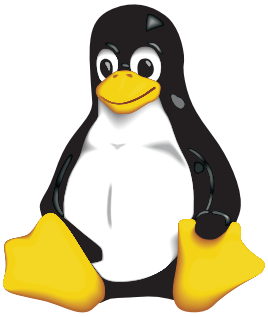

COLLEGE OF COMPUTING TECHNOLOGY - DUBLIN
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

SYSTEM ANALYSIS & DESIGN

Assessment 2 – Model an ATM Machine using UML, CASE, and XP tools

Adelo Vieira
Student Number: 2017279
Lecturer: Mr. Michael Weiss
April 23, 2018

Contents

List of Figures

1 Overview

We have been given the task of planning and modeling a Java program which represents an ATM machine. The machine would present the user with a menu of options that allow the user to do the following:

- Ask user to enter their pin code (we assume that their ATM card is inserted into the ATM)
- Display balance
- Withdraw cash
- Make a deposit
- Change their pin code
- Log out of the system

- The user would have no overdraft facility and therefore they would get an error message if they attempt to withdraw more cash then they have deposited into their account.

In this project we describe and carried out some phases of the process of developing a software for an ATM Machine.

The idea behind this project is model, using UML, CASE, and XP tools, the initial phase of the Development process.

Ir order to do so, we are going to start collecting all the information we need to create a List requirements.

We will derive the system requirements and the corresponding UML diagrams for the system from:

- The description of the assigned project.
- Our previous experience in using ATM machines:
  - We will reflect on the process of using a bank ATM for everyday transactions such as depositing and withdrawing funds using an ATM.
  - This will help us to identify the key areas of functionality that should be modeled by generating some Scenario for the use of the ATM.

We will generate a Class diagram using a **domain model** approach. Since we are using an Object-Oriented approach, this phase is crucial to have an initial idea of our classes and the relationship between them.

We will then build a set of set of CRC, a State Diagram and an Activity Diagram

2 Requirements engineering

2.1 Elicitation

In order to collect the information we need to create a List of requirements for the ATM Machine software, we are going to study the provided Flow of activity (Table 1) of the ATM.

(1) Checks if the card is locked

(2) If not:

- (a) **Asks for and accepts a PIN from the user**
- (b) **If the PIN is:**
  - (1) OK (a match with the card's pin)
    - (a) Asks for an amount to withdraw from the user and
    - (b) If the amount the user wishes to withdraw is
      - (1) less than the Account balance
        - (a) Tells the user that the withdrawal was successful
        - (b) Displays the balance
        - (c) Program ends
      - (2) Greater than the Account balance
        - (a) Program tells the user
        - (b) Asks the user if they would like to try another amount
          - (1) If yes, goes to step 2.a.1.a (this should happen every time the amount requested is higher than the account balance, not just after it happens once)
          - (2) If no, program ends
  - (2) Not OK (not a match with the card's PIN)
    - (a) Goes to step 2.a
    - (b) If this happens three times
      - (1) Card is locked
      - (2) Program ends

(3) If card is locked:

- (a) **Informs the user**
- (b) **Program ends**

Table 1: Flow of activity of the ATM

2.2 Specification

Description	Nouns (possible objects)
The machine would present the user with a Main Menu of options that allow the user to do the following: - Ask user to enter their pin code (we assume that their ATM card is inserted into the ATM) - Display balance - Withdraw cash - Make a deposit - Change their pin code - Log out of the system	ATM Machine PIN Code Balance Cash Deposit
The ATM System has to be able to read the bank card	Bank card
The ATM system also has to be able to communicate with the customer through the Key pad and the options displayed on the screen. - The customer will enter the PIN - Select the operation he wants to perform as show above in the Main Menu * Amount to withdraw	Customer Key pad Screen Main Menu Amount to withdraw
After reading the bank card, the ATM system will communicate with the Bank System in order to gather the customer banck information: - The first thing the ATM system has to verify with the Back System is if the card is locked - PIN (in order to validate PIN entered by the user) - Balance - If the user can overdraft or not	Bank System Customer bank information Overdraft
The ATM System also has to be able to communicate with the Cash Container of the Machine if order to: - Verify how much Cash is available and in which banknotes denominations - Count cash required for user - Dispense cash	Cash container of the Machine Available cash Available banknotes denominations

