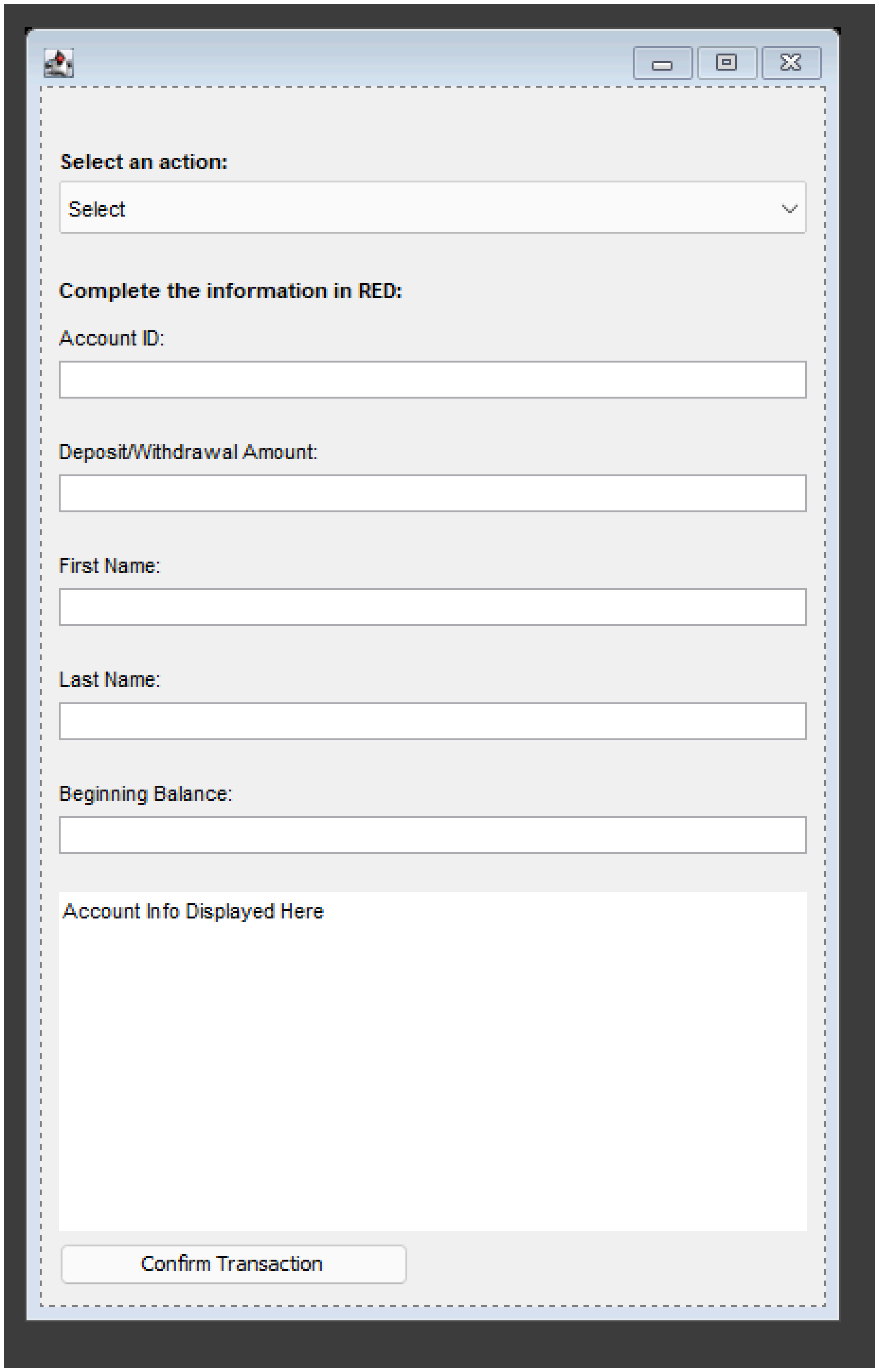
Credit Name: CSE 3010 - Computer Science 3

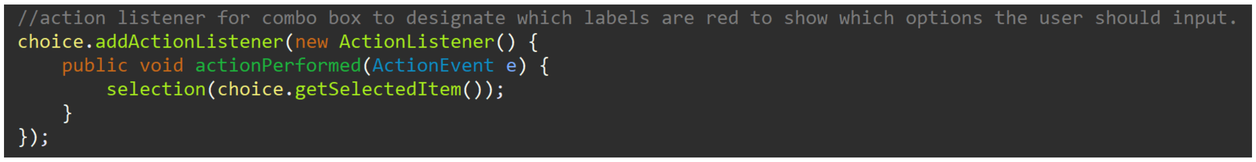
Assignment: LocalBankGUI

How has your program changed from planning to coding to now? Please Explain



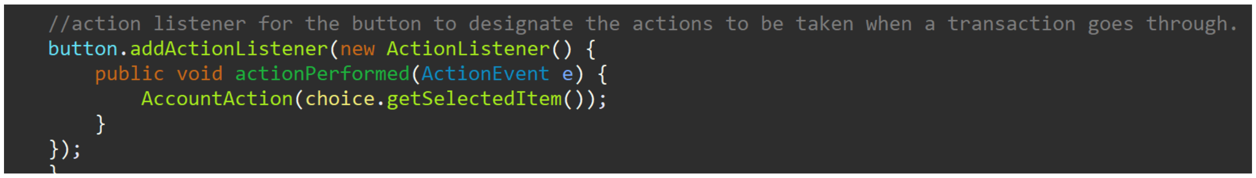
I started by setting up the components in the graphical user interface.

I have a combo box (labelled choice) for the account action the user takes, with each option inputted in the combo box. Then I have labels which tell the user the information to fill in each box and an associated JTextField for the information to be inputted and a JTextArea for the information to be outputted(the account info and confirmation information). Lastly there is a confirm Transaction button which allows the user to start each action/transaction or end the application.



