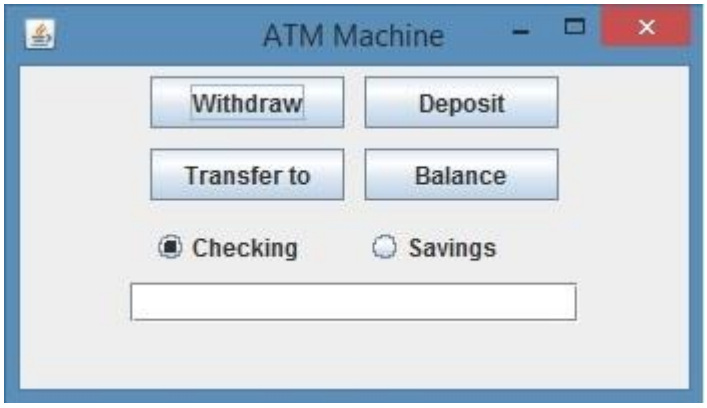
Project 2

1. Specification

The second project involves writing a program that implements an ATM machine. The interface to the program should be a Java Swing GUI that looks similar to the following:



The program should consist of three classes.

1. The first class **P2GUI** should define the GUI and should be hand-coded and not generated by a GUI generator. Class P2GUI must contain two **Account** objects, one for the checking account and another for the savings account. In addition to the main method and a constructor to build the GUI, event handlers will be needed to handle each of the four buttons shown above.

- When the **Withdraw** button is clicked, an attempt to withdraw the funds is made from the account selected by the radio buttons. The attempt might result in an exception being thrown for insufficient funds. In this situation, a **JOptionPane** window should be displayed explaining the error.

- When the **Deposit** button is clicked the specified amount should be deposited into the selected account.

- Clicking the **Transfer** button signifies transferring funds to the selected account from the other account. The transferred amount should be multiple of 10 dollars. The attempt might result in an exception being thrown for insufficient funds. In case of any error involving the transfer operation, a **JOptionPane** should be displayed explaining the error.

- Clicking the **Balance** button will cause a **JOptionPane** to be displayed showing the current balance in the selected account.

Beside the specific checks indicated for each operation, the program should also check that all user inputted amounts are valid numeric values. Also, as a result of successful execution of the Withdraw, Deposit and Transfer operations, an acknowledge message will displayed in a **JOptionPane** window.

2. The second class is **Account**. It must have a constructor with one parameter representing the initial deposit plus four methods that correspond to each of the four buttons in the GUI. It must also incorporate logic to deduct a service charge of $1.25 when five total withdrawals are made from either account. Note that this means, for example, if two withdrawals are made from the checking and two from the savings, any withdrawal from either account thereafter incurs the service charge. After the charge, the counter of withdrawals is reset. The methods that perform the withdrawals and transfers must throw an **InsufficientFunds** exception whenever an attempt is

made to withdraw or transfer more funds than are available in the account. Note that when service charges apply, there must also be sufficient funds to pay for that charge.

3. The third class is **InsufficientFunds**, which is a user defined checked exception.

Your program should compile without errors.

The Google recommended Java style guide (https://google.github.io/styleguide/javaguide.html) should be used to format and document your code. Specifically, the following style guide attributes should be addressed:

- Header comments include filename, author, date and brief purpose of the program.
- In-line comments used to describe major functionality of the code.
- Meaningful variable names and prompts applied.
- Class names are written in UpperCamelCase.
- Variable names are written in lowerCamelCase.
- Constant names are in written in All Capitals.
- Braces use K&R style**.**

In addition the following design constraints should be followed:

- Declare all instance variables private
- Avoid the duplication of code

Test cases should be supplied in the form of a table with columns indicating what aspect is tested, the input values, expected output, actual output and if the test case passed or failed. This table should contain 5 columns with appropriate labels and a row for each test case. Note that the actual output should be the actual results you receive when running your program and applying the input for the test record. Be sure to select enough different kinds of employees and situations to completely test the program.

2. Submission Requirements

Submit the following to the Project 2 assignment area no later than the due date listed in your LEO classroom.

1. All **.java** source files (**no other file types should be submitted**). The source code should use Java code conventions and appropriate code layout (white space management and indents) and comments. All submitted files may be included in a .zip file.

