Documentation for Program

David G. Fields

Seth Riley-Nelson

Henry Onesengmani

Professor: Ms. Forde

Project # 1 Customer

CSC 225-800 Advanced JAVA Programming

Summer 2014

Table of Contents

[Program Abstract 2](#_Toc369196268)

[Program Design 2](#_Toc369196269)

[UML Diagram 2](#_Toc369196270)

[Source Code 3](#_Toc369196271)

[**Input** 11](#_Toc369196272)

[**Assigned Test Data** 11](#_Toc369196273)

[**Output** 14](#_Toc369196274)

[**Assigned Test Run** 14](#_Toc369196275)

# Program Abstract

This program will be used with the BankTestDriver program to manage the accounts of customers for ES&L Banking.

# Program Design

## UML Diagram

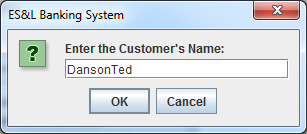
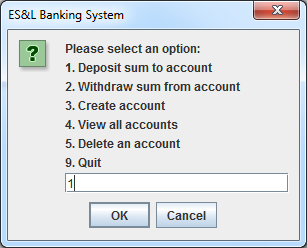
|  |
| --- |
| Customer |
| ~name  ~idNumber  ~balance  ~phoneNumber  **- FEE 1.50** |
| +Customer()  +Customer(name: String, idNumber: String, balance: double, phoneNumber: long)  +withdraw(amount:double):double  +deposit(amount:double):double  +addInterest():double  +getBalance():double  +setBalance(balance:double):void  +getPhoneNumber():String  +setPhoneNumber(phoneNumber:String):void  +getIdNumber():String  +setIdNumber(idNumber:String):void  +getName():String  +setName(name:String):void  +deleteCustomer(Customer[] custArray, int Index)  +format(double):void  +nameSort(Customer[] custArray, int count):void  +findIndex(Customer[] custArray, String name, int count):int  +toString(name, idNumber, balance, phonenumber):String  +addNewCustomer():void  +toString():String |

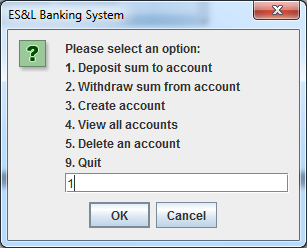
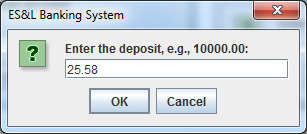
# Source Code

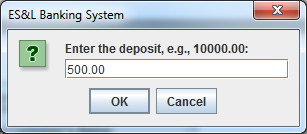
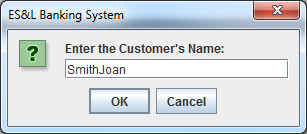
/\*\*Program CSC 225 Prog   
\*Course Title: Advanced JAVA Programming  
\*Course Number: CSC 225-800  
\*Instructor: Ms. Christine Forde  
\*@authors David Fields, Seth Riley-Nelson, Henry Onesengmani  
\*@version 1.0, 9/24/2013  
\*  
\*Description: Program CSC 225 Prog 1 Customer  
\*Will be used with the Bank Test Driver program to manage the accounts  
\*of ES&L Bank customers.  
\*  
\*Input:We will input a deposit or withdrawal amount. We will have the  
\*ability to create or delete an account.  
\*  
\*Compute:The program will compute banking fees,and interest earned   
\* on different transactions.   
\*  
\*  
\*@authors David Fields, Seth Riley-Nelson, Henry Onesengmani  
\*@version 1.0, 06/18/2014  
\*/  
  
import javax.swing.\*;  
import java.text.\*;  
import java.io.\*;  
import java.util.\*;  
  
  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Due Date:06/18/2014<p>  
Program Description: Customer - Service class to be used with the client  
class BankTestDriver, to manage the accounts of ES&L Bank customers.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
public class Customer  
{   
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Class Variables   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
 private String name;  
 private String idNumber;  
 private double balance;  
 private String phoneNumber;   
 final double FEE = 1.50;   
 DecimalFormat fmt2 = new DecimalFormat ("$0.00");  
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Default Constructor  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
 /\*  
 \* Initializes instance variables for the default constructor.  
 \*/  
  
 public Customer()  
 {  
 name = null;  
 idNumber = null;  
 balance = 0.0;  
 phoneNumber = null;  
 }  
   