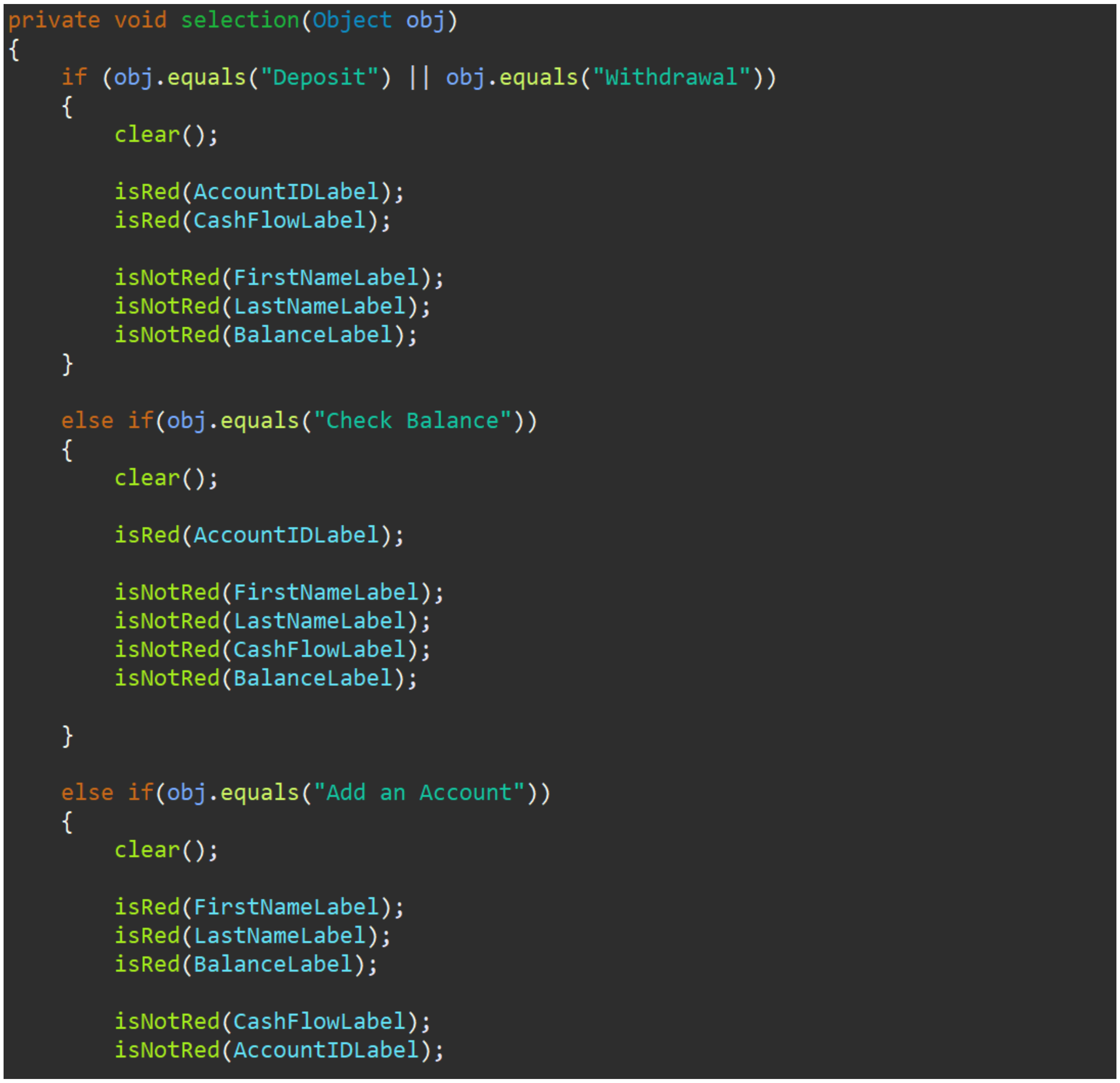
```
//action listener for combo box to designate which labels are red to show which options the user should input.
choice.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        selection(choice.getSelectedItem());
    }
});
```

There is an action listener for the combo box selections which changes text colour to red of only the relevant labels where the user needs to type information. This is done with a method called selection.

