Russell LaCour

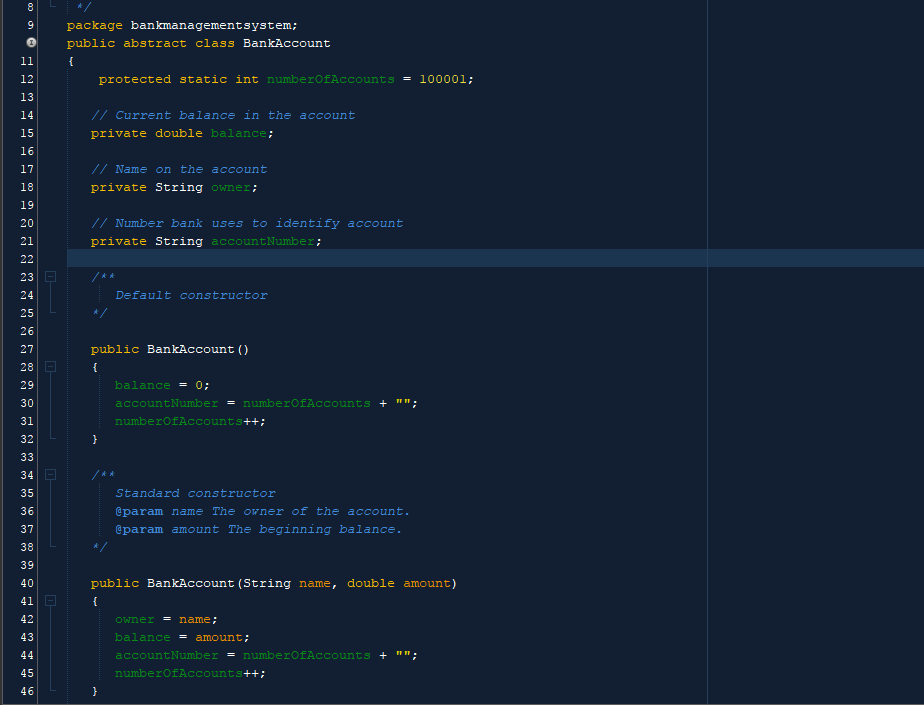
CSCI 185 M-01

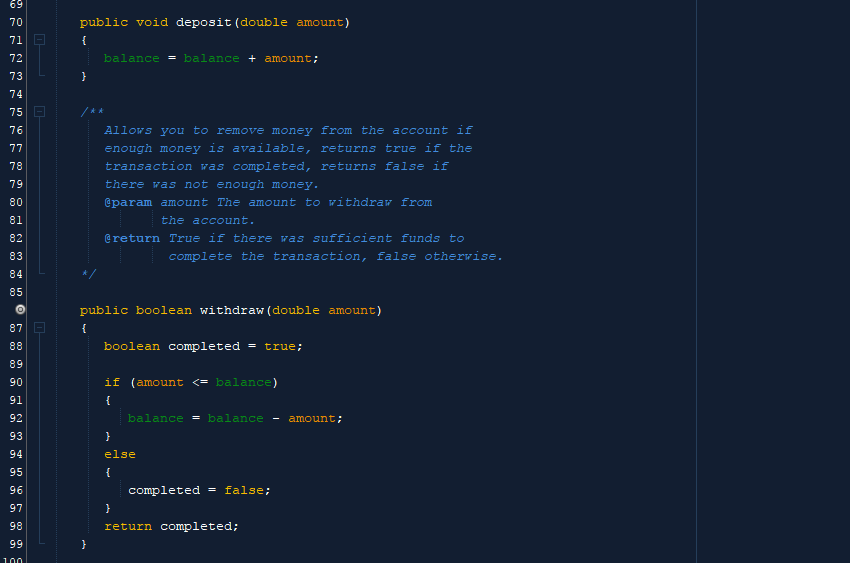
Simo

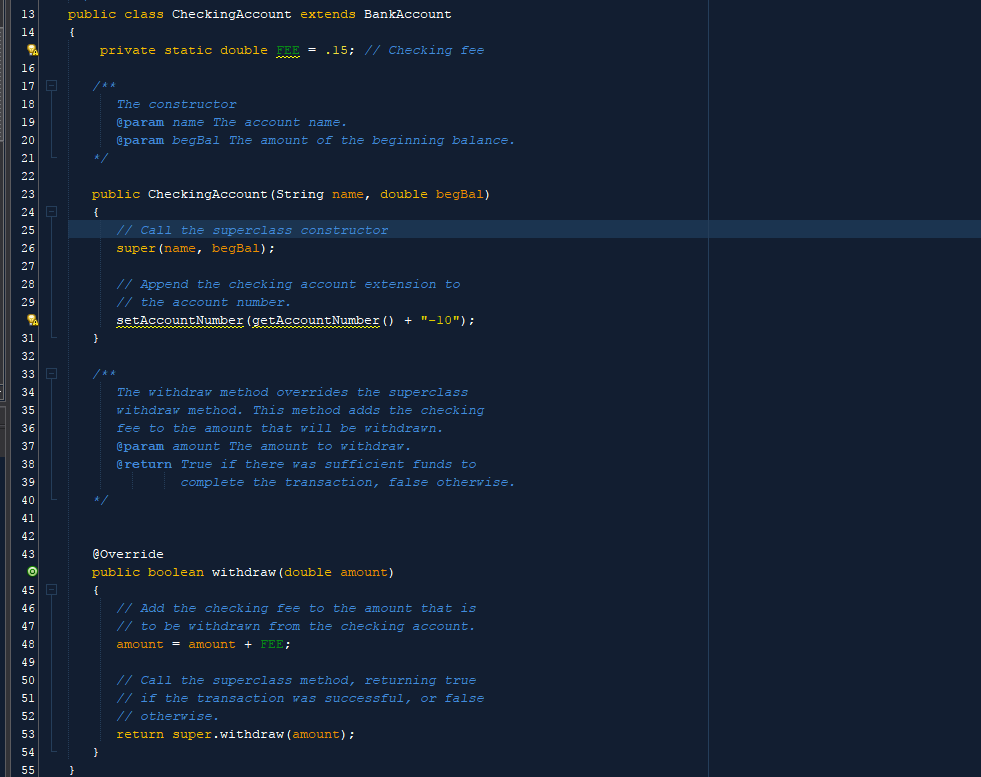
12/11/2017

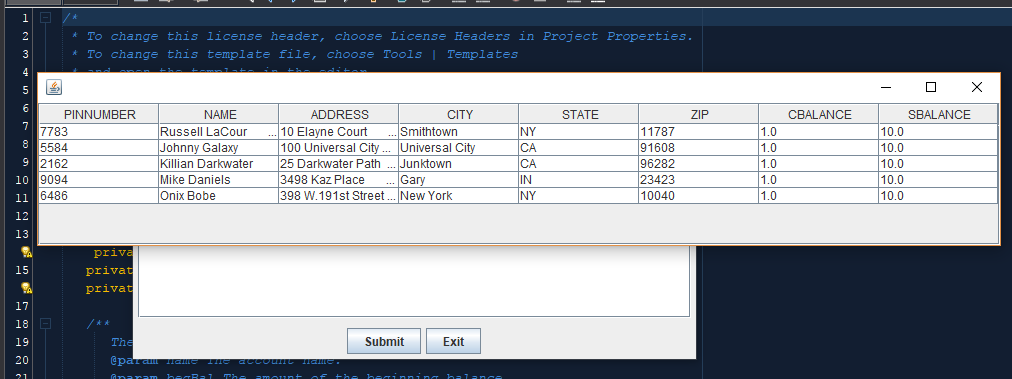
Bank Management System for Java

**Case Study:**

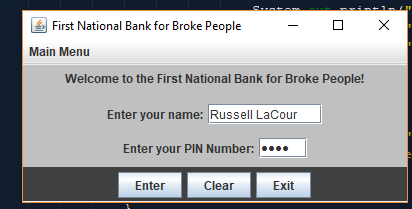
For my project, I had to create a bank management system that is written in Java. The first thing I did was create an abstract class for the bank account’s basic functions, and a checking account class and savings account class that are derived from the bank account class. The base class contains a protected integer variable called numberOfAccounts, which gives an account number to the checking and savings account. The Bank Account class contains a constructor that takes a String called “Name” and a double variable called “amount” and assigns an account number to that account.

