

## CPSC 233 –Individual Assignment 8 and Assignment 09

### Requirements

Update the Customer, BankAccount and SavingsAccount classes such that they can be constructed from information stored in a file and that they can be saved to a text file. The following describe how the tests expect this information be saved to a file and read from a file.

#### Customer

- Add a constructor that takes a `BufferedReader` (import from `java.io`) as an argument. This constructor should be declared to throw `IOException` (also imported from `java.io`). Read two lines from this `BufferedReader`.  
*BufferedReader + FileReader*
  - If there are no lines in the file, use defaults for the instance variables.
  - The first line should contain the customer's name. If this line contains the text `null`, throw an `IOException` with the message 'Customer is null in file'.  
*"null"*
  - If there is a first line but no second line, throw an `IOException` with the message 'No customer ID found in file'.
- Add a method called `save` that takes a `PrintWriter` as an argument (import from `java.io`). The method should be declared to throw `IOException`. Write the name and id to the `PrintWriter` each on their own line.  
*pw.println pw*

#### BankAccount

- Add a constructor that takes a `BufferedReader` (import from `java.io`) as an argument. This constructor should be declared to throw `IOException` (also imported from `java.io`). The first line from the reader should be the balance, the second line the accountNumber. Once those have been read, create the accountHolder by invoking the Customer constructor that takes a `BufferedReader`. If this Customer constructor throws an `IOException`, set the accountHolder to null.
- Add a method called `saveToTextFile` that takes a filename (as a string) as an argument and is declared to throw an `IOException`. It should create a `PrintWriter` object using the filename provided as an argument. Then write the balance and account number, each on their own line, to the file.

If the accountHolder is null, write the string `null` to the file. Otherwise, invoke 'save' on the accountHolder and pass the `PrintWriter` object to it as an argument.

Before returning from the `saveToTextFile` method, make sure to close the `PrintWriter` object.

#### SavingsAccount

- Add a constructor that takes a `BufferedReader` as an argument. This constructor should be declared to throw `IOException`. It should invoke the parent constructor that takes a `BufferedReader` as well.

The first line that this constructor will read will contain the annual interest rate. The next line will contain the minimum balance.

- Add a method called `saveToTextFile` that takes a filename as a string as an argument and is declared to throw an `IOException`.

It should first invoke the `saveToTextFile` method in the parent, passing it the same filename.

Then create a `PrintWriter` object that opened for *appending*. Write, to the end of the file, the `annualInterestRate` and the `minimumBalance`, each on their own line.

Before returning from the `saveToTextFile` method, make sure to close the `PrintWriter` object.