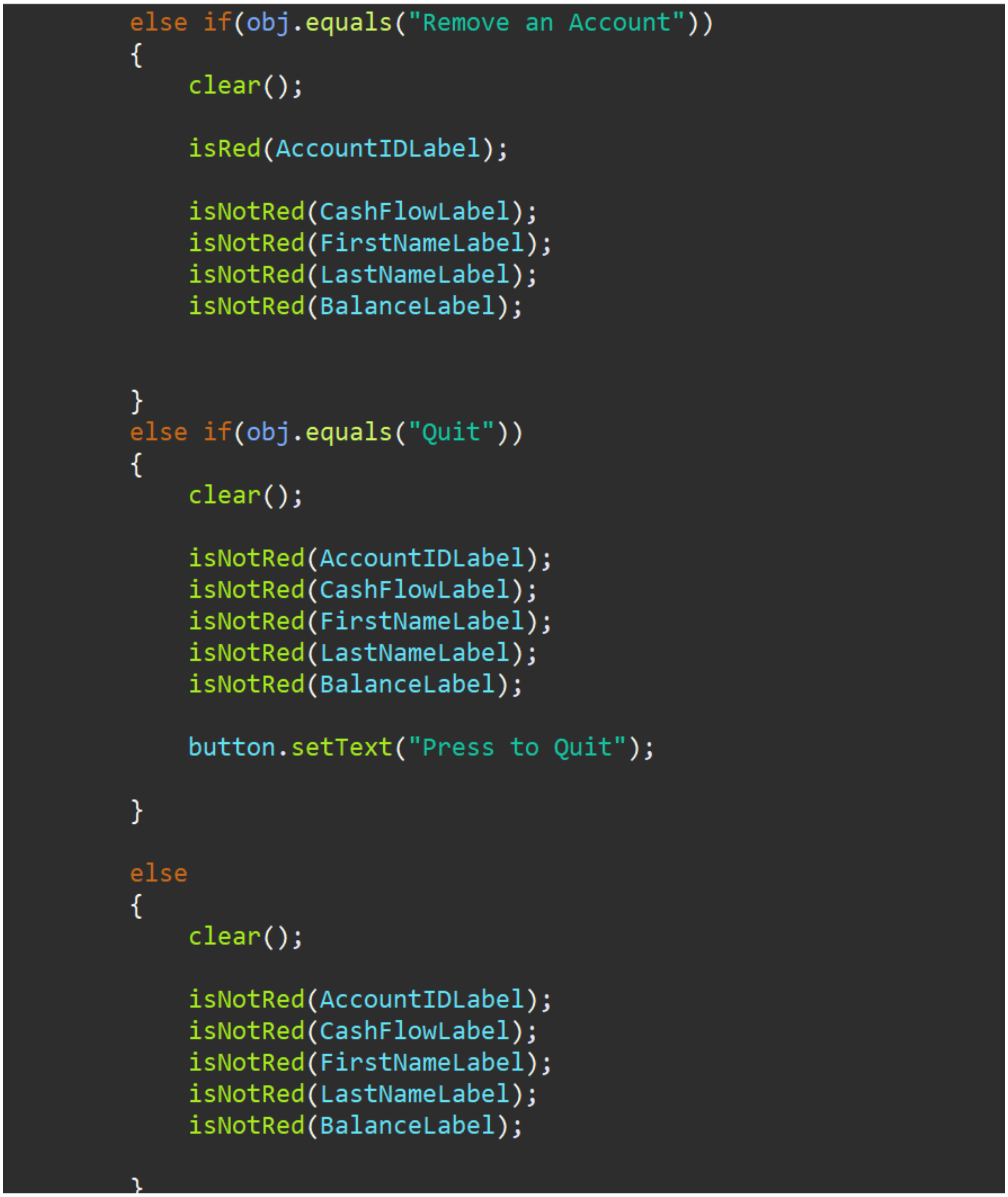


```
    //action listener for the button to designate the actions to be taken when a transaction goes through.
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            AccountAction(choice.getSelectedItem());
        }
    });
    }
```

Our JButton also has an action listener attached which tells the program what action to actually completes. Its what initiates the action which adds or removes accounts or funds from the account.

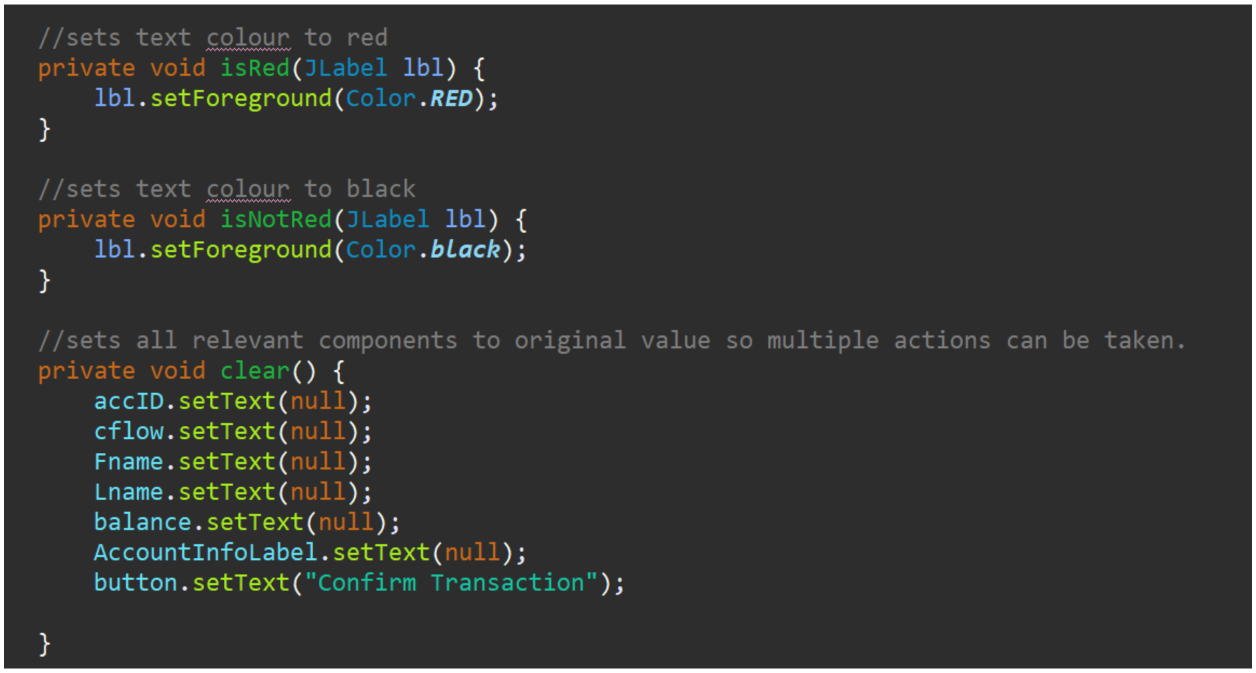


```java
private void selection(Object obj)
{
    if (obj.equals("Deposit") || obj.equals("Withdrawal"))
    {
        clear();

        isRed(AccountIDLabel);
        isRed(CashFlowLabel);

        isNotRed(FirstNameLabel);
        isNotRed(LastNameLabel);
        isNotRed(BalanceLabel);
    }

    else if(obj.equals("Check Balance"))
    {
        clear();

        isRed(AccountIDLabel);

        isNotRed(FirstNameLabel);
        isNotRed(LastNameLabel);
        isNotRed(CashFlowLabel);
        isNotRed(BalanceLabel);

    }

    else if(obj.equals("Add an Account"))
    {
        clear();

        isRed(FirstNameLabel);
        isRed(LastNameLabel);
        isRed(BalanceLabel);

        isNotRed(CashFlowLabel);
        isNotRed(AccountIDLabel);
```



```
        else if(obj.equals("Remove an Account"))
        {
            clear();

            isRed(AccountIDLabel);

            isNotRed(CashFlowLabel);
            isNotRed(FirstNameLabel);
            isNotRed(LastNameLabel);
            isNotRed(BalanceLabel);


        }
        else if(obj.equals("Quit"))
        {
            clear();

            isNotRed(AccountIDLabel);
            isNotRed(CashFlowLabel);
            isNotRed(FirstNameLabel);
            isNotRed(LastNameLabel);
            isNotRed(BalanceLabel);

            button.setText("Press to Quit");

        }

        else
        {
            clear();

            isNotRed(AccountIDLabel);
            isNotRed(CashFlowLabel);
            isNotRed(FirstNameLabel);
            isNotRed(LastNameLabel);
            isNotRed(BalanceLabel);

        }
```

