.bridgeCars.getFirst().id ==
id&&timeCrossed <= Timer.timer)
5.       _mutex.lock()
6.       Bridge.bridgeCars.remove()
7.       _mutex.unlock()
8.       bridgeTraveled = true
9.     try
10.      Thread.sleep(1000)
11.    catch (InterruptedException e)
```

12. **e.printStackTrace();**
13. **print "Vehicle number " + id + " crossed the bridge coming from the " + direction==0 ? "left" : "right"**

3 Application design

Abordarea problemei propune urmatoarea strategie. Podul dispune de un numar ce reprezeinta capacitatea maxima de masini aflate pe pod, de o directie de mers care se schimba din 10 in 10 secunde, de o lista de masini aflate pe pod, de o lista de masini ce vin din directia stanga si o lista de masini ce vin din directia dreapta. O masina dispune de directia din care pleaca si de un timp necesar pentru a traversa podul. Rand pe rand fiecare prima masina din cele doua liste verifica daca ii este permisa intrarea pe pod (daca directia de traversare a podului este aceeasi cu directia din care aceasta pleaca, daca numarul maxim de masini aflate pe pod nu a atins capacitatea maxima si daca). In cazul in care aceste criterii sunt indeplinite aceasta 'intra' pe pod fiind stearsa din lista de masini initiala si introdusa in lista de masini aflate pe pod, si iese dupa ce timpul necesar de traversare s-a scurs, fiind eliminata din lista. Dupa cele 10 secunde, directia de mers se schimba si vor purcede pe pod masinile care indeplinesc conditiile mentionate mai sus din directia opusa.

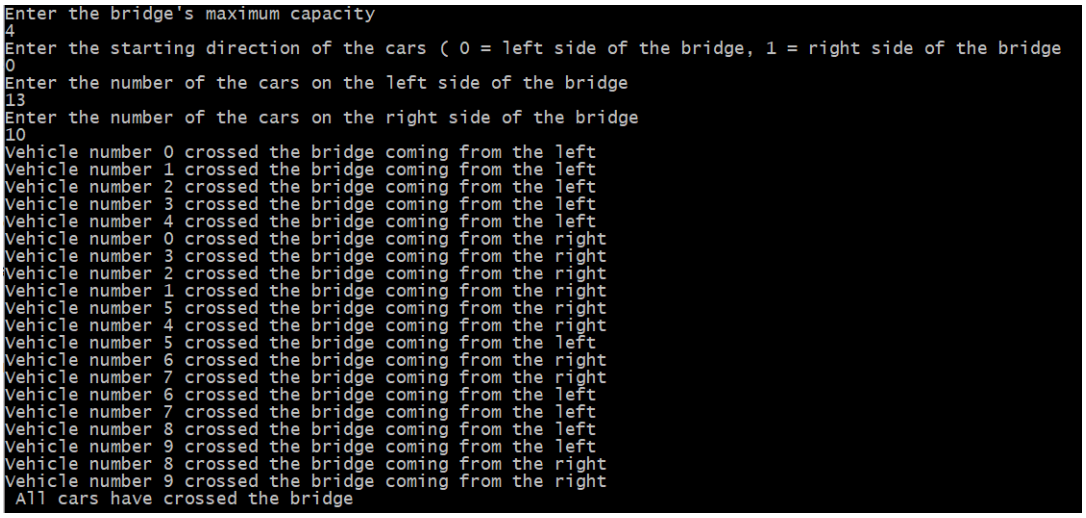
3.1 Build Instructions

Pentru rularea programului, se executa fisierul BridgeCrossing.bat (este necesara o versiune instalata de JDK)

3.2 Input data

Input:
1.Capacitatea totala a masinilor de pe pod (Ex. 4)
2.Directia din care va pleca primul convoi de masini (0 = stanga, 1 = dreapta))
3.numarul de masini din convoiul aflat in stanga
4.numarul de masini din convoiul aflat in dreapta

3.3 Output data



3.4 Modules

Fisierul Main.java Aplicatia principala.

Fisierul Bridge.java Contine clasa Bridge, care are ca membrii functia run() si ChangeDirection()

Fisierul Vehicle.java Contine clasa Vehicle care extinde clasa Thread si contine functia run() pentru fiecare vehicul.

Fisierul Timer.java contine clasa Timer care extinde casa Thread si este folosita pe post de counter pentru schimbarea directie de traversare

3.5 Functions

public void run() din clasa Bridge

Verifica daca toate masinile au trecut, controleaza intrarea si iesirea masinilor de pe pod si schimba perodic directia sensului de mers .

public void ChangeDirection()

Functie folosita pentru schimbarea directie de traversare.

public void run() din clasa Vehicle

Masinile verifica daca unt eligibile pentru a traversa podul

4 Conclusion

Aceasta tema de casa m-a ajutat sa imi dezvolt cunsotiintele legate atat de programare in limbajul Java, cat si despre notiunile de concurenta. Pe viitor doresc sa imbunatatesc aceasta aplicatie pentru utilizarea zavoarelor si imbunatatirea timpului de executie.

References

[1] Scott Oaks, Henry Wong, Java Threads (3rd ed.). Understanding and Mastering Concurrent Programming. O'Reilly Media. 2009

[2] George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair, Distributed Systems. Concepts and Design (5th edition), Addison-Wesley, 2011