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Non-Default Constructor  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
 /\*  
 \* Accepts a value for the class variables.  
 \*/  
   
 public Customer(String name,  
 String idNumber,  
 double balance,  
 String phoneNumber)  
 {  
 this.name = name;  
 this.idNumber = idNumber;  
 this.balance = balance;  
 this.phoneNumber = phoneNumber;  
 }  
   
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 // Instance Methods  
 //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
 /\*  
 \* The following six methods are the accessor and mutator  
 \* methods of the Customer class  
 \*/  
   
   
 public String getName()  
 {  
 return name;  
 }  
   
 public void setName(String name)  
 {   
 this.name = name;  
 }  
   
   
   
 public String getIdNumber()  
 {  
 return idNumber;  
 }  
   
 public void setIdNumber(String inIdNumber)   
 {   
 this.idNumber = inIdNumber;  
 }  
  
   
   
 public double getBalance()  
 {  
 return balance;  
 }  
   
 public void setBalance(double inbalance)  
 {   
 this.balance = inbalance;  
 }  
  
  
   
 public String getPhoneNumber()  
 {  
 return phoneNumber;  
 }  
   
 public void setPhoneNumber(String InPhoneNumber)   
 {   
 this.phoneNumber = InPhoneNumber;  
 }  
   
   
   
 /\*  
 \* The following is the withdraw method used to process  
 \* a customer request  
 \*/  
   
   
 public double withdraw(double amount)   
 {   
 if(amount < 0)  
 {  
 JOptionPane.showMessageDialog(null,"Error: Withdraw amount is invalid."  
 + "\nCustomer: " + name + "\nRequested " + fmt2.format(amount + FEE),  
 "ES&L Banking System",JOptionPane.ERROR\_MESSAGE);   
 }  
   
   
 if(amount > balance)  
 {  
 JOptionPane.showMessageDialog(null,"Error: Insufficient funds"  
 + "\nCustomer: " + name + "\nRequested: " + fmt2.format(amount + FEE)  
 + "\nAvailable: " + fmt2.format(balance),  
 "ES&L Banking System",JOptionPane.ERROR\_MESSAGE);   
 }   
 else  
 {  
 balance -= (amount + FEE);   
 }  
   
 return balance;  
 }//end of withdraw method  
   
   
   
 /\*  
 \* The following is the deposit method used to process  
 \* a customer request  
 \*/  
  
   
   
 public double deposit(double amount)   
 {   
   
 if(amount < 0)  
 {  
 JOptionPane.showMessageDialog(null,"Error: Deposit amount is invalid."  
 + "\nCustomer: " + name + "\nRequested " + amount,  
 "ES&L Banking System",JOptionPane.ERROR\_MESSAGE);   
 }   
   
   
 else  
 {  
 balance = ((balance + amount));   
 }  
   
 return balance;  
 }//end of deposit method  
   
   
   
 /\*  
 \* The following is the addInterest method used to add interest to  
 \* a customer account  
 \*/   
   
 public double addInterest()  
 {  
 double interest = balance \* .045;   
 balance += interest;  
 return balance;  
   
 }//end addInterest  
   
   
 /\*  
 \* The following is the addNewCustomer method used to process  
 \* a customer request  
 \*/   
   
 public void addNewCustomer (Customer []custArray,  
 String name,   
 int count,   
 String idnumber,   
 double balance,String phoneNumber)  
 throws IllegalArgumentException,  
 ArrayIndexOutOfBoundsException  
 {   
 custArray[ count ] = new Customer (name, idnumber,balance, phoneNumber);  
 }  
   
  
 //---------------------------------------------------------  
 // Sort the custsArray array in alphabetical order by name  
 // using select sort  
 //---------------------------------------------------------  
 public static void nameSort(Customer [] custsArray, int count)  
 {  
 for (int i = 0; i < count - 1; i++)  
 {  
 boolean exchange = false;  
 int smallPos = i;  
  
 for (int j = i+1; j < count; j++)  
 if (custsArray[smallPos].getName().compareTo(custsArray[j].getName())  
 > 0)  
 {  
 smallPos = j;  
 exchange = true;  
 }  
  
 //switch smallest to ith location  
 if (exchange)  
 {  
 Customer temp = custsArray[i];  
 custsArray[i] = custsArray[smallPos];  
 custsArray[smallPos] = temp;  
 exchange = false;  
 }  
  