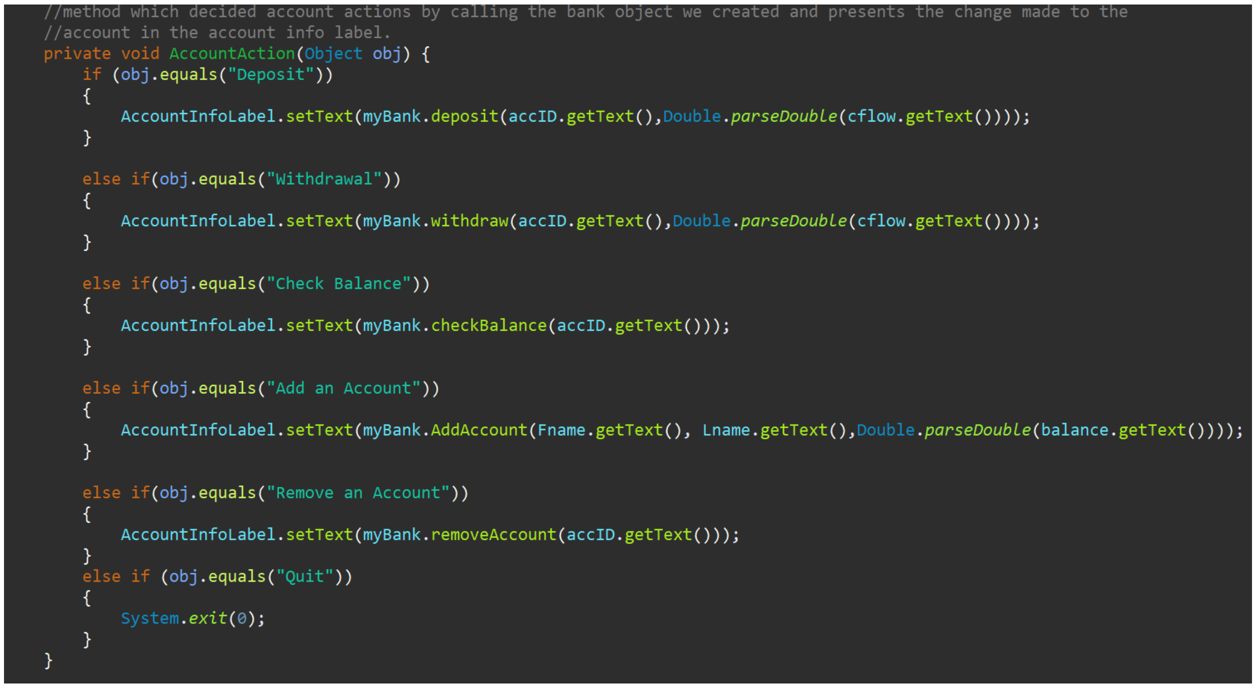
The selection method is a simple one which sets the labels to be red as red and sets all others to blacks while also resetting the text fields and the account info text field.



```java
//sets text colour to red
private void isRed(JLabel lbl) {
    lbl.setForeground(Color.RED);
}

//sets text colour to black
private void isNotRed(JLabel lbl) {
    lbl.setForeground(Color.black);
}

//sets all relevant components to original value so multiple actions can be taken.
private void clear() {
    accID.setText(null);
    cflow.setText(null);
    Fname.setText(null);
    Lname.setText(null);
    balance.setText(null);
    AccountInfoLabel.setText(null);
    button.setText("Confirm Transaction");

}
```

