

# Programming Logic and Design

## CA-PLDES

### Project

Student Name: \_\_\_\_\_

Student Number: \_\_\_\_\_

Instructor: \_\_\_\_\_

Date: \_\_\_\_\_

Results:

Total: \_\_\_\_\_/100

---

# Programming Logic and Design – Project

## *Automatic Teller Simulator*

### **INTRODUCTION**

This project will allow you to apply your knowledge and skill in program logic and design. You will need to read and analyse the information provided and extract the vital pieces of information that will allow you to design the program logic and features of the solution to the problem presented. For the purposes of this project you will need to leverage the major components, features techniques and procedures that you learned in this course in order to complete the requirements of this project. You will need to create flowcharts, write pseudocode, prepare diagrams, apply modularization techniques, use the common programming structures of sequence, selection and looping and design the mainline logic as part of your final solution.

### **OBJECTIVES**

The main objectives of this project are to:

- Interpret specifications and analysis performed
- Design a solution based on the requirements and specifications
- Design the logic required for a complete program design solution

### **TIME REQUIRED**

You will require 8-10 hours to complete this project.

### **REQUIRED MATERIAL**

You will need the following material to complete this project:

- Visio (Standard or Professional version) or other Flowchart software (At a minimum, Microsoft Word can be used)
- Microsoft Word ( for document preparation)

In programming there are generally multiple possible solutions to the same problem.

Your solution may not be identical to someone else's, but that does not mean that it is wrong. Your solution will not and should not look exactly like your colleague's solution. Any good solution is acceptable if produced following the principles and guidelines presented in this course. Keep in mind that this is your first course, and the topics that have been presented, have been done so at an introductory level with the objective of introducing you to the fundamentals of programming logic and design. These same topics will be continuously re-enforced throughout the rest of the program and you will have the opportunity to expand on them as you progress. Therefore, the expected solution for this project will be one that is at an introductory level where you apply the design skills as you have learned in the course.

## SPECIFICATIONS

In this project you will be designing the logic for a basic ATM banking machine. Your ATM must have at least three screens (login keypad screen, main transactions screen, administrator screen). You will assume that for the purposes of this project, that your client data, (PIN numbers, names etc are stored in an appropriate file or database within the system. You do not have to design the storage itself, but simply access it as required for validation purposes. Your system will have one single administrator who can perform the tasks described below.

### Key Functions

The main functions of the ATM simulator may be summarized as follows:

- Before performing any transaction, the user must enter his or her name and PIN (personal identification number) on an input screen (login keypad. Since the operation of this input screen should simulate the normal operation of an ATM, the PIN should not appear as text on the screen. If the user name / PIN combination is not valid, an appropriate message must appear on the screen and the user is prompted to try again.

In addition to the message which appears after every unsuccessful attempt, if after three tries the PIN matching the name has not been entered, the application should display a message requesting the user to try using the ATM again later and not allow further attempts. The names and PINs of users must be validated using data contained in the CustomerInfo database having the following structure:

- ◆ name (String)
- ◆ PIN (String – 4 characters)
- Once access has been authorized, the main transaction screen of the application should allow the user to carry out one of the following transactions:
  - ◆ deposit
  - ◆ withdrawal
  - ◆ transfer
  - ◆ bill payment
- When a user performs an operation, the application should first ask if it should be done using a chequing or savings account, and then ask for the transaction amount. Assume that customer banking account details are stored in a general database named Accounts.dat with the following structure:
  - ◆ account type (1 character)

- ◆ PIN (String – 4 characters)
- ◆ account number (String – 16 characters)
- ◆ account balance (single)
- For a deposit, the user must enter the amount and, if required, be able to select the account type to be credited. The chequing account is the default for this transaction.
- For a withdrawal, the user must enter the amount and, if required, be able to select the account type to be debited. The chequing account is the default for this transaction subject to a maximum of \$1,000 per transaction. If the user enters an amount greater than \$1000, the system must present a message indicating that \$1000 is the maximum per withdrawal. The ATM accepts only transactions for which the amount entered is a multiple of \$10. There is no daily maximum amount apart from the user's account balance in which case the system must also inform the user if the account balance is less than the transaction amount. In addition, the ATM Machine starts its day with a full \$20,000 in funds. Each withdrawal must also be validated against the amount left in the machine. If the withdrawal amount is greater than the amount left in the machine, the system must inform the user that the machine has insufficient funds (not the user account)
- For a transfer, the user must be prompted to enter the amount and the type of transfer (from chequing to savings, or vice versa). This transaction is subject to a maximum \$10,000. The system must allow only a transfer from checking to savings, or from savings to chequing. The system must also validate that the account balance from which the amount is being transferred has sufficient funds to cover the transaction amount. ( Hint: the validation of account balance is repeated in several places → consider modularization here!)
- For a bill payment, which is done from a chequing account only, the user must enter the amount of the transaction. The chequing account is debited by the same amount. In addition, a \$1.25 fee is charged to the chequing account. The maximum per transaction is \$10,000. Again, the account from which the amount is being taken, must have sufficient funds to cover the transaction
- The application must check the account balance before doing a transaction. Any transaction that would result in a negative balance must be rejected.
- The balance of the account affected by a transaction should be updated and displayed after each transaction.
- The user should be able to do as many transactions as he or she would like to do before leaving the ATM.
- A warning message should inform the user that the ATM can no longer carry out withdrawals when there is no money available. When a withdrawal transaction event occurs for an amount greater than the balance remaining in the ATM, the ATM should advise the user they can charge the transaction amount to the amount still available in the user's account.
- When the application is initiated at start of each day, the bank's balance money is automatically refilled with up to \$5,000 for a maximum of \$20,000.

