build passing coverage 88%

Transaction Processing Application

This is a transaction processing application which provides a set of features covering the specification of the requirements described here.

The application calculates and logs to terminal/console the relative account balance for a group of account transactions within a stipulated time frame and the number of transactions that are included.

Deliverables

1. Source code
2. Documentation
3. Exception documentation and recovery strategies
4. Test harness
5. Test coverage

Table of Contents

- Application design and some design decisions
- Prerequisites
- Running the test suite
- Building the source
- Running the app from terminal
- Code coverage
- Code formatting
- Some exceptions and recovery measures

Application design and some design decisions

A class diagram showing how the various pieces and components fits together can be found here. Public interface methods to the system contains code documentation describing the operation.

RelativeAccountBalance

RelativeAccountBalance implements AccountBalance interface and it's responsibility is to **collate/compute the total of credit and debit transactions**. It has a `balance()` method which returns a Result.

Invoking `Result.balance()` returns the relative balance while `Result.transactionsIncluded()` returns the number of transactions included.

This object has the following invariant. For it to be in a valid state, transactions must be `NonNull`

```
  public RelativeAccountBalance(@NonNull ITransactions<Transaction>
transactions) {
    this.transactions = transactions;
  }
```

Transactions

A Transactions object is the representation of the list of transaction entries. It implements `ITransactions<Transaction>` and has operations for retrieving credit and debit transactions. Each entry is represented as a Transaction.

For the transactions object to be in a valid state, it must satisfy the following invariants

1. `Transaction scope` and input `transactionDataSet` cannot be null.
2. `transactionDataSet` cannot be empty.

Transactions invariants and preconditions

```
  public Transactions(
      @NonNull List<Transaction> transactionDataSet, @NonNull
TransactionQueryScope queryScope) {
    this.transactionDataSet = transactionDataSet;
    this.queryScope = queryScope;
    Preconditions.checkArgument(!transactionDataSet.isEmpty(),
"Transaction dataset is empty");
  }
```

TransactionQueryScope

A TransactionQueryScope is the notion of a group of related account transactions that were created within a given TimeFrame. This object is then used to query those accounts that fall within that scope.

TransactionType

TransactionType is represented as an enum of either `PAYMENT` or `REVERSAL`

```
public enum TransactionType {
  PAYMENT,
  REVERSAL;
}
```

Credit & debit transactions

Every transaction is 2-phased. Money flows out on one end and money is received at the other end.

Credit transactions are transactions in which money flows into an account. These may also be referred to as Accounts receivable (AR). It uses `Transaction.getToAccountId()` to identify these cases.

Debit transactions are transactions in which money flows out of an account. These may also be referred to as Accounts payable (AP). It uses `Transaction.getFromAccountId()` to identify these cases.

Prerequisites

- JDK 11+ or higher
- Maven

Running the test suite

Running this command will compile as well as run all tests

```
mvn compile test
```

Executing this command will yield the following console output

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.CreditTran
sactionsTestCase
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.171 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.CreditTran
sactionsTestCase
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.TimeFrameT
est
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.003 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.TimeFrameT
est
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nsInvariantsTestCase
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.003 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nsInvariantsTestCase
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.DebitTrans
actionsTestCase
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.DebitTrans
actionsTestCase
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nUtilsTest
```

```
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.043 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nUtilsTest
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nTest
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.011 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.Transactio
nTest
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.RelativeAc
countBalanceTest
[INFO] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 s - in
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing.RelativeAc
countBalanceTest
[INFO] Running
au.com.mebank.codingchallenge.joshluisaac.FinancialTransactionApplicationT
est
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.021 s - in
au.com.mebank.codingchallenge.joshluisaac.FinancialTransactionApplicationT
est
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 55, Failures: 0, Errors: 0, Skipped: 0
```

Building the source

This will download all the required dependencies and create an executable JAR file in the target directory. The executable JAR was created using Maven Shade Plugin

```
mvn clean install
```

Running the app from terminal

Execute the below command to build and execute the app from terminal.