Table 2: Problems and requirements list

3 Class Diagram

We use the Class diagram to model classes and the relationships between classes, and also to model higher-level structures comprising collections of classes grouped into packages. [Britton and Doake (2004)]

To build our Class diagram, we are going to follow some of the steps of the **domain model** approach. The stages of this approach are (in bold we highlight the steps we have followed in this project):

- **Identify the objects and derive classes**
- **Identify attributes**
- **Identify relationships between the classes**
- Write a data dictionary to support the class diagram
- **Identify class responsibilities using CRC cards**

- Separate responsibilities into operations and attributes
- Write process specifications to describe the operations

After organizing the nouns (derived from the Problems and requirements list shown on Table **??**) into different categories, we obtained the following list:

We have to say that at this stage it is very difficult to really obtain a reliable list of classes. First because we are at a very early stage of development process and second because it is really complicate to understand all the devices that interact in a ATM and have to communicate with the software.

- ATMCard : This class could also be part of the Account class because you need to have an account to have a card.
  This class needs to communicate with the Account class and with the CardReader class

  Some parameters in this class:

  - pin
  - locked

  Some of the methods in this class:

  - pinOK()
  - lockCard()
  - isLocked()
- CardReader : This class in going to be responsible of communicate with the card reader device. This class (through the card reader device) is going to determine the User account and communicate this information to the Account class.
- Account : This class in going to be responsible communicate with the with the relevant banking system and gather all the account information of the user.

  Some parameters in this class will be:

  - Balance
  - Overdraft
- KeyPad : We are going to need a class to deal with the keypad device.
  This class needs to communicate with:

  - The Account Class
  - ScreenClass

  Some parameters in this class:

  - pinEntered
  - AmountToWithdraw

- Screen: This class is going to be responsible of display the different Menus (options) depending on the options entered by the user when pressing the corresponding buttons (KeyPad).

  This class needs to communicate with:

  - The KeyPad class in order to be able to show options depending on what is entered on the Keypad
  - The Account class in order to display options taken from the bankSystem database.

  Some of the methods in this class would be:

  - displayMainMenu()
  - displayWithdrawOptions()
  - displayDepositOptions()
  - displayOtherServices()
  - displayChangePINMenu()

- We could also need the following clases:
  - **CashDispenserDevice**
  - **CashDepositDevice:** Device of the ATM responsible for take the cash when making a deposit.
  - **CashContainerDevice**
    * Available cash
    * Available banknotes denominations