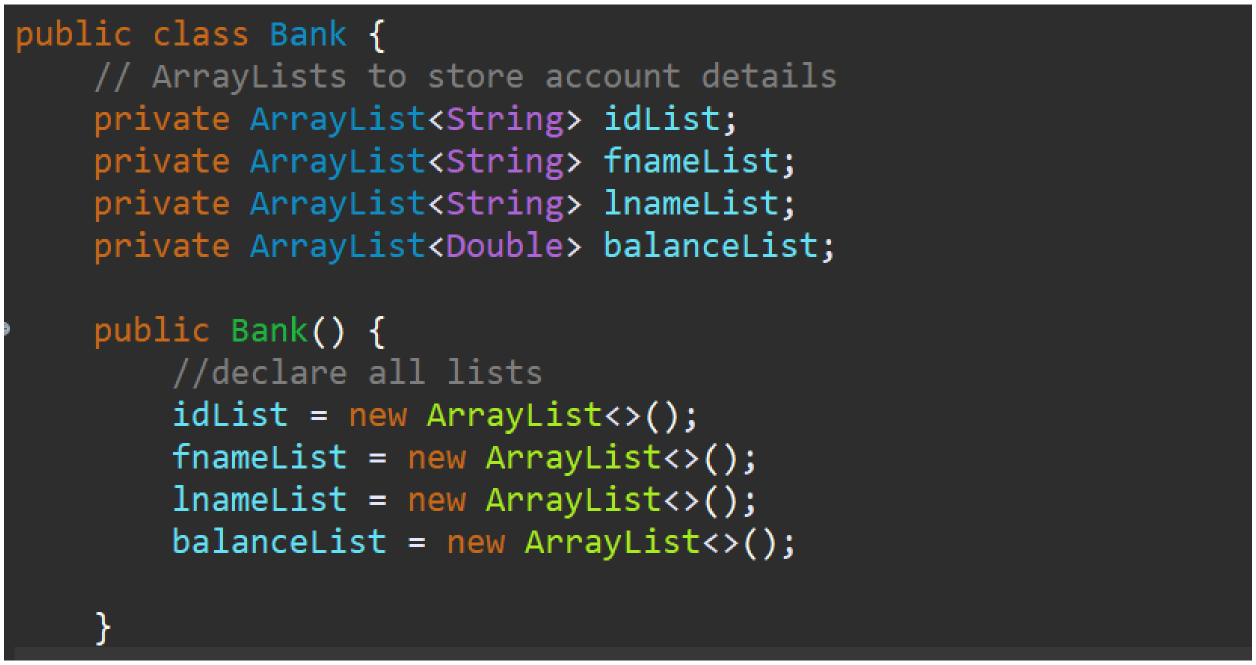
the isRed method works by setting its JLabel parameter's foreground to red
The isNotRed works by setting it to black instead.

The clear() method sets all text fields and account info to null as well as the button's text to confirm transaction.



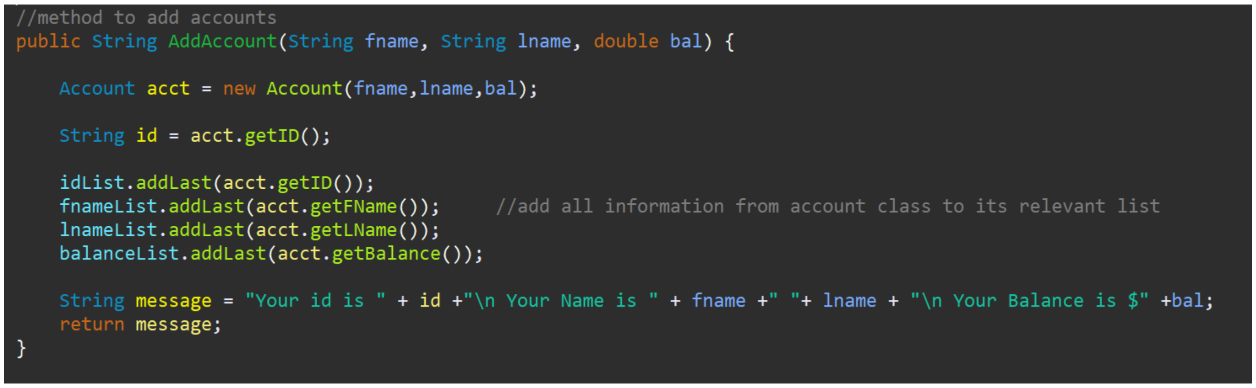
```java
//method which decided account actions by calling the bank object we created and presents the change made to the
//account in the account info label.
private void AccountAction(Object obj) {
    if (obj.equals("Deposit"))
    {
        AccountInfoLabel.setText(myBank.deposit(accID.getText(),Double.parseDouble(cflow.getText())));
    }

    else if(obj.equals("Withdrawal"))
    {
        AccountInfoLabel.setText(myBank.withdraw(accID.getText(),Double.parseDouble(cflow.getText())));
    }

    else if(obj.equals("Check Balance"))
    {
        AccountInfoLabel.setText(myBank.checkBalance(accID.getText()));
    }

    else if(obj.equals("Add an Account"))
    {
        AccountInfoLabel.setText(myBank.AddAccount(Fname.getText(), Lname.getText(),Double.parseDouble(balance.getText())));
    }

    else if(obj.equals("Remove an Account"))
    {
        AccountInfoLabel.setText(myBank.removeAccount(accID.getText()));
    }
    else if (obj.equals("Quit"))
    {
        System.exit(0);
    }
}
```

The account action method works based on the selected option in the JLabel(which is its parameter) and uses our Bank object class to perform the various actions and indicate them on our AccountInfoLabel.



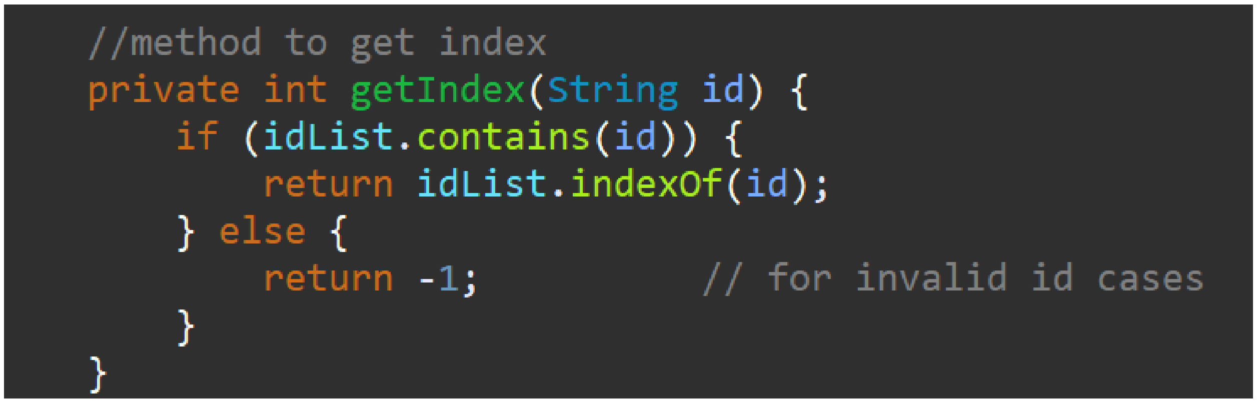
```java
public class Bank {
    // ArrayLists to store account details
    private ArrayList<String> idList;
    private ArrayList<String> fnameList;
    private ArrayList<String> lnameList;
    private ArrayList<Double> balanceList;

    public Bank() {
        //declare all lists
        idList = new ArrayList<>();
        fnameList = new ArrayList<>();
        lnameList = new ArrayList<>();
        balanceList = new ArrayList<>();

    }
```

In our bank class we have 4 ArrayLists for our bank information and they are all instance variables because we will call them in other methods in this class.



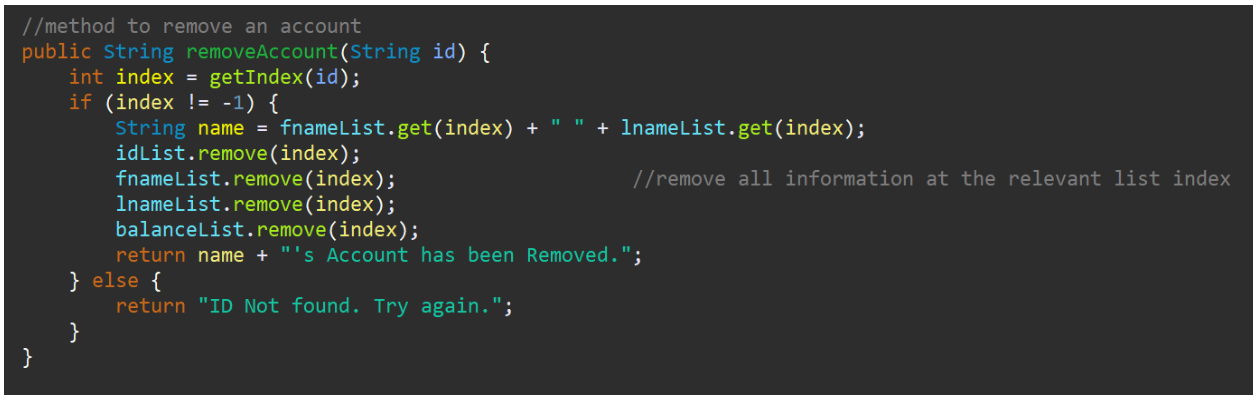
```java
//method to add accounts
public String AddAccount(String fname, String lname, double bal) {

    Account acct = new Account(fname,lname,bal);

    String id = acct.getID();

    idList.addLast(acct.getID());
    fnameList.addLast(acct.getFName());     //add all information from account class to its relevant list
    lnameList.addLast(acct.getLName());
    balanceList.addLast(acct.getBalance());

    String message = "Your id is " + id +"\n Your Name is " + fname +" "+ lname + "\n Your Balance is $" +bal;
    return message;
}
```

This Bank class relies on another class called Account and that object is used in this method AddAccount. Based on the information given to the AddAccount's parameters an account is made with those same values. The Account class makes an id which, in addition to the other information we add to its list. Since this is all done at once for a single account, every component of an individuals information has the same index in the list.
Lastly a message is returned which indicates the users bank info.



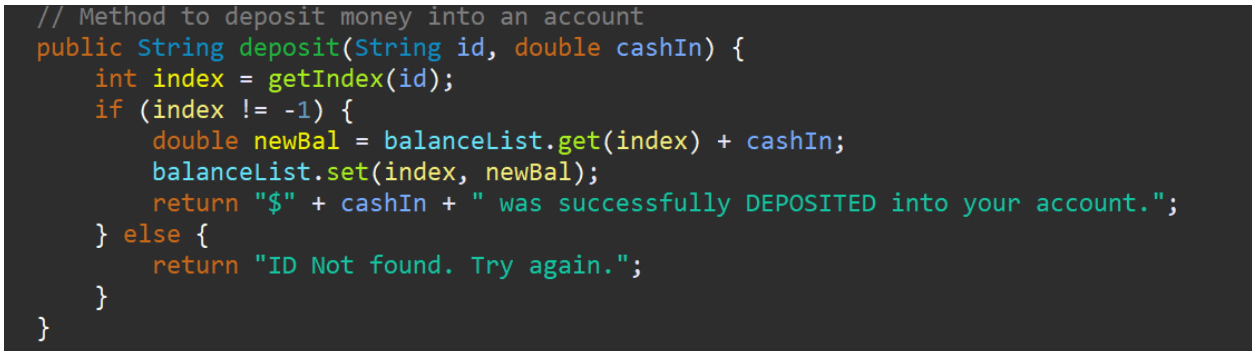
```java
    //method to get index
    private int getIndex(String id) {
        if (idList.contains(id)) {
            return idList.indexOf(id);
        } else {
            return -1;          // for invalid id cases
        }
    }
```

We also have a method to get the index of an id in a list. If the list contains the id then the index of the id is returned otherwise -1 is returned(this no. was chosen as it is not possible for an index to be -1 which differentiates it from the index.)



```java
//method to remove an account
public String removeAccount(String id) {
    int index = getIndex(id);
    if (index != -1) {
        String name = fnameList.get(index) + " " + lnameList.get(index);
        idList.remove(index);
        fnameList.remove(index);                    //remove all information at the relevant list index
        lnameList.remove(index);
        balanceList.remove(index);
        return name + "'s Account has been Removed.";
    } else {
        return "ID Not found. Try again.";
    }
}
```

