Scala Project Part 2 – Executors, Atomicity, and Concurrent Collections

TDT4165 – Programming Languages 2015

1 Exercise Description

Traditional online banking applications are currently experiencing great competition from new players in the market who are offering direct transactions with a few seconds of response time. Banks are therefore looking at possibilities of changing their traditional method which involves batch transactions at given times of day with hours in between. They must now update their software systems to adapt to the current demand, which means transactions must be handled in real-time.

Your overall task for this project is to implement features of a real-time banking transaction system.

In the zipped folder `Exercise` is the source code for Part 2 of the project. Unzip the folder and import its contents to the Scala IDE of your choice. The file structure is presented below.

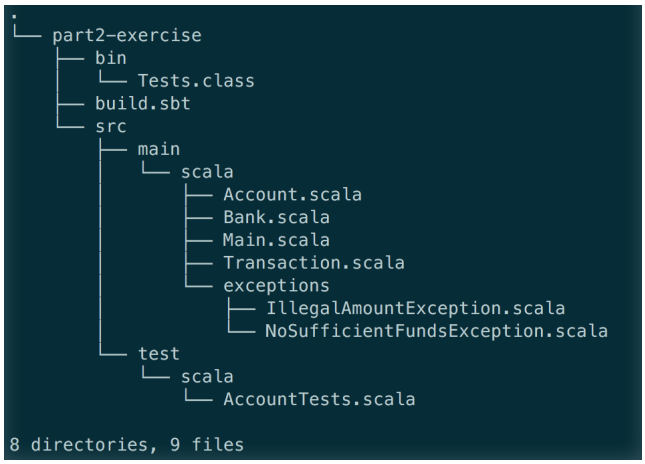

Figure 1: File structure

In this part, you will implement transactions using the `Executor` abstraction. New transactions should be added to a concurrent collection `TransactionQueue`, and from here it should be handled by the executor in `Bank`.

For this part, you will be working in `Bank.scala` and `Transaction.scala`. The main objective is to implement missing functionality and pass all of the tests. `Account.scala` is already implemented for you.

- **Bank**

`getUniqueId` is now called `generateAccountId`, and it should be thread safe. (Hint: atomicity.) You will also have to implement continuous processing of new transactions in the transaction queue.

- **Transaction** and `TransactionQueue`
  `TransactionQueue` should be a concurrent collection in which common queue-operations should be implemented (these are explained in the source code). A `Transaction` object represents a transaction between two accounts and may succeed or fail, and your task here is to extend the `run` method. Failed transactions should be retried up to a variable amount of times (`allowedAttempts`).

2 Running the Tests

2.1 IDE

To run the tests within a Scala IDE, simply run `AccountTests.scala`. If you do not have the option of running this file, make sure the `src/test/scala`-directory is marked as a source folder (Eclipse) or a test source root (IntelliJ), and that your project has `scalatest.jar` in its build path (this is not necessary if you are running the tests through sbt).

2.2 Command Line

To run the tests from the command line, `cd` into the `part2-exercise`-directory, and run the `sbt test` command.

If you have not yet installed sbt, visit `www.scala-sbt.org/release/tutorial/Setup.html` and follow the installation instructions for your OS.

3 Deliverables and Deadline

To submit your solution, upload the files `Transaction.scala`, `Bank.scala`, and any other modified/added files to itslearning before **November 8th**.

Your code should be presented to a TA during lab hours before **November 13th**. For your submission to be approved, 70% test coverage is required; 9 of the 13 tests need to pass. These 9 tests *must* include Test 07: “Account IDs are unique”.