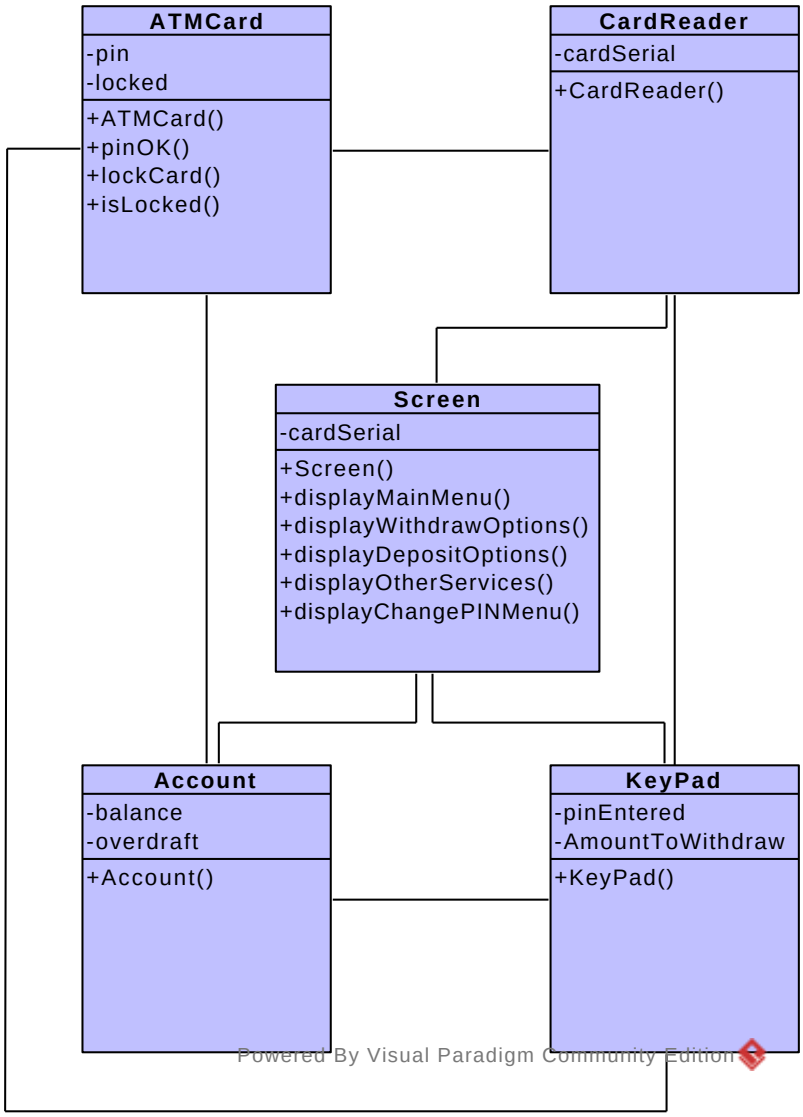

Figure 3.1: Class Diagram

4 CRC cards

A set of CRC cards modelling all classes of the user interaction with the ATM:

Account	
Responsibility	**Colloborator**
Communicate with the with the relevant banking system Gather all the account information of the user	ATMcard KeyPad Screen

Figure 4.1: CRC cards - Account

ATMCard	
Responsibility	**Colloborator**
Encapsulate the information related to the card	CardReader KeyPad Account

Figure 4.2: CRC card - ATMCard

CardReader	
Responsibility	**Colloborator**
Communicate with the card reader device Determine the User account and communicate this information to the Account class	ATMcard KeyPad Screen

Figure 4.3: CRC cards - CardReader

KeyPad	
Responsibility	**Colloborator**
Deal with the keypad device	ATMcard Account CardReader Screen

Figure 4.4: CRC cards - KeyPad

Screen	
Responsibility	**Colloborator**
Display the different Menus (options)	KeyPad Account CardReader

Figure 4.5: CRC cards - Screen

5 State Diagram

A State Diagram modelling the state of the PIN code of the ATM card (locked/unlocked)

