Pre-paid Card System

Guillermo Román

Programming Project

Escuela Técnica Superior de Ingenieros en Informática
Universidad Politécnica de Madrid

November 13, 2019

1 Norms

- A first meeting with the teacher must take place between the following dates:

  10th December 2019 - 20th December 2019

  This meeting is the presentation of the first *sprint* you have to do in the project. In this metting, the design of the project will be discussed with the teacher and **some requirements of the project should be working for this meeting**. The mark of this first evaluation will be the **20% of the final grade of the project**.

- The deadline for finishing the project is on:

  Friday 18th January 2018

  Modifications done after this date will not be considered.

- For the project evaluation, the project source code will be taken from the `GitLab` URL provided in the registration form, thus, check that you have the correct version pushed in the repository for the deadline date.

- A system for detecting plagiarisms will be used and the groups involved in plagiarism will be penalized with a failing grade. According to the UPM norms, other disciplinary measures can be adopted in case of plagiarism.

2 Tools

The use of the following tools and libraries is mandatory:

- `Java` 1.8 (or greater)
- `GIT` as version control system
- `JUnit` 5 for test automation
- `Maven` as build tool

2.1 GitLab

Remember that the use of GitLab as remote GIT server is mandatory. The GitLab server of our course is available at the URL:

`http://costa.ls.fi.upm.es/gitlab`

The GitLab project used to develop the assignment must fulfill the following requirements:

- The *Project visibility* must be **private**. If your code is accessible by other groups is your responsibility.
- All members of the group must be registered in GitLab and your username must be your *número de matrícula*.
- All project members must have, at least, *developer* role for accessing the project.
- Email notifications are mandatory and must be sent to all project members from the beginning of the project.
- The username `@pproject` must have access to the project with *developer* role.
- In order to monitor the activity of the project, the email `programming.project@fi.upm.es` must be included in the notification emails from the beginning of the assignment.

The repository must include a README.md file. The README.md file must contain the authors of the project. This file must also include relevant information to use the system, mainly a description of the interface and how to use it. Remember that the format of the README.md file is *Markdown*[1].

2.2 Maven

The use of Maven as build tool is also mandatory and the file `pom.xml` file must be in the root directory of the project. The project must follow the standard directory layout of Maven and details of this layout can be found at:

`https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html`

The `pom.xml` must include the dependencies needed to work with JUnit 5.20 and it must work without any manual intervention, that is, after cloning the GIT project, the command `mvn test` must properly work passing all tests of the system.

The ***groupid*** of the project must be `es.upm.pproject` and the ***artifactid*** must be the name of the project, that is, *prepaid-system*.

2.3 JUnit

It is mandatory to develop your own test suite using `JUnit 5.20` to check the correct behaviour of the program developed. Take a look to the *Testing* slides to read the good practices in testing. Remember that all tests must be automatically run by the Maven commands. Those tests that do not executed automatically by means of the Maven command will not be considered.

[1] `https://www.markdownguide.org/`

2.4 SonarQube

`SonarQube` will be used to evaluate the assignment. In addition to the functional requirements described in Section 4 your project must passed the *Quality Gate* established in `SonarQube` server. The `SonarQube` server that must be used is accessible by means of the following URL:

`http://costa.ls.fi.upm.es:9000/sonar`

Remember that he use of `SonarLint` Eclipse plugin is recommended:

`https://www.sonarlint.org/`

Its installation instructions can be found in Eclipse Marketplace:

`https://marketplace.eclipse.org/content/sonarlint`

2.5 Continuous Integration

The use of *Continuous Integration* ideas is recommended and the project should include a file `.gitlab-ci.yml` which launches `mvn test` in order to run the tests.

2.6 Agile

The project must be developed following the ideas and good practices of Agile (see the slides of Agile for details). To do so, you have to use `GitLab` **to register the *backlog* of the project**. The GitLab of the project must include the following elements:

- The project must include (at least) **two milestones**, the first one corresponding to the first sprint (December) and the second one for the final deadline of the project (January).
- To register the different backlog elements you have to use `GitLab` ***issues*** (do not forget to create the corresponding label if needed). **All backlog elements must be assigned to a milestone**.
  - **Features** must be added with label *feature*
  - **Stories** must be added with label *user-story*
  - **Bugs** must be added with label *bug*
  - **Work-items** must be added with label *work-item*
- All backlog elements must also have a **priority label** (*high*, *medium*, *low*) (do not forget to create them label if needed)
- **Planning and review meetings** must be registered as an issue (open and closed inmediately) with label *meeting*

2.7 Graphical User Interface

The application **must include a *graphical user interface* (GUI)** to interact with the user. It is recommended to use the classes of *Java Swing* graphical library to develop the GUI of the application, however, other libraries like *JavaFX* can be used.