The following functions concern the ATM's functioning with respect to the system administrator and the internal mechanisms of the ATM, not the user.

- The system administrator, as any other user, must enter his or her name and PIN (personal identification number) on the same input screen. The system administrator may perform only system transactions (he or she has no personal account).
- The System Administrator must login with the unique User Name of SysAdmin and the PIN 13579. When these credentials are entered, the system must automatically display only the Administrator screen with its options
- The supervisor can cause interest to be paid to all savings accounts at the rate of 1% ( $\text{Balance} * \text{rate}/365/100$ ).
- Once access has been authorized, a special menu is displayed. This menu offers the following options:
  - ◆ pay interest
  - ◆ refill the ATM with money
  - ◆ take the ATM out of service
  - ◆ print the accounts report
- The system administrator puts in more money in batches of \$5,000. There should not be more than \$20,000 available in the ATM.
- Each withdrawal from the machine by any user must be tracked and when the entire \$20,000 is depleted, the system will automatically put itself in out of service mode

*Note: If all the key functions mentioned above are met, a maximum mark of 100% may be given.*

## **REQUIREMENTS:**

### **Model the ATM requirements:**

Make sure you read through all of the specifications and extract the key pieces of information. Be on the lookout for processes that repeat themselves in various locations, this can be helpful to apply modularization techniques. Consider the various messages that the system is required to produce and document them,

### **Design the Program Logic**

You will create the various, flowcharts and/or pseudocode to model the logic for the ATM system.

#### 1. Flowcharts and Pseudocode

Model an algorithm for each of the processes described above using a flowchart or pseudocode (you must provide more than one example of each technique in your project). That means that in some cases, you will use a flowchart to illustrate your logic while in other you will use pseudocode.

Use the same approach as used throughout the book and your assignments, ie: the common configuration for Mainline Logic (Housekeeping tasks, Detail loop tasks, and End of Job tasks). Keep in mind that there are several unique processes, each with their own detailed steps, validations inputs and outputs. Be careful that you identify the various variables that may be required to complete a process and declare them properly where applicable.

You will be able to copy and modify to save some time. Be very careful when using copy and edit that you make all required changes.

## **General Assumptions**

This project requires that information (users, accounts, machine funds balance etc) be read from a data storage source and written back to the data storage source. The data storage source could take any form (file, random or direct access, database, etc) You are not responsible for creating the read/write logic for these processes. Instead, simply use generic process indicating the reading and writing of information where applicable. For example, when the system needs to validate a user name and PIN, you do not need to design all the steps involved such as open customer file, read customer records, compare Name, etc, Simply use a generic process that illustrates reading the information and comparing it, and then determine whether it is valid or not.

You are only designing the logic, not implementing the code, or worried about any language at this point.

This is a very simplistic model of an ATM, refrain from trying to mimic a real life ATM by saying that this is not realistic, or that in real life my ATM does this or that. The goal here is to practice program logic and design. Stay within the requirements don't read more than what is provided even if it may not be what you expect from an ATM.

## **GUI DESIGN**

This ATM proposes three screens that the users will see at any given point in time. As part of any program design, the GUI design is an important component as it allows the designer to adapt the screen layouts to meet the design and functionality standards.

### **Requirements**

1. Create the three wireframes for the three main screens that your program will use. Your wireframes, must include all the options that the program will provide. Some objects on the screen will be input, objects, others will be output objects while others will invoke a procedure to occur.
2. Prepare a storyboard showing how the screens are supposed to function together.
3. Prepare an Object dictionary which lists the interface objects used in the program, where they are found ( Screen) any variables that they impact and any procedure, method or function that they invoke. ) (See pages 517-520 in your text for examples

## **CLASS DESIGN:**

Object Oriented design was briefly introduced in this course. Although we are not expecting that you will be able to create an entire solution to this project as an Object Oriented design, you should be able to identify and design a number of classes for this solution.

After Analyzing the requirements above, identify at least 3 components that can be designed as classes. You can do more if you wish, but you must produce at least 3.

Your task is to produce 3 complete class diagrams for these classes. Where applicable, you must show inheritance from the base class in your diagrams. Your class diagrams must use the proper format and naming conventions

## MARKING SCHEME

Evaluation Elements	% of mark
Analysis and Design of the general functions of the ATM	10
Logic required by the processes/methods (pseudocode and flowcharts) for each process	25
Validation of input and output	10
Modularization	10
Proper use of symbols, structure, indentation, and terminology in flowcharts/pseudocode	20
GUI design documents	15
Class diagrams	10
<b>TOTAL</b>	<b>100</b>

## WHAT TO SUBMIT

Your project must include the following:

- A project report containing:
  - ◆ a title page with your name, the submission date and your instructor's name
  - ◆ a table of contents
  - ◆ the project specifications
  - ◆ all relevant documentation including diagrams, flowcharts, pseudocode

### Penalties

- A penalty of 5% will be deducted from your mark for each day your project is late.
- Any project submitted more than 3 calendar days late will receive a maximum mark of 60%.