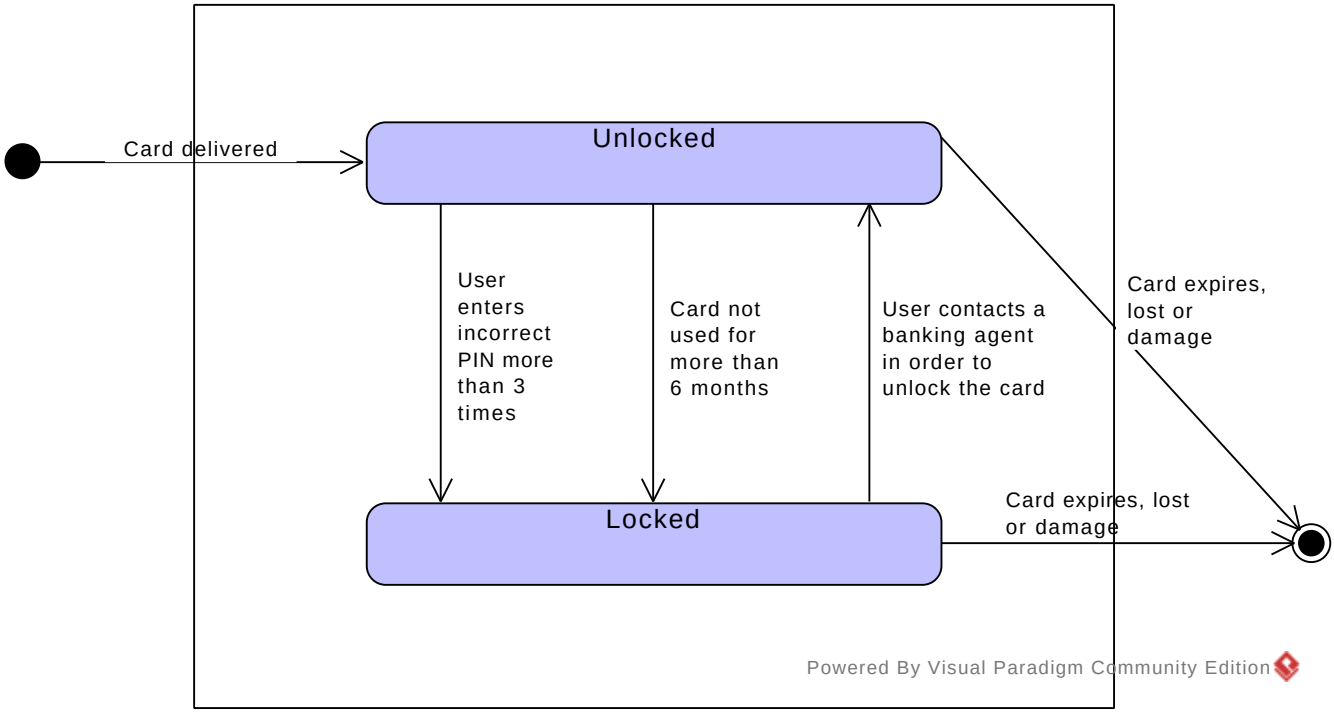

Figure 5.1: State Diagram

6 Activity Diagram

An Activity Diagram which models the entire flow of activity as shown in Table 1.