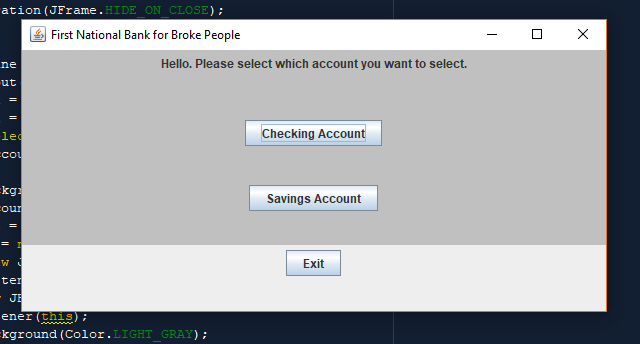
Also inside the Bank Account class are methods that calculate the deposit and withdraw functions.

The CheckingAccount class contains a constructor that calls the super constructor in the base class, and appends them into its own constructor. It also contains a static double variable called fee, where it is equal to 0.15. The class has a withdraw method that overrides the base class.

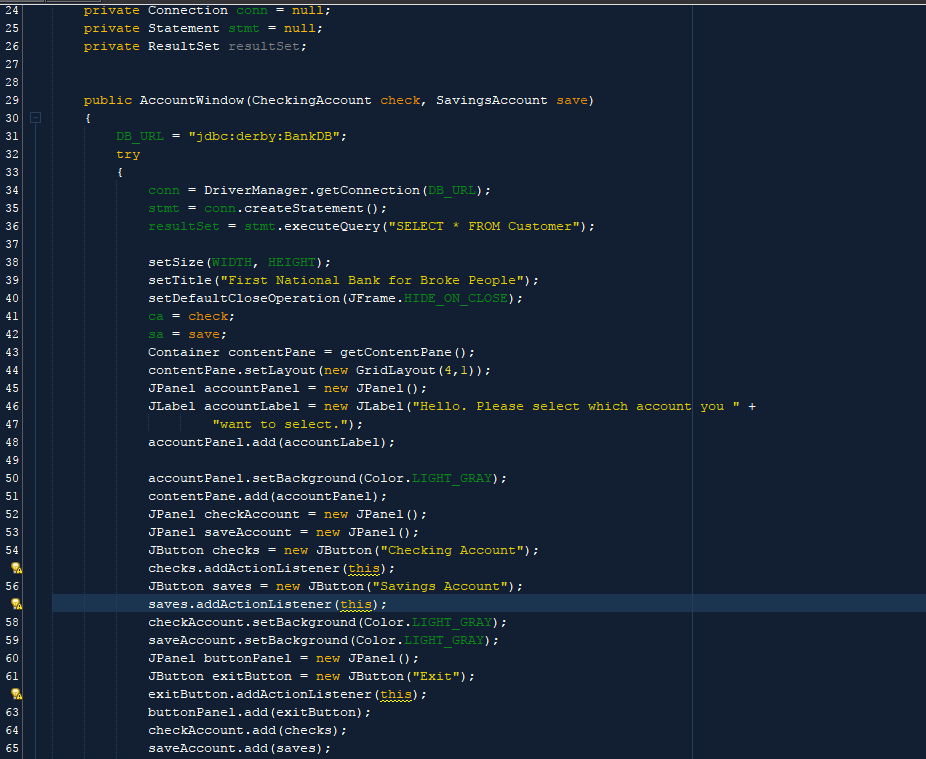
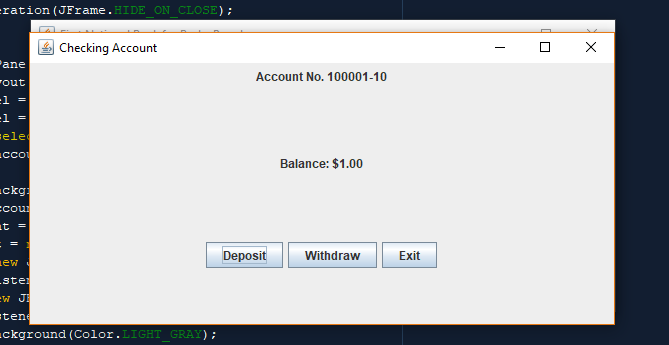
After creating the bank account classes, the next thing that I created was the Bank’s database. The database was created using Derby and is embedded into the Java project. The database contains a table called “Customer”, where it stores their name, PIN number, address, checking balance and savings balance. The customer table contains 5 clients, including myself.

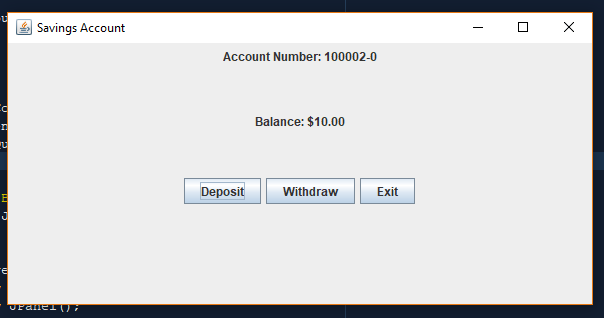
After the database was created, I went into my main class, BankManagementSystem, and changed some of the code around to implement the database. When you first run the program, a window would pop up that will ask the user to type in their name and PIN number.

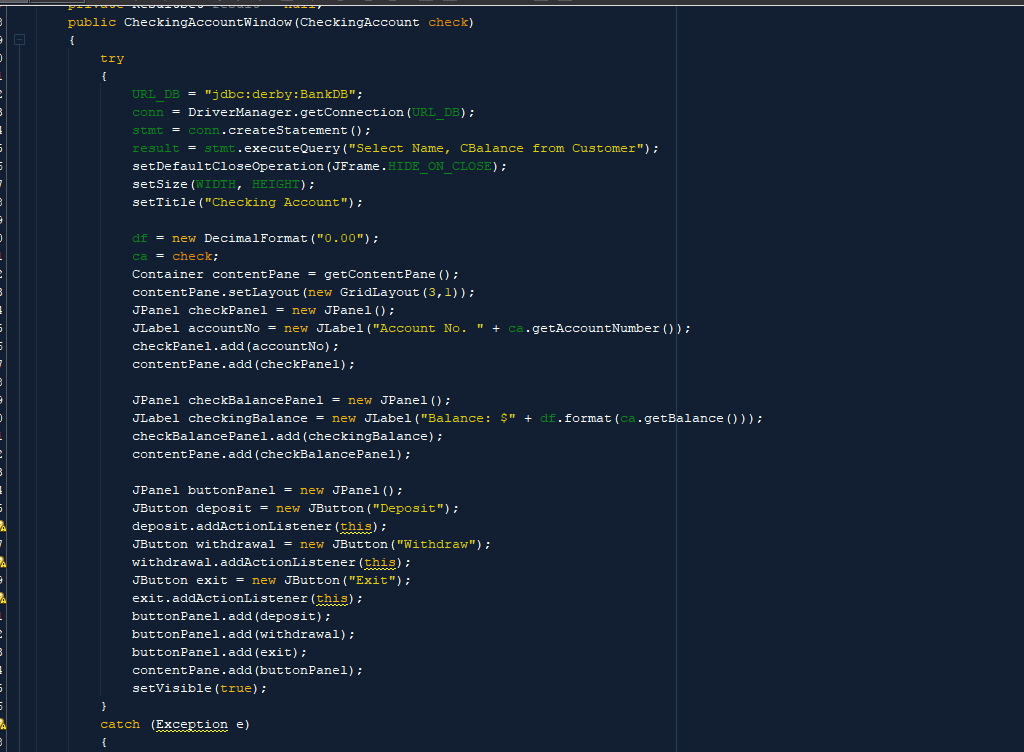


If the user’s name and PIN number matched with anyone’s in the database, they would be sent to the AccountWindow GUI. However, when I first tested the program, it occurred to me that it would open the account window regardless of what the user typed. To resolve this issue, I added an if statement where if the user and pin text fields are not null, then the ResultSet would look through the database, and if the input matched with the customer’s name and PIN, they would be able to access the account window. The class also creates and maps the checking and savings account to the database.

The account window class takes the checking and savings accounts as an argument. From there, the user can decide whether they want to access their checking account or savings account. Each user starts out with a balance of $1.00 in their checking, and a balance of $10.00 in their savings.



The checking account window and savings account window classes take their respective accounts as an argument, so the user can deposit and withdraw whatever amount they want, as long as it doesn’t go beyond their current balance. Both classes contain two private inner classes for the deposit and withdraw windows and functions.



One of the problems I encountered was when I was coding the deposit inner class, it wouldn’t take the user’s input and store it into the account. At first, I tried to create another method that would take the string and convert it into a double, add it onto the current balance. However, it created some problems with the algorithm. But I later found out that the method was suffice enough to add the amount. The same thing was true for the withdraw inner class.

