Exercises: Test Driven Development

This document defines the exercise for the "Java Advanced" course @ Software University.

Chainblock

Your good old friend John has invited you to join his team in the creation of a one of a kind database. You are ready to revolutionize the fintech industry beyond blockchains and payment solutions. You have the honor of taking the first step in the creation process. You need to implement a really fast data structure that will hold Transactions.

`TransactionImpl` will hold:

- **`int id`** – unique transaction id
- **`TransactionStatus status`** – enumeration for transaction status
- **`String from`** – the sender of the transaction
- **`String to`** – the receiver of the transaction
- **`double amount`** – the amount of the transaction

You need to support the following operations (and they should be **fast**):

- **`add()`** – Add a transaction to the record. You will need to implement the **`contains(Transaction)`** methods as well you need to store only unique transactions by id.
- **`contains(Transaction)`** – checks if a given transaction is present in the record. Keep in mind that **transaction id is the only unique identifier**.
- **`contains(id)`** – checks if a transaction with the given **`id`** exists in the record.
- **`getCount`** – returns the number of transactions in the record.
- **`changeTransactionStatus(id, status)`** – changes the status of the transaction with the given id or throws **`IllegalArgumentException`** if no such transaction exists.
- **`removeTransactionById(id)`** – remove the transaction from the record if the id exists, otherwise throw **`IllegalArgumentException.`**
- **`getById(id)`** – return the transaction with the given **`id`**. If such transaction doesn't exist, throw **`IllegalArgumentException`**.
- **`getByTransactionStatus(status)`** – return the transactions with the given **status ordered by amount descending**. If there are no transactions with the given status, throw **`IllegalArgumentException.`**
- **`getAllSendersWithTransactionStatus(status)`** – returns all senders which have transactions with the given status ordered by transactions amount (if there are multiple transactions with the same sender, return them all). If no transactions exist, throw **`IllegalArgumentException.`**
  Example:
  "Sam" has 3 sent transactions -> 2 of them successful (5 leva and 6 leva sent) and 1 aborted transaction.
  "Peter" has 1 sent transaction and it is successful (2leva sent).
  The result of the call should be "Sam", "Sam", "Peter".
- **`getAllReceiversWithTransactionStatus(status)`** – returns all receivers which have transactions with the given status in the same way as "**`getAllSendersWithTransactionStatus`**". Throw **`IllegalArgumentException`** if no such transactions are present in the record.
- **`getAllOrderedByAmountDescendingThenById()`** – returns all transactions ordered by amount descending and by id.

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

- **`getBySenderOrderedByAmountDescending(sender)`** – search for all transactions with a specific sender and return them ordered by amount descending. If there are no such transactions throw **`IllegalArgumentException.`**
- **`getByReceiverOrderedByAmountThenById(receiver)`** – returns all transactions with a particular receiver ordered by amount descending, then by id ascending. If there are no such transactions throw **`IllegalArgumentException.`**
- **`getByTransactionStatusAndMaximumAmount(status, amount)`** – returns all transactions with given status and the amount less or equal to a maximum allowed amount ordered by amount descending. Returns an empty collection if no such transactions were found.
- **`getBySenderAndMinimumAmountDescending(sender, amount)`** – returns all transactions with the given sender and amounts bigger than the given amount ordered by amount descending. If there are no such transactions throw **`IllegalArgumentException.`**
- **`getByReceiverAndAmountRange(receiver, lo, hi)`** – returns all transactions with the given receiver and amount between lo (inclusive) and hi (exclusive) ordered by amount descending then by id. If there are no such transactions throw **`IllegalArgumentException.`**
- **`getAllInAmountRange(lo, hi)`** – returns all transactions within a range by insertion order (the range is inclusive). Returns an empty collection if no such transactions were found.

The input will always be valid.

Input / Output

You have given an **IntelliJ Java project**(unfinished project) holding the interface **`Chainblock`**, the unfinished classes **`ChainblockImpl`** and **`TransactionImpl`**.

Your task is to **finish this class** to make the tests run correctly.

- You are **not allowed to change the interface**.
- You can add to the **`TransactionImpl`** class, but don't remove anything.
- You can edit the **`ChainblockImpl`** class if it implements the **`Chainblock`** interface.

Interface

The interface looks like the code below:

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

```java
public interface Chainblock extends Iterable<Transaction> {

    int getCount();
    void add(Transaction transaction);
    boolean contains(Transaction transaction);
    boolean contains(int id);
    void changeTransactionStatus(int id, TransactionStatus newStatus);
    void removeTransactionById(int id);

    Transaction getById(int id);

    Iterable<Transaction> getByTransactionStatus(TransactionStatus status);
    Iterable<String> getAllSendersWithTransactionStatus(TransactionStatus status);
    Iterable<String>getAllReceiversWithTransactionStatus
            (TransactionStatus status);

    Iterable<Transaction> getAllOrderedByAmountDescendingThenById();
    Iterable<Transaction> getBySenderOrderedByAmountDescending(String sender);
    Iterable<Transaction> getByReceiverOrderedByAmountThenById(String receiver);
    Iterable<Transaction> getByTransactionStatusAndMaximumAmount
            (TransactionStatus status, double amount);

    Iterable<Transaction> getBySenderAndMinimumAmountDescending(String sender, double amount);
    Iterable<Transaction> getByReceiverAndAmountRange(String receiver, double lo, double hi);
    Iterable<Transaction> getAllInAmountRange(double lo, double hi);
}
```

SoftUni

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us: