1 INTRODUCTION
Exercise 2 is to be done as a group, with your fellow teammates. The reasoning for doing this as a group is that the most difficult part of learning a new programming language and beginning to work with a development environment comes at the beginning. However, the following exercises, which will most likely build on this exercise, will most likely be individual efforts.

You are to model in Java code an ATM machine. The classes you need to model are the patron and the account. Section 1 details the attributes and behaviors of a patron. Section 3 identifies the attributes and behaviors of an account.

2. PATRON
The patron class is the main class. In the real world, we would have a single ATM object that serves as driver for a collection of patron objects. However, for simplicity in our first exercise, we will have a single patron class that provides the dialog, including authenticating the user’s access with a password, and drives our program. The user manipulates the program through a simple menu, command line-driven interface using Standard I/O. For example:

Welcome to the Keatings Savings and Loan ATM.
Where we treat your money as our own.
What is your name? __Rayzer
What is your password? __Cheatem&Howl

Please select one of these services:
1. Withdraw money
2. Deposit money
3. Transfer money
4. All account balances
5. Add new account
6. Close existing account

The patron class holds a list of accounts. In other words, it instantiates objects of type account and manipulates them to provide the desired services. In the beginning, the patron holds one saving account with a minimum balance of $50. The user of the program can add more accounts as desired.

A patron must hold the following data:

1. Name: Of type java.lang.String, this attribute is self explanatory. You may want to hard code this as a named constant.
2. Identification Number: Also of type java.lang.String,. We will assume that this is a social security number. You do not have to enforce any check to make sure that the entered data is in the appropriate format.
3. Password. A string which you will use to authenticate the user. You can hardcode the Password as a named constant.

3. Accounts: This is a list of bank accounts that a patron has at our bank. I recommend using the `java.util.Vector` class to manage the list. The `java.util.Vector` is Java’s implementation of a linked list class. The list is necessary because any given patron could have one or more checking, savings, money market, or loan accounts.

### 2.2 Behaviors

All behaviors necessary for a patron are enumerated below:

1. Authenticate. The patron must verify that the user is who he/she says he is by checking the entered user name and password.
2. Add New Account. Opening account requires that the user specifies an account type and an account name.
3. Close an Account. Removes a specified named account from the patron’s account list and prints a message indicating the final balance of the account.
4. List All Accounts: Dump out the information from all accounts (one account per line to standard I/O).

### 3 BANK ACCOUNT

A bank account must include the following attributes and behaviors.

#### 3.1 Attributes

1. Type. CHECKING, SAVINGS or LOAN.
2. Name: A string specifying the account name.
4. Interest Rate. Another float that represents the interest rate for this account.

#### 3.2 Services

An account must support the following operations:

1. Withdraw. Subtract the specified amount from this account provided that the funds are available. Make sure that the amount being withdrawn is positive.
2. Deposit. Adds the specified amount to this account. Ensure that the amount deposited is positive.
3. Apply Interest. Make a deposit of b x r where b is the current balance and r is the interest rate for this account. This only applies to SAVINGS and LOAN type accounts.
4. Balance Inquiry. Return the current balance for this account.
5. Funds Transfer. Deposit a specified amount to this account by withdrawing it from another specified account.
6. Change Interest Rate. Set the interest rate to a new value. This only applies to SAVINGS and LOAN type accounts.
7. Printing an account should result in the following formatted string to standard I/O.
Type: Name: Rate: Balance

For example, a checking account with a balance of $1536.45 should return the following formatted string.

Checking Account: CA1: 0.0%: $1536.45

4. GRADING
You will put a hardcopy of your code in the cs2704 turn-in box located in McBryde 116/118 by 5:00 PM of the due date (Feb. 16 for MWF class, Feb. 17 for TH class). You will also send your source file as an e-mail attachment to the GTA. Send it to Wei Wang wwang@cs.vt.edu if you are in the MWF class. If you are in the TH class, send it to Scott Guyer (sguyer@cs.vt.edu).

This exercise will be graded on the following:

- Implementation of the 6 services listed above in the sample dialogue.
- Adherence to other specifications described above.
- Logical organization of code.
- Documentation of code

In your documentation, you MUST specify which section you are in, all participants, and your class’s section number. Also specify the development environment and compiler version you used.