Object-Oriented Programming 2

Rolf Haenni & Andres Scheidegger

Exercises 1

1. Multi-Threading

Write a simple class `BancAccount` with the following methods:

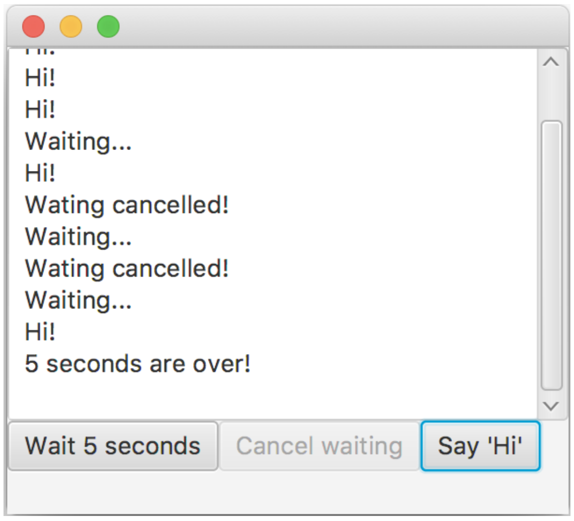
- `void deposit(int amount)`: deposits the given amount
- `void withdraw(int amount)`: withdraws the given amount (throws an exception if amount>balance)
- `int getBalance()`: returns the current balance
- `void randomTransfer(BankAccount other)`: transfers a random amount to another account (0 ≤amount≤balance)

Write a main program which generates n bank accounts $A_1, \ldots, A_n$ (each of which with an initial balance of CHF100) and m threads. Each thread repeatedly picks at random two bank accounts A_i and A_j, $i \neq j$, and transfers a random amount from A_i to A_j. Terminate the threads after 1 second.

Test your program for different values n and m to see if it contains race conditions, for example by checking that at the end the sum of all balances is n times CHF100 or by observing if exceptions are thrown. If yes, try to solve the race conditions using synchronized methods/blocks or locks. Test your program to see if it generates deadlocks. If yes, modify your implementation to avoid them.

2. Background Threads in JavaFX

Take the example of using the `Task` class for executing background work in a separate thread. Modify the example such that the task can be canceled by clicking an additional button (see picture below).



```
Hi!
Hi!
Waiting...
Hi!
Wating cancelled!
Waiting...
Wating cancelled!
Waiting...
Hi!
5 seconds are over!
```