2. The solution description document **P2SolutionDescription** (.pdf or .doc / .docx) containing the following:
(1) Assumptions, main design decisions, error handling;
(2) Test cases table
(3) Screen captures showing successful program compilation and test cases execution. Each screen capture should be properly labeled, clearly indicated what the screen capture represents.
(4) Lessons learned from the project;

3. Grading Rubric

The following grading rubric will be used to determine your grade:

Attribute	**Meets**	**Does not meet**

P2GUI class	35 points	0 points
	a) Defines the GUI.	a) Does not define the GUI.
	b) Contains the main method and a constructor to build the GUI.	b) Does not contain the main method and a constructor to build the GUI.
	c) Contains event handlers to handle each of the four buttons.	c) Does not contain event handlers to handle each of the four buttons.
	d) Contains Withdrawal checks to ensure the value in the text field is numeric and positive.	d) Does not contain Withdrawal checks to ensure the value in the text field is numeric and positive.
	e) Provides ability to attempt to withdraw the funds is made from the account selected by the radio buttons.	e) Does not provide ability to attempt to withdraw the funds is made from the account selected by the radio buttons.
	f) An exception is thrown for insufficient funds, or if value is not numeric positive using a JOptionPane explaining the error.	f) An exception is not thrown for insufficient funds, or if value is not numeric positive using a JOptionPane explaining the error.
	g) The $1.25 charge after 5 withdrawals is correctly determined.	g) The $1.25 charge after 5 withdrawals is incorrectly determined.
	h) Provides ability to attempt Deposit when Deposit button is clicked.	h) Does not provide ability to attempt Deposit when Deposit button is clicked.
	i) Contains Deposit checks to ensure the value in the text field is numeric and positive.	i) Does not contain Deposit checks to ensure the value in the text field is numeric and positive
	j) Contains Transfer button functionality providing transferring funds to the selected account from the other account.	j) Does not contain Transfer button functionality providing transferring funds to the selected account from the other account.
	k) Contains transfer checks to confirm that the amount supplied is numeric positive and multiple of $10 and that there are sufficient funds in the account from which the funds are being transferred.	k) Does not contain transfer checks to confirm that the amount supplied is numeric positive and multiple of $10 and that there are sufficient funds in the account from which the funds are being transferred.
	l) Upon successful withdrawal /transfer/deposit, a JOptionPane window is displayed confirming that these operations have succeeded.	l) No feedback is provided to the user upon successful execution of the operations of withdrawal /transfer/deposit.
	m) Contains Balance button	m) Does not contain Balance button

	functionality that will cause a JOptionPane window to be displayed showing the current balance in the selected account. n) The main class contains two Account objects, appropriate defined, one for the checking account and another for the savings account.	functionality that will cause a JOptionPane window to be displayed showing the current balance in the selected account. n) The main class does not contain two Account objects, appropriate defined, one for the checking account and another for the savings account.
Account Class	**25 points** a) Contains a constructor plus 4 methods that corresponds to each of the four buttons in the GUI. b) Incorporates logic to deduct a service charge of $1.25 when more than four total withdrawals are made from either account. c) The methods that performs the withdrawals / transfers throw an InsufficientFunds exception whenever an attempt is made to withdraw / transfer more funds than are available in the account. d) Checks that there must be sufficient funds to pay for service charge.	**0 points** a) Does not contain a constructor plus 4 methods that corresponds to each of the four buttons in the GUI. b) Does not incorporate logic to deduct a service charge of $1.25 when more than four total withdrawals are made from either account. c) The method that performs the withdrawals / transfers does not throws an InsufficientFunds exception whenever an attempt is made to withdraw / transfer more funds than are available in the account. d) Does not check that there must be sufficient fund
InsufficientFundsException Class	**20 points** a) Is a user defined checked exception class. b) Handles the required exceptions.	**0 points** a) Is not a user defined checked exception class. b) Does not handle the required exceptions.
Test Cases	**10 points** a) Test cases are supplied in the form of table with columns indicating test case objective, the input values, expected output, actual output and if the test case passed or failed. b) Enough scenarios selected to completely test the program. c) Test cases were included in the supporting word or PDF documentation.	**0 points** a) No test cases were provided.
Documentation and Style guide	**10 points**	**0 points**

	a) Solution description document **P2SolutionDescription** includes all the required sections appropriate titled. **Source code** criteria b) Header comments include filename, author, date and brief purpose of the program. c) In-line comments used to describe major functionality of the code. d) Meaningful variable names and prompts applied. e) Class names are written in UpperCamelCase. f) Variable names are written in lowerCamelCase. g) Constant names are in written in All Capitals. h) Braces use K&R style. i) Declare all instance variables private. j) Avoids the duplication of code.	a) No **solution description document** is included. **Source code** criteria b) Java style guide was not used to prepare the Java code. c) All instance variables not declared private. d) Duplication of code was not avoided.