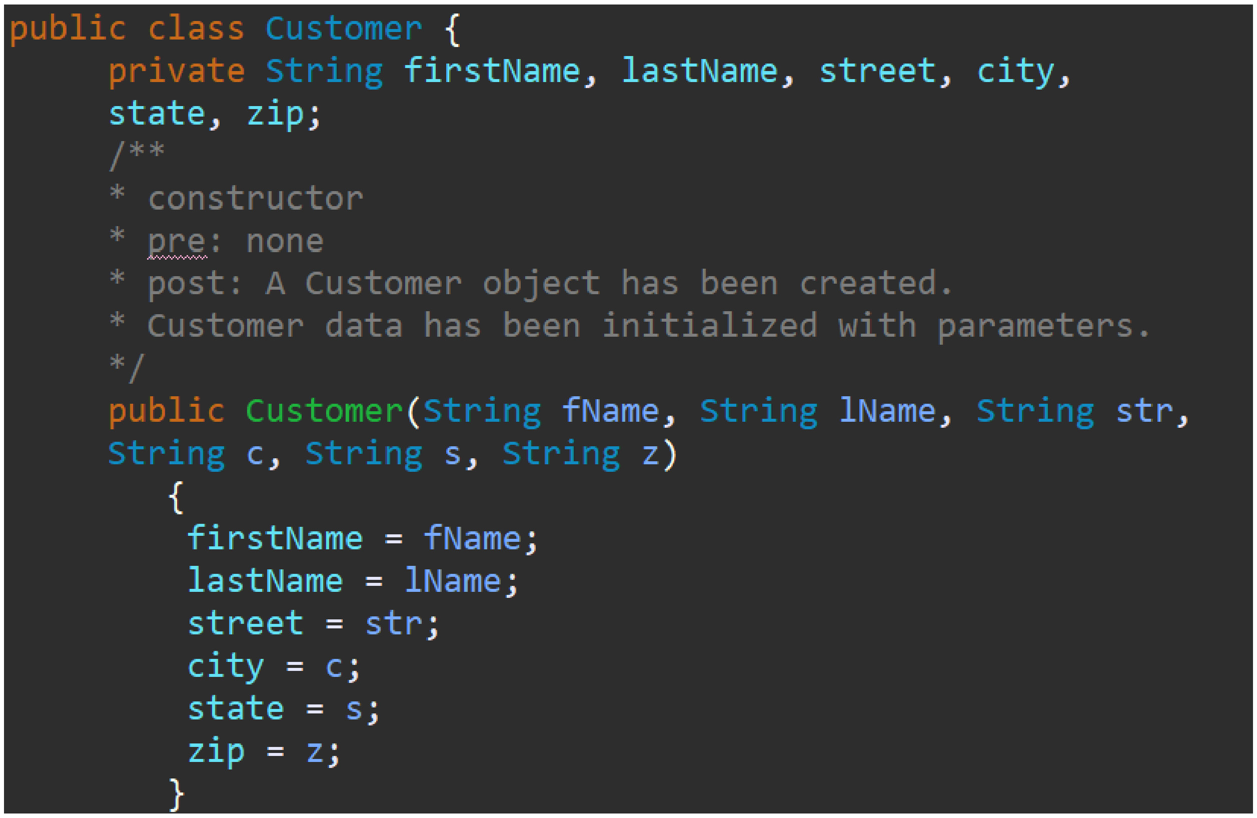
Credit Name: CSE 3130 - Object Oriented Programming 2

Assignment: Bank/PersonalAcct/BusinessAcct.

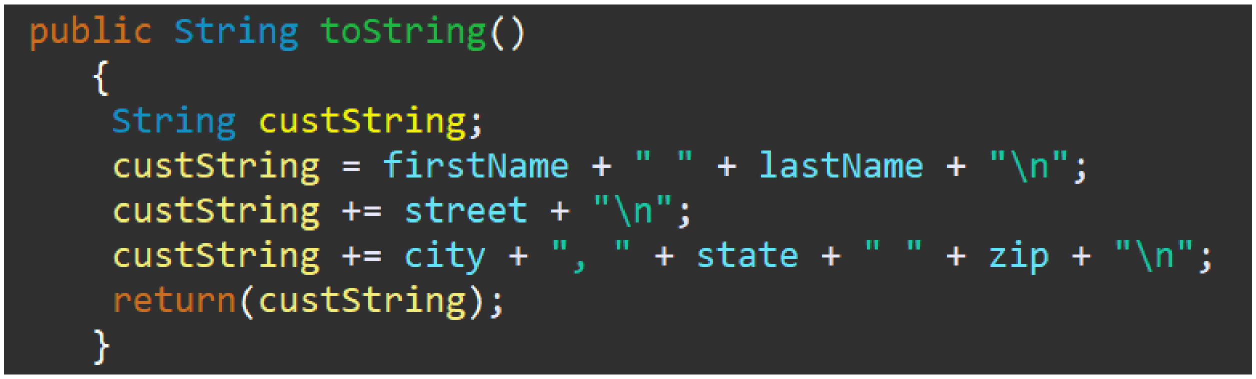
How has your program changed from planning to coding to now? Please Explain

I used the bank account program in chapter 7 as a starting point and formed the Customer class and Account class in the same way.



```
public class Customer {
     private String firstName, lastName, street, city,
     state, zip;
     /**
     * constructor
     * pre: none
     * post: A Customer object has been created.
     * Customer data has been initialized with parameters.
     */
     public Customer(String fName, String lName, String str,
     String c, String s, String z)
        {
         firstName = fName;
         lastName = lName;
         street = str;
         city = c;
         state = s;
         zip = z;
        }
```