 }//end for i loop  
  
 } //end method nameSort  
  
 public int findIndex (Customer [] custsArray,String xnam, int count)  
 {  
   
 for (int index = 0; index < count;index++) //search the entire  
 //table until a match is found  
 {  
   
 if (custsArray[index].getName().compareTo(xnam) == 0)  
 return index;  
 } //end index < custsArray count  
 //return a -1 to imply a matching name- name not found!  
 return -1;  
 }// end for givenNameGetCustomer  
   
 public int deleteCustomer(Customer[] custsArray, int index, int count, String xnam) {  
   
 if (index != -1)  
 {  
 if (count >= 1 && count <= custsArray.length)  
 {  
 custsArray[index] = custsArray[count-1];   
 count--; //decrement count now that we have one less  
 //element  
  
 JOptionPane.showMessageDialog  
 (null, xnam + " is deleted. ",  
 "ES&L Banking System",  
 JOptionPane.INFORMATION\_MESSAGE);  
 } // end nested if statement  
  
 } // end if stmt  
  
 else  
 {  
 JOptionPane.showMessageDialog  
 (null, xnam + " was not found ",  
 "ES&L Banking System",  
 JOptionPane.INFORMATION\_MESSAGE);  
  
 } //end else stmt  
   
 return count;  
  
 }  
  
 //-----------------------------------------------------------------  
 // Returns a one-line description of the customer as a string.  
 //-----------------------------------------------------------------  
 public String toString ()  
 {  
 return (name + "\t" + idNumber + "\t" + fmt2.format(balance) + "\t" + phoneNumber);  
 }  
   
}//End customer class

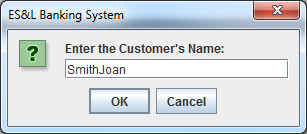
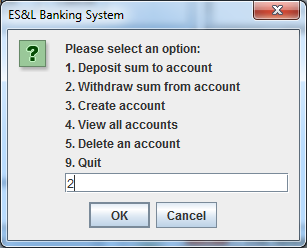
**Input**

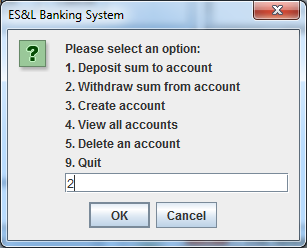
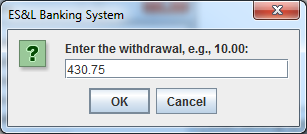
**Assigned Test Data**

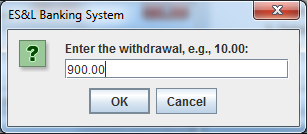
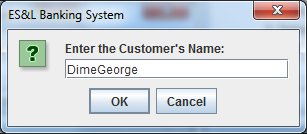


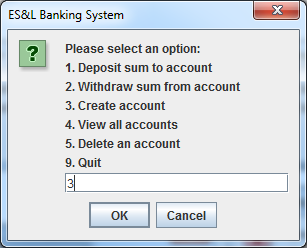
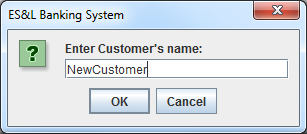
Ï

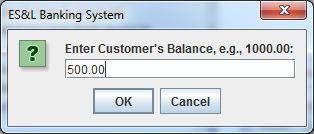
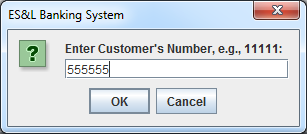


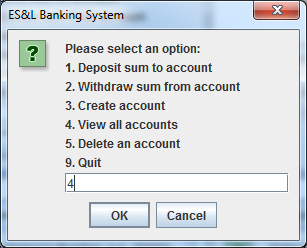
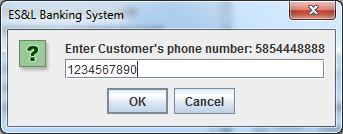


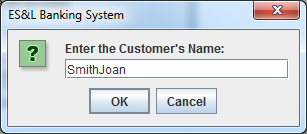
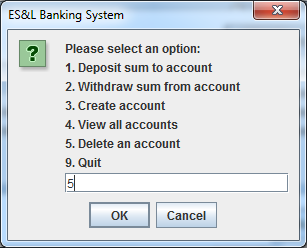




ÏÏ







**Output**

**Assigned Test Run**