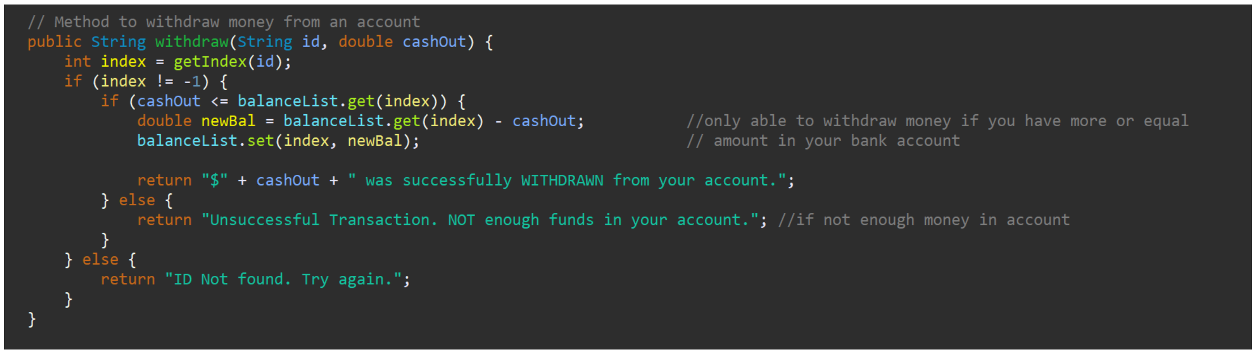
Then to remove an account we get the index of the id and then we remove all information at that index in each ArrayList. But if the ID given in the first place was invalid it is caught because our getIndex method would return -1 which is then incorporated into a if-else statement.



```java
 // Method to deposit money into an account
 public String deposit(String id, double cashIn) {
     int index = getIndex(id);
     if (index != -1) {
         double newBal = balanceList.get(index) + cashIn;
         balanceList.set(index, newBal);
         return "$" + cashIn + " was successfully DEPOSITED into your account.";
     } else {
         return "ID Not found. Try again.";
     }
 }
```

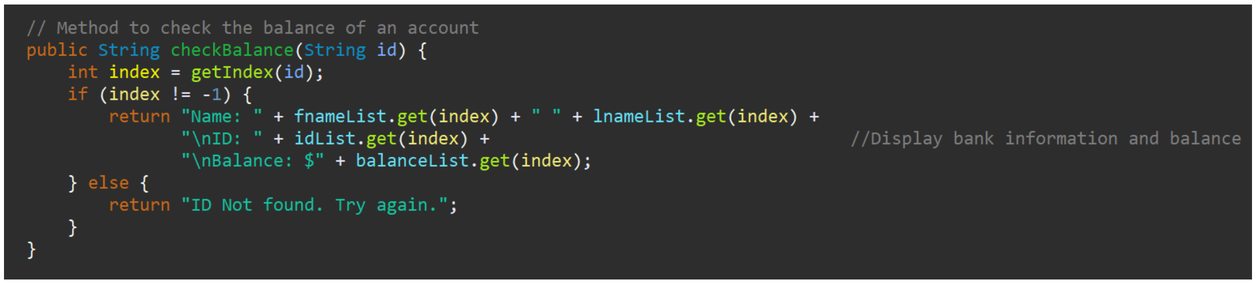
when depositing money with a valid ID, we again get the index of id and use it to retrieve our current bank balance from the ArrayList. Then we add the amount of cash the user indicates to the current balance and replace the current bank balance with the new balance and display a confirmation message.

\



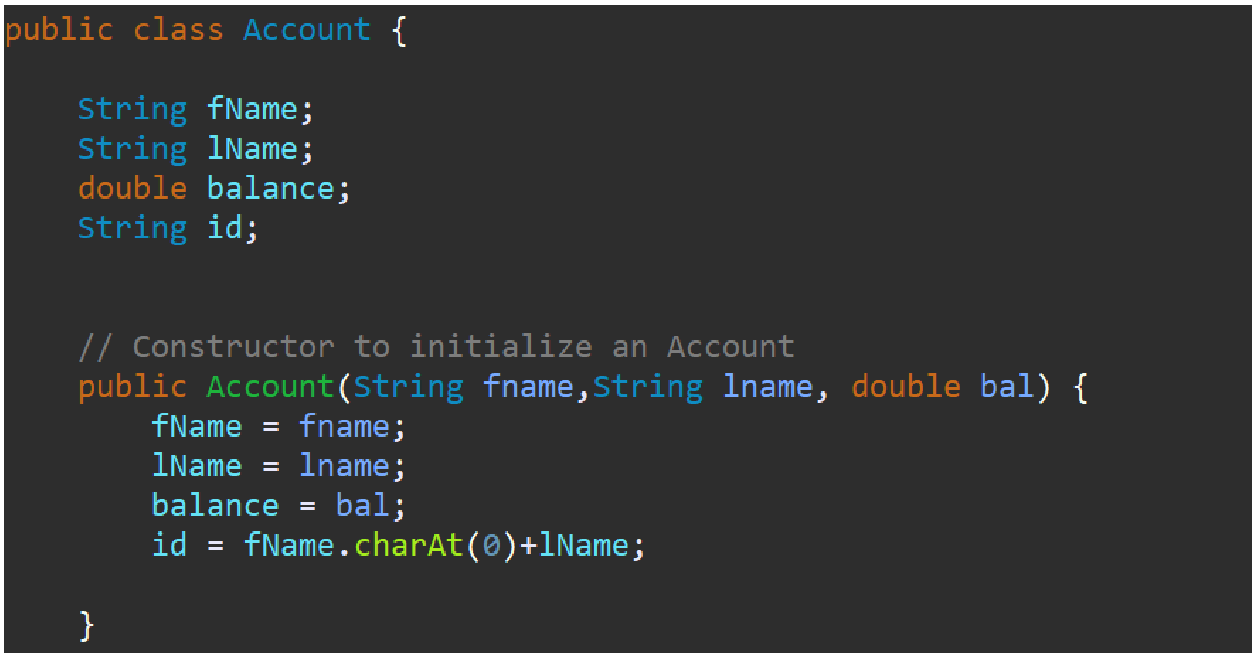
```java
// Method to withdraw money from an account
public String withdraw(String id, double cashOut) {
    int index = getIndex(id);
    if (index != -1) {
        if (cashOut <= balanceList.get(index)) {
            double newBal = balanceList.get(index) - cashOut;           //only able to withdraw money if you have more or equal
            balanceList.set(index, newBal);                              // amount in your bank account

            return "$" + cashOut + " was successfully WITHDRAWN from your account.";
        } else {
            return "Unsuccessful Transaction. NOT enough funds in your account."; //if not enough money in account
        }
    } else {
        return "ID Not found. Try again.";
    }
}
```

In a withdraw method it works the same way but we subtract the amount the user designates instead. Furthermore, we need to ensure the withdrawal amount is always less than the bank balance which is handled with an if-else statement. If withdrawal cash is greater than bank balance, then an invalid statement is returned.



```java
// Method to check the balance of an account
public String checkBalance(String id) {
    int index = getIndex(id);
    if (index != -1) {
        return "Name: " + fnameList.get(index) + " " + lnameList.get(index) +
               "\nID: " + idList.get(index) +                                   //Display bank information and balance
               "\nBalance: $" + balanceList.get(index);
    } else {
        return "ID Not found. Try again.";
    }
}
```

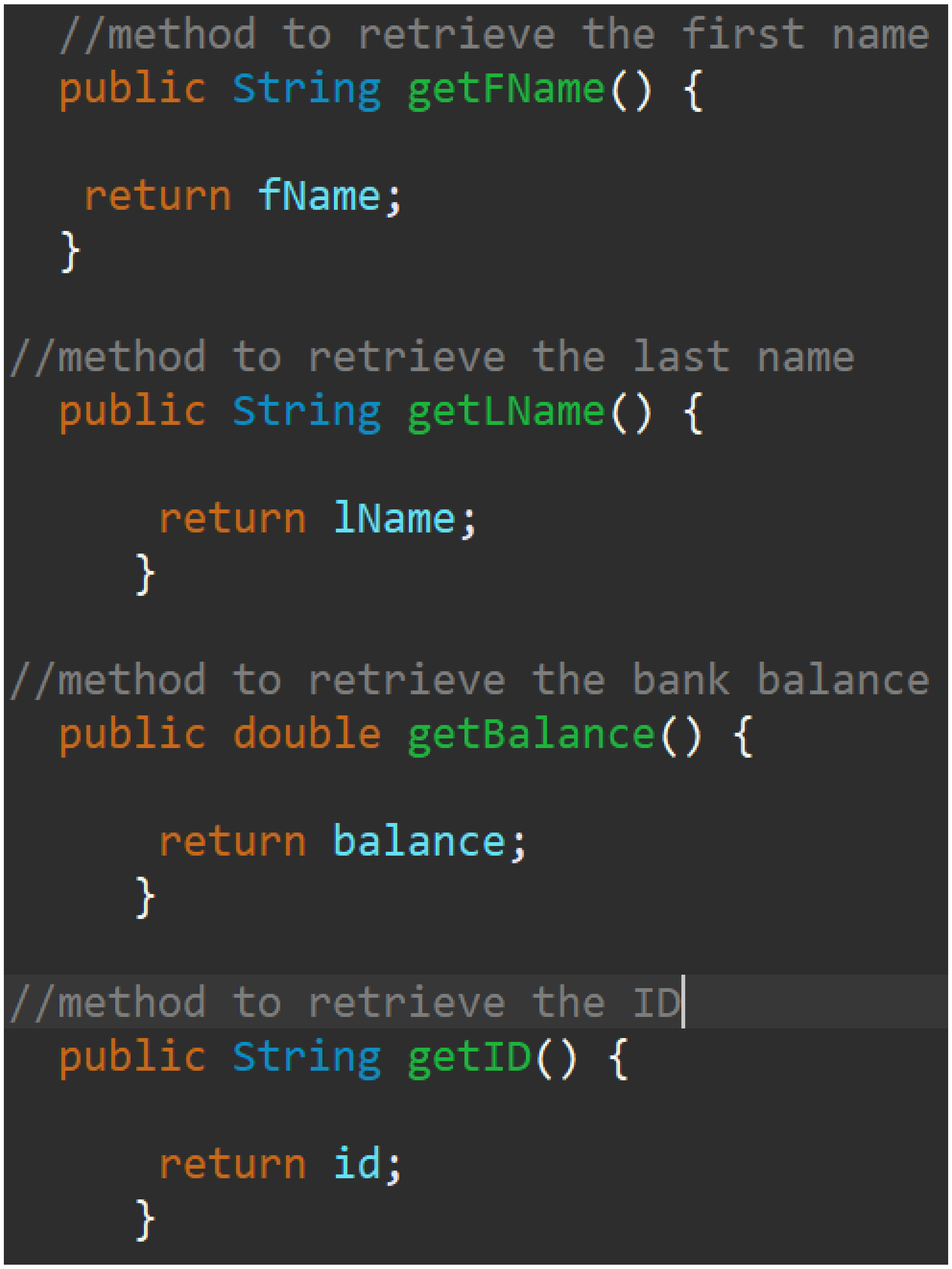
Our last method in bank class is Check Balance. This just gets the index of id in the list and returns all the information in each list at that index.



```java
public class Account {

    String fName;
    String lName;
    double balance;
    String id;


    // Constructor to initialize an Account
    public Account(String fname,String lname, double bal) {
        fName = fname;
        lName = lname;
        balance = bal;
        id = fName.charAt(0)+lName;

    }
```

The account class initializes a bank account with the name, and balance from the Bank class and then forms an ID made by the first character of the first name combined with the last name with no spaces.

The first name,last name , balance and id are all instance variables because i will retrieve them in the methods below.



```
  //method to retrieve the first name
  public String getFName() {

   return fName;
  }

//method to retrieve the last name
  public String getLName() {

      return lName;
    }

//method to retrieve the bank balance
  public double getBalance() {

      return balance;
    }

//method to retrieve the ID
  public String getID() {

      return id;
    }
```

These methods simply return all the information indicated.