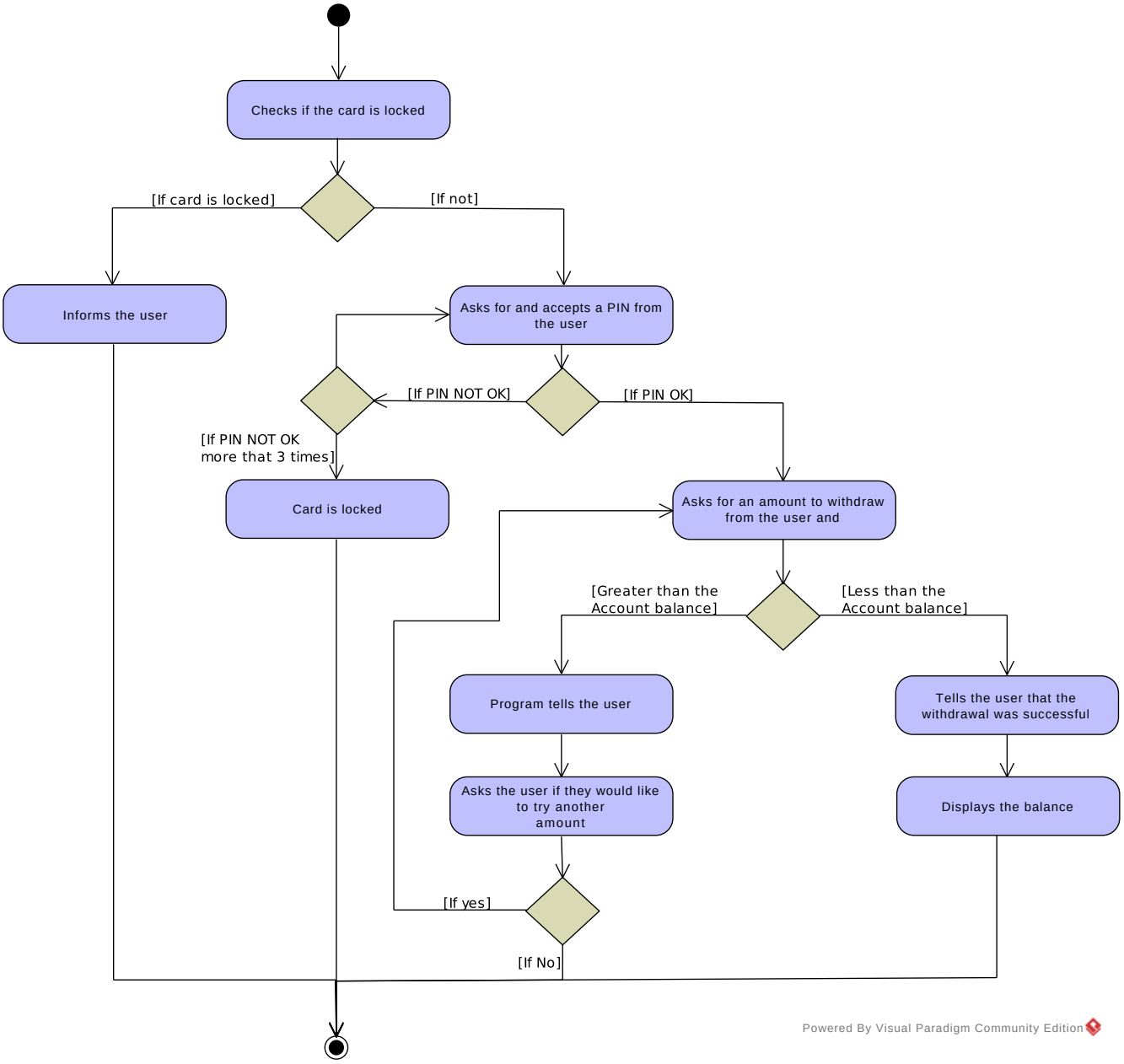

Table 3: Activity Diagram based on the provided Flow of activity shown on Table 1)

7 User Story

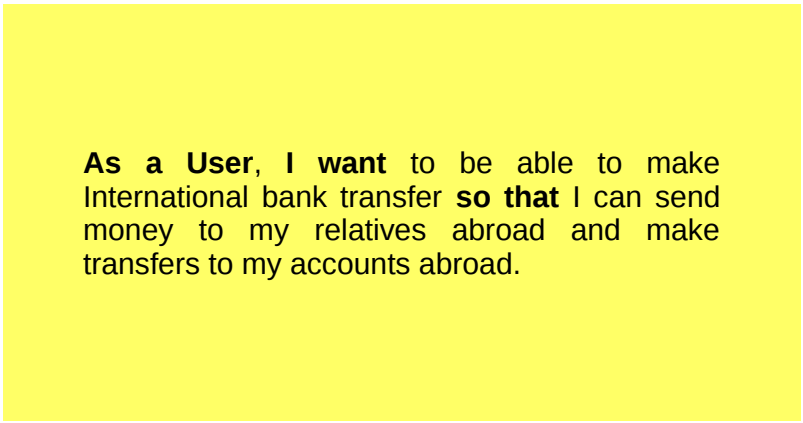

Figure 7.1: User Story, following standard Agile and eXtreme Programming (XP) practices, that details an ATM user interaction feature that we would like to have as part of the ATM that is not currently part of the typical ATM system.

8 Analysis of the sections of Java code

An analysis of the sections of Java code *Account.java* and *ATMCard.java* is shown below:

```
1  // This class is in charge of deal with the bank card.
2  class ATMCard {
3       private boolean locked = false;
4       private int pin = 1234; // The integer variable "pin" is equal to the correct PIN
5
6       // This function checks the entered PIN is valid
7       // So, it returns "true" if the user enters the correct PIN or "false" if
8       // a wrong PIN is entered
9       public boolean pinOK(int p) {
10           if(p == pin){
11                return true;
12           }
13           return false;
14      }
15
16      // This function is in charge of lock the card.
17      // If called, this function makes the variable "locked" equal to "true"
18      public void lockCard() {
19           locked = true;
20      }
21
22      // This function return the value of the private variable "locked"
23      // It can be used to verify if the bank card is locked by returning the value of "locked"
24      // So, if the value of locked is "true", the card is currently locked.
25      // In Object Oriented Programming, this kind of functions is used to access private variables
26      public boolean isLocked() {
27           return locked;
```

```
    }
}
```

```
class Account {
    private int balance = 5000;

    // This function verifies if the amount to withdraw is less than the account balance.
    // The variable "a" refers to the amount to withdraw entered by the user.
    public boolean debit(int a) {
        if((balance-a) > 0) { // if balance - amount to withdraw > 0, then
            balance = balance -a; // Update balance
            return true; // Debit successful
        }
        else { // If balance - amount to withdraw < 0, then
            return false; // Insufficient funds
        }
    }

    // This function return the value of the private variable "balance"
    // In Object Oriented Programming, this kind of functions is used to access private variables
    public int getBalance() {
        return balance;
    }
}
```

