This is a standalone application implemented using Java (JDK1.7).

**Libraries used:**

* JDK1.7
* JUnit 4

As , there is a limitation to implement this application of having a maximum of 2 external libraries, it is implemented without any framework, using core java design patterns.

**Solution Approach**

1. Instructions are created via facade by using InstructionBuilder interface which has a logic to create Instructions. It’s corresponding implementer class is a singleton class which creates the instructions and stores them in a list which is a single instance of the list throughout the program run.
2. In InstructionBuilder, while saving the instructions, it’s incoming/outgoing amount is calculated and date adjustments are made. i.e., if the settlement date falls on weekends, it is incremented to the next working day considering the currencies.
3. Once the instructions are created, service class methods are called via facade layer which has the business logic to calculate the total incoming and outgoing amount for the day based on the amount calculated as price per unit \* agreedFx \* units in the InstructionBuilder.
4. In the service class, firstly, the instructions that qualify for report generation are fetched. If the settlement date falls on the report generation date, then only it is qualified for total amount calculation and ranking for that day.
5. Test cases are written using JUnit to test each and every functionality of the program.

**Assumptions**

1. All the instruction data are in memory and directly passed via facade to instruction builder from the main function.
2. The settlement date is kept as the day on which the report would run to get it qualified for report generation.
3. The output is in simple text and no UI is used.
4. Test cases are created individually as per test classes.

**Class Diagram**

