Документация за курсов проект по Операционни Системи

на тема

“Реализиране на пример за deadlock и предотвратяването му“

Изготвил: Йордан Димчев Димитров, КСИ, гр.49, 121216226

Проектът за реализиране на deadlock е написан на Java 8. Свързан е с банкиране и прехвърляне на сума от един акаунт в друг. Състои се от следните класове и енумерации:

- Account(id, amount) – в него се пазят сумите на клиентите на банката;

- Customer(id, account) – абстрактен клас, който служи за клиент на банката;

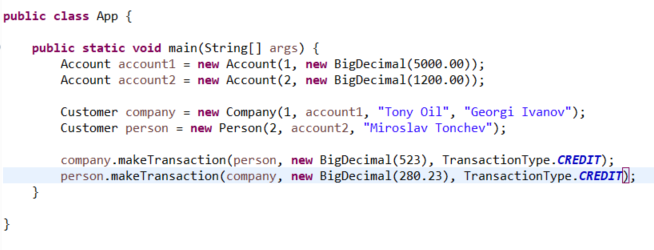
- Person(fullname) – наследник на Customer, който ще участва в транзакциите;

- Company(companyName, owner) – наследник на Customer, който участва в транзакциите;

- ConsoleLogger – помощен клас за логване на прогреса на процесите в конзолата;

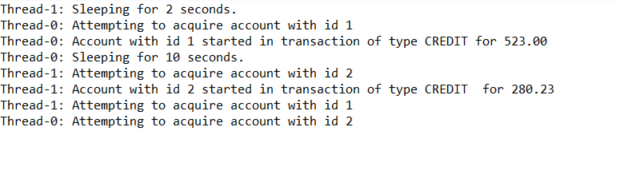
- TransactionType(DEBIT, CREDIT) – енумерация за тип на транзацията.

Ако имаме две инстанции на Customer, едната на Person, а другата- Company. Идеята за постигане на deadlock е, че когато се извършва транзакция между акаунти, те се заключват от текущата нишка един по един. За примера са нужни две успоредни транзакции между два еднакви акаунта – Т1, започната от Company и Т2 – от Person. Класовете имат метода makeTransaction, който приема като параметри втория клиент от транзакцията, сума и тип на транзакцията. Държанието на метода за Person и Customer е идентично, с разликата, че при Person се изчаква 2 секунди, преди да се започне T2, за да сме сигурни, Т1 е първа, а съответно при Company, след като заключи собствения си акаунт, заспива за 10 секунди, като по този начин се подсигуряваме, че Person ще започне Т2 преди Т1 да е заключила и неговия акаунт. Така Т1 държи заключен акаунта на Company, а Т2 – на Person, и всяка чака да се освободи другия акаунт, за да приключи транзакцията, стигайки до deadlock.



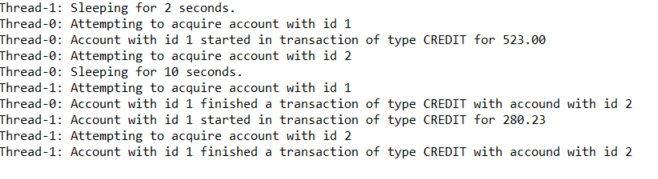
За заключването на акаунтите използваме synchronized block, в който се извършват сметките по заключения акаунт. В този блок има втори synchronized block, който изпълнява същото за втория акаунт от транзакцията.

От резултатът в конзолата се вижда, че нито една от транзакциите не се изпълнява докрай, причината за което е настъпилият deadlock.

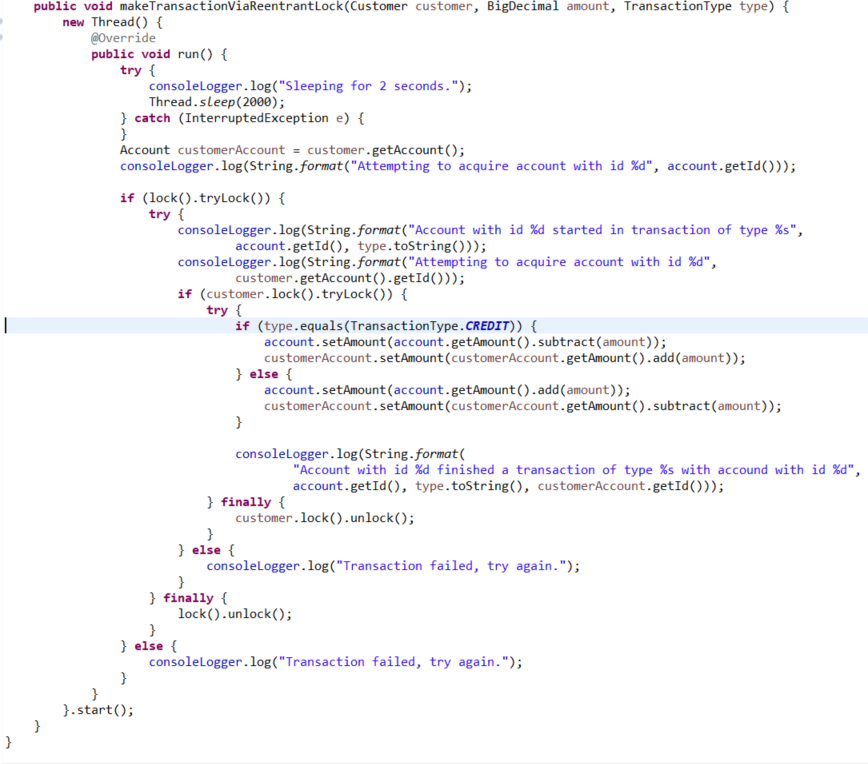


За да се защити системата от този deadlock са използвани два подхода. Първият е чрез приоритизиране на акаунта, който първи ще бъде заключен в една транзакция. За горния пример ако и двете транзакции започват със заключване на акаунта на Company, а след това този на Person, то deadlock-а няма да присъства. За сравняване на акаунтите взимаме константна стойност, която отличава акаунтите- id. Няма значение в какъв ред ще ги приоритизиране, но да речем че за този пример, при транзакция първо ще се заключва акаунта с по-малка стойност на id.

И двете транзакции приключват успешно, тъй като няма как транзакция със заключен акаунт да чака за акаунт с по-малка стойност на id-то от вече заключения.



Вторият начин за избягване на deadlock, представен в проекта, е чрез проверка дали акаунта е заключен преди опит за заключване. Ако е, се освобождава заключеният от нишката ресурс. За тази цел се изполва Reentrant Lock от java.utils.concurrent.locks, който има нужните за тази цел методи.



Така едната нишка не успява да присвои заключения от другата акаунт и отключва своя, съобщавайки, че не е успяла да извърши транзакцията, докато другата получава нужния ресурс и приключва успешно.

