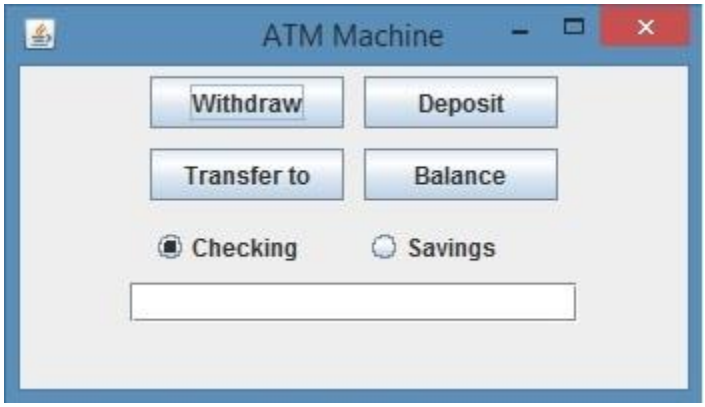
Project 2

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



The program should consist of three classes.

1. The first class should define the GUI. 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, several checks must be made. The first check is to ensure the value in the text field is numeric. Next a check must be made to ensure the amount is in increments of $20. At that point 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, If any of those three errors occur a `JOptionPane` window should be displayed explaining the error. Otherwise a window should be displayed confirming that the withdrawal has succeeded. When the *Deposit* button is clicked the only necessary check is to ensure that the amount input in the textfield is numeric. Clicking the *Transfer* button signifies transferring funds to the selected account from the other account. The checks needed are to confirm that the amount supplied is numeric and that there are sufficient funds in the account from which the funds are being transferred. Clicking the *Balance* button will cause a `JOptionPane` window to be displayed showing the current balance in the selected account. The main class must contain two `Account` objects, one for the checking account and another for the savings account.

2. The second class is `Account`. It must have a constructor plus a method that corresponds to each of the four buttons in the GUI. It must also incorporate logic to deduct a service charge of $1.50 when more than four 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. The method that performs the withdrawals must throw an `InsufficientFunds` exception whenever an attempt is made to withdraw 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.

The google recommended Java style guide, provided as link in the week 2 content, 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 table with columns indicating the input values, expected output, actual output and if the test case passed or failed. This table should contain 4 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 scenarios to completely test the program.

Note: All code should compile and run without issue.

Submission requirements

Deliverables include all Java files (.java) and a single word (or PDF) document. The Java files should be named appropriately for your applications. The word (or PDF) document should include screen captures showing the successful compiling and running of each of the test cases. Each screen capture should be properly labeled clearly indicated what the screen capture represents. The test cases table should be included in your word or PDF document and properly labeled as well.

Submit your files to the Project 2 assignment area no later than the due date listed in your LEO classroom. You should include your name and P2 in your word (or PDF) file submitted (e.g. firstnamelastnameP2.docx or firstnamelastnameP2.pdf).

Grading Rubric:

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

Attribute	Meets	Does not meet
GUI Class	35 points Defines the GUI. Contains the main method and a constructor to build the GUI. Contains event handlers to handle each of the four buttons. Contains Withdrawal checks to ensure the value in the text field is numeric. Contains Withdrawal checks to ensure the amount is in increments of $20. Provides ability to attempt to withdraw the funds is made from the account selected by the radio buttons. An exception is thrown for insufficient funds, or if value is not numeric or is value is not in $20 increment using a JOptionPane window explaining the error. Upon successful withdrawal, a window is displayed confirming that the withdrawal has succeeded. Provides ability to attempt Deposit when Deposit button is clicked.	0 points Does not define the GUI. Does not contain the main method and a constructor to build the GUI. Does not contain event handlers to handle each of the four buttons. Does not contain Withdrawal checks to ensure the value in the text field is numeric. Does not contain Withdrawal checks to ensure the amount is in increments of $20. Does not provide ability to attempt to withdraw the funds is made from the account selected by the radio buttons. An exception is not thrown for insufficient funds, or if value is not numeric or is value is not in $20 increment using a JOptionPane window explaining the error. Upon successful withdrawal, a window is not displayed confirming that the withdrawal has succeeded.

	Contains Deposit checks to ensure the value in the text field is numeric. Contains Transfer button functionality providing transferring funds to the selected account from the other account. Contains transfer checks to confirm that the amount supplied is numeric and that there are sufficient funds in the account from which the funds are being transferred. Contains a Balance button will cause a JOptionPane window to be displayed showing the current balance in the selected account. The main class contains two Account objects, one for the checking account and another for the savings account.	Does not provide ability to attempt Deposit when Deposit button is clicked. Does not contain Deposit checks to ensure the value in the text field is numeric. Does not contain Transfer button functionality providing transferring funds to the selected account from the other account. Does not contain transfer checks to confirm that the amount supplied is numeric and that there are sufficient funds in the account from which the funds are being transferred. Does not contain a Balance button will cause a JOptionPane window to be displayed showing the current balance in the selected account. The main class does not contain two Account objects, one for the checking account and another for the savings account. Code does not Compile.
Account class	25 points Contains a constructor plus a method that corresponds to each of the four buttons in the GUI. Incorporates logic to deduct a service charge of $1.50 when more than four total withdrawals are made from either account.	0 points Does not contain a constructor plus a method that corresponds to each of the four buttons in the GUI. Does not incorporate logic to deduct a service charge of $1.50 when more than four total withdrawals are made from either account.

	The method that performs the withdrawals throws an InsufficientFunds exception whenever an attempt is made to withdraw more funds than are available in the account. Checks that there must be sufficient funds to pay for service charge.	The method that performs the withdrawals does not throws an InsufficientFunds exception whenever an attempt is made to withdraw more funds than are available in the account. Does not check that there must be sufficient funds to pay for service charge. Code does not Compile.
InsufficientFundsException Class	20 points Is a user defined checked exception class. Handles all user-defined exceptions.	0 points Is not a user defined checked exception class. Does not handle all user-defined exceptions. Code does not Compile.
Test Cases	10 points Test cases are supplied in the form of table with columns indicating the input values, expected output, actual output and if the test case passed or failed. Enough scenarios selected to completely test the program. Test cases were included in the supporting word or PDF documentation.	0 points No test cases were provided.
Documentation and Style guide	10 points Screen captures were provided and labeled for compiling your code, and running each of your test cases. Header comments include filename, author, date and brief purpose of the program.	0 points No documentation included. Java style guide was not used to prepare the Java code. All instance variables not declared private.

	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**.** Declare all instance variables private. Avoids the duplication of code.	Duplication of code was not avoided.