9 Testing

When developing a software, one of the most important steps in the process, is ensuring that the code that has been developed is tested. We need to make sure that the code work correctly all the time when used.

Even more, when we are developing a software for a back system, the quality and reliability of the software is extremely important.

As soon as any changes are introduced into the code, all of the tests have to be repeated to ensure that the changes that have been made do not unintentionally hurt other elements of the application that has been developed. [Goslin]

9.1 Code Level Testing

We can use a number of different tools to ensure that the code works are expected. In this kind of test we need to consider what data is passed to the function and what is returned after the function has been called. One of the most popular tool when developing in Java in the JUnit Testing. [Goslin]

9.1.1 JUnit Testing

The testing framework JUnit allows to specify what a specific method in our program should return, and if we get the answer that we are expecting then the test will pass, if we don't then the test will fail. [Goslin]

9.2 User Interface Testing

We need to test all of the different scenarios that may happen even if they seem not to be logical. For example:

- What happens if the user decides to press a button many times in a row?
- What happens if the user try to insert a card when there is already a card inside?
- What happens if the user insert an invalid card?

9.3 Test Scenarios of ATM Machine

There are a lot of many different scenarios that can be carried out when testing an ATM Machine, a complete list of scenarios is provides on artoftesting.com:

«

- Verify the slot for ATM Card insertion is as per the specification
- Verify that user is presented with options when card is inserted from proper side
- Verify that no option to continue and enter credentials is displayed to user when card is inserted correctly
- Verify that font of the text displayed in ATM screen is as per the specifications
- Verify that touch of the ATM screen is smooth and operational
- Verify that user is presented with option to choose language for further operations
- Verify that user asked to enter pin number before displaying any card/bank account detail
- Verify that there are limited number of attempts upto which user is allowed to enter pin code
- Verify that if total number of incorrect pin attempts gets surpassed then user is not allowed to continue further- operations like blocking of card etc gets initiated
- Verify that pin is encrypted and when entered

- Verify that user is presented with different account type options like- saving, current etc
- Verify that user is allowed to get account details like available balance
- Verify that user same amount of money gets dispatched as entered by user for cash withdrawal
- Verify that user is only allowed to enter amount in multiples of denominations as per the specifications
- Verify that user is prompted to enter the amount again in case amount entered is not as per the specification and proper message should be displayed for the same
- Verify that user cannot fetch more amount than the total available balance
- Verify that user is provided the option to print the transaction/enquiry
- Verify that user user's session timeout is maintained and is as per the specifications
- Verify that user is not allowed to exceed one transaction limit amount
- Verify that user is not allowed to exceed one day transaction limit amount
- Verify that user is allowed to do only one transaction per pin request
- Verify that user is not allowed to proceed with expired ATM card
- Verify that in case ATM machine runs out of money, proper message is displayed to user
- Verify that in case sudden electricity loss in between the operation, the transaction is marked as null and amount is not withdrawn from user's account

» [artoftesting.com]

Declaration

I hereby declare that all of the work shown here is my own work.

Student's Name: Adelo Vieira

Student Number: 2017279

Date: April 23, 2018

Bibliography

artoftesting.com. *Test Scenarios of ATM Machine*. URL `http://artoftesting.com/manualTesting/atm.html`. 13, 14

Carol Britton and Jill Doake. *A student guide to object-oriented development*. Butterworth-Heinemann, 1 edition, November 22 2004. 3

John Erickson and Keng Siau. *Theoretical and Practical Complexity of UML*. Americas Conference of Information Systems (AMCIS), 2004.

Dr. Kyle Goslin. *Lecture of the GUI course at CCT: Java testing*. 12, 13