**Lab 1 - "Non-cooperative" multithreading**

Bank Accounts

Domain classes:

* Transaction:
  + serialNumber – integer
  + sourceAccount
  + destinationAccount
  + amount - integer
* Account:
  + Id – integer
  + balance – integer
  + List of transactions
  + mutex

Hardware specifications:

* MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports)
* Processor: 2,3 GHz Intel Core i5
* Memory: 8 GB 2133 MHz LPDDR3
* Startup Disk: Macintosh HD

Mutexes:

In the Transaction class we have the main method which perform the transfer of the money from one account into another.

Each accout has a mutex which protects, for each run, the modification of the accounts’ balance and logs. We lock and unlock the mutex for each account in the critical section, which is in the transferMoney() method. The source and destination accounts are being locked by priority: the account with the smallest id is being locked first, so that we will avoid the deadlock.

In this mnethod, we decrease the amount of money from the source’s balance and add it to the destiantion’ s balance.

Also, for each transaction, we run a check function which verifies if the total amount of money per account is justified and if the initial sum of money for all the accounts is constant.

Tests:

There have been executed 6 tests: the first three had 20 accounts and 2transactions/thread and the other three had 50 accounts and 10 transactions/thread.

* 5 threads, 20 accounts and 2 transactions/thread exectuted in: 0.04 s
* 50 threads,20 accounts and 2 transactions/thread exectuted in: 0.072
* 100 threads,20 accounts and 2 transactions/thread exectuted in: 0.128
* 5 threads,50 accounts and 10 transactions/thread exectuted in: 0.032
* 50 threads,50 accounts and 10 transactions/thread exectuted in: 0.136
* 100 threads,50 accounts and 10 transactions/thread exectuted in: 0.184

Teofana-Ionela Moisi

Group 934/2