```
mvn clean install && java -jar -DaccountId="ACC334455" -Dfrom="20/10/2018
12:00:00" -Dto="20/10/2018 19:00:00" -DcsvFile="sampleDataSet.csv"
target/mebank-codingChallenge-Joshua-0.0.1-SNAPSHOT.jar
```

or the following command to execute the application alone after building

```
java -jar -DaccountId="ACC334455" -Dfrom="20/10/2018 12:00:00" -
Dto="20/10/2018 19:00:00" -DcsvFile="sampleDataSet.csv" target/mebank-
codingChallenge-Joshua-0.0.1-SNAPSHOT.jar
```

JVM/Command-line arguments

- AccountId: `-DaccountId="ACC334455"`

  > The accountId that would be used to query the transactions data set.

- Start Date: `-Dfrom="20/10/2018 12:00:00"`

  > Transaction start date or start of time frame.

- End Date: `-Dto="20/10/2018 19:00:00"`

  > Transaction end date or end of time frame. Must be on or after start date.

- CSV File Path: `-DcsvFile="sampleDataSet.csv"`

  > Transaction data set.

Executing the above command will produce this output. The results and the command line arguments which was used to produce that result is presented to the user.

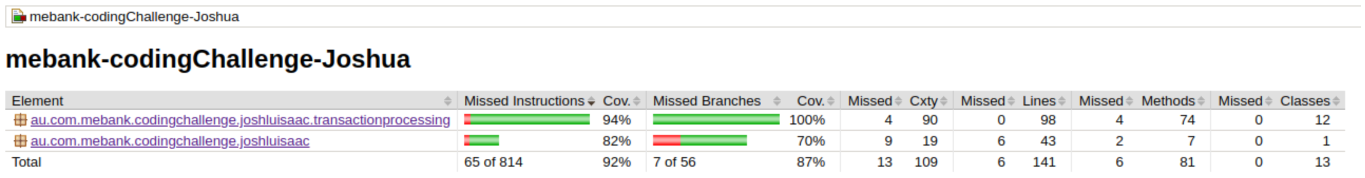
```
2019-12-17 09:57:21,159 INFO  Printing Command line arguments
>>> csvFile: sampleDataSet.csv
>>> to: 20/10/2018 19:00:00
>>> from: 20/10/2018 12:00:00
>>> accountId: ACC334455