3 Introduction

This document describes the requirements of a program for managing pre-paid cards and pay with them. Prepaid systems refers to services paid for in advance. In the case of pre-paid credit cards, are cards whose balance only depends on the ammount of money charged on them. The use of prepaid credit cards has been increased in the last years due to the huge ammount of purchases on the internet.

4 Requirements

The following list summarizes the requirements of the system that must be implemented:

1. The system offers the following operations:
   (a) Buy a card
   (b) Pay
   (c) Charge money
   (d) Change PIN
   (e) Consult balance
   (f) Consult movements
2. The user can buy a new card. The credit card number (12-digit) will be **sequentially** generated and the expiration date will be one year after the buying date. The user must introduce the following data in the operation form:
   - Name
   - Surname
   - PIN number (4 digits and double field to check it)
   - Amount

   The generated card must be registered in the system and it must show a ticket with the following information:

   ```
   Dear $NAME $SURNAME,

   Card Number: $CARDNUMBER
   Balance: $BALANCE

   Thanks for using our system
   ```

3. Note that the PIN number cannot be saved in *plain text*, it must be saved *digested* by applying a hash function (e.g. function SHA256). A hash function is a *mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash) and is designed to be a one-way function, that is, a function which is infeasible to invert* [5]. Java API can be used to generate the hash function of the PIN number and we have to save this hash value of the PIN number.
4. The user can *charge* some amount of money in a card. The form will ask the user for the credit card number, the ammount of money and the PIN. The PIN must be checked before accepting the payment. If the operation succedded, the system must show a receipt similar to this one:

   ```
   Dear $NAME $SURNAME,

   Amount: $AMOUNT
   Card Number: XXXX XXXX $LAST4DIGITS
   Balance: $BALANCE

   Thanks for using our system
   ```

5. The user can *pay* an amount of money by introducing the credit card number, the amount to pay and the PIN. The system must check whether the user have enough money in the card and, if so, decrement from the balance the ammount introduced. The pin number must be checked before accepting the payment. If the operation succedded, the system must show a receipt similar to this one:

```
Dear $NAME $SURNAME,

Amount: $AMOUNT
Card Number: XXXX XXXX $LAST4DIGITS
Balance: $BALANCE

Thanks for using our system
```

6. The user can *change the PIN* by introducing the old PIN and the new one. If the old PIN is incorrect the system must show an error.
7. The user can check the balance of a card by introducing the card number and its PIN, and, if they are correct, the system must produce the following ticket:

```
Dear $NAME $SURNAME,

Card Number: XXXX XXXX $LAST4DIGITS
Balance: $BALANCE

Thanks for using our system
```

8. The user can consult the movements done with a card in the current session by introducing the card number and the PIN. The movements must be shown in decreasing order, that is, the last movement must be shown the first one. The following ticket format must be produced:

```
Dear $NAME $SURNAME,

Card Number: XXXX XXXX $LAST4DIGITS

01/02/18    13.12
02/02/18    23.22
04/03/18    65.71
05/08/18    12.92

Thanks for using our system
```

9. The movements and the cards information must be saved on files when the application is closed and loaded when the applicatoin is open.
10. The card operations must perform the following *checks* and show an error message in the following cases:
    - The card is not registered in the system (pay, charge, change PIN, consult Balance and consult movements)
    - The PIN in incorrect (pay, charge, change PIN, consult Balance and consult movements)
    - The card does not have enough money in the card (pay)
    - The card has expired (pay, charge)

References

[1] `https://maven.apache.org/`

[2] `https://www.oracle.com/technetwork/java/javase/tech/index-137868.html`

[3] `https://git-scm.com/`

[4] `https://www.sonarqube.org/`

[5] `https://en.wikipedia.org/wiki/Cryptographic_hash_function`