2019-12-17 09:57:21,170 INFO  Printing results
>>> Relative  balance for the period is: -25.00
>>> Number of transactions included is: 1
```

Code coverage

Jacoco code coverage

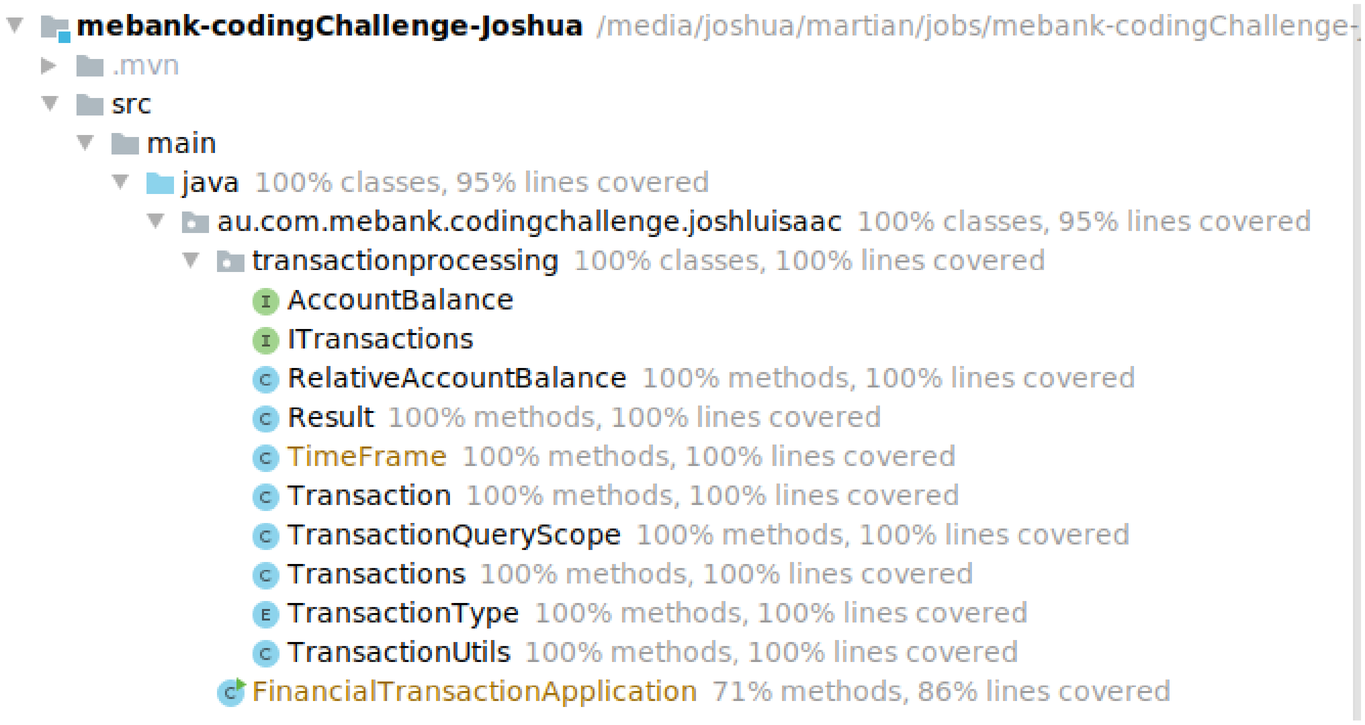
While the goal of the test harness is to cover as much edge and corner cases, that naturally led to a wider coverage of over 85%. Code coverage was both executed as part of maven build cycle using JaCoCo and from IDE



mebank-codingChallenge-Joshua

mebank-codingChallenge-Joshua

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
au.com.mebank.codingchallenge.joshluisaac.transactionprocessing		94%		100%	4	90	0	98	4	74	0	12
au.com.mebank.codingchallenge.joshluisaac		82%		70%	9	19	6	43	2	7	0	1
Total	65 of 814	92%	7 of 56	87%	13	109	6	141	6	81	0	13

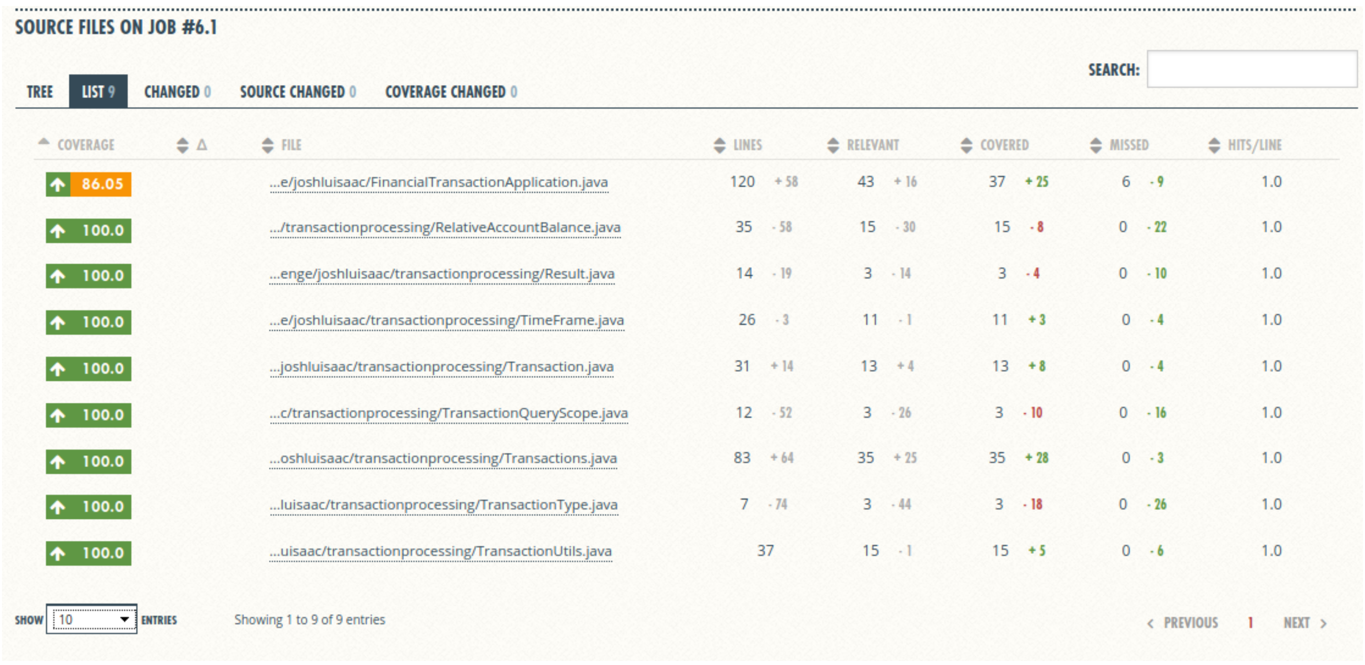


```
mebank-codingChallenge-Joshua /media/joshua/martian/jobs/mebank-codingChallenge-
  .mvn
  src
    main
      java 100% classes, 95% lines covered
        au.com.mebank.codingchallenge.joshluisaac 100% classes, 95% lines covered
          transactionprocessing 100% classes, 100% lines covered
            AccountBalance
            ITransactions
            RelativeAccountBalance 100% methods, 100% lines covered
            Result 100% methods, 100% lines covered
            TimeFrame 100% methods, 100% lines covered
            Transaction 100% methods, 100% lines covered
            TransactionQueryScope 100% methods, 100% lines covered
            Transactions 100% methods, 100% lines covered
            TransactionType 100% methods, 100% lines covered
            TransactionUtils 100% methods, 100% lines covered
          FinancialTransactionApplication 71% methods, 86% lines covered
```

Coverall report

Executing the following command will generate Jacoco and coveralls coverage reports.

```
mvn clean test jacoco:report coveralls:report
```



SOURCE FILES ON JOB #6.1

TREE | LIST 9 | CHANGED 0 | SOURCE CHANGED 0 | COVERAGE CHANGED 0 SEARCH:

COVERAGE	Δ	FILE	LINES		RELEVANT		COVERED		MISSED		HITS/LINE
86.05		...e/joshluisaac/FinancialTransactionApplication.java	120	+ 58	43	+ 16	37	+ 25	6	- 9	1.0
100.0		.../transactionprocessing/RelativeAccountBalance.java	35	- 58	15	- 30	15	- 8	0	- 22	1.0
100.0		...enge/joshluisaac/transactionprocessing/Result.java	14	- 19	3	- 14	3	- 4	0	- 10	1.0
100.0		...e/joshluisaac/transactionprocessing/TimeFrame.java	26	- 3	11	- 1	11	+ 3	0	- 4	1.0
100.0		...joshluisaac/transactionprocessing/Transaction.java	31	+ 14	13	+ 4	13	+ 8	0	- 4	1.0
100.0		...c/transactionprocessing/TransactionQueryScope.java	12	- 52	3	- 26	3	- 10	0	- 16	1.0
100.0		...oshluisaac/transactionprocessing/Transactions.java	83	+ 64	35	+ 25	35	+ 28	0	- 3	1.0
100.0		...luisaac/transactionprocessing/TransactionType.java	7	- 74	3	- 44	3	- 18	0	- 26	1.0
100.0		...uisaac/transactionprocessing/TransactionUtils.java	37		15	- 1	15	+ 5	0	- 6	1.0

SHOW 10 ENTRIES Showing 1 to 9 of 9 entries ‹ PREVIOUS 1 NEXT ›

Code formatting

Source code was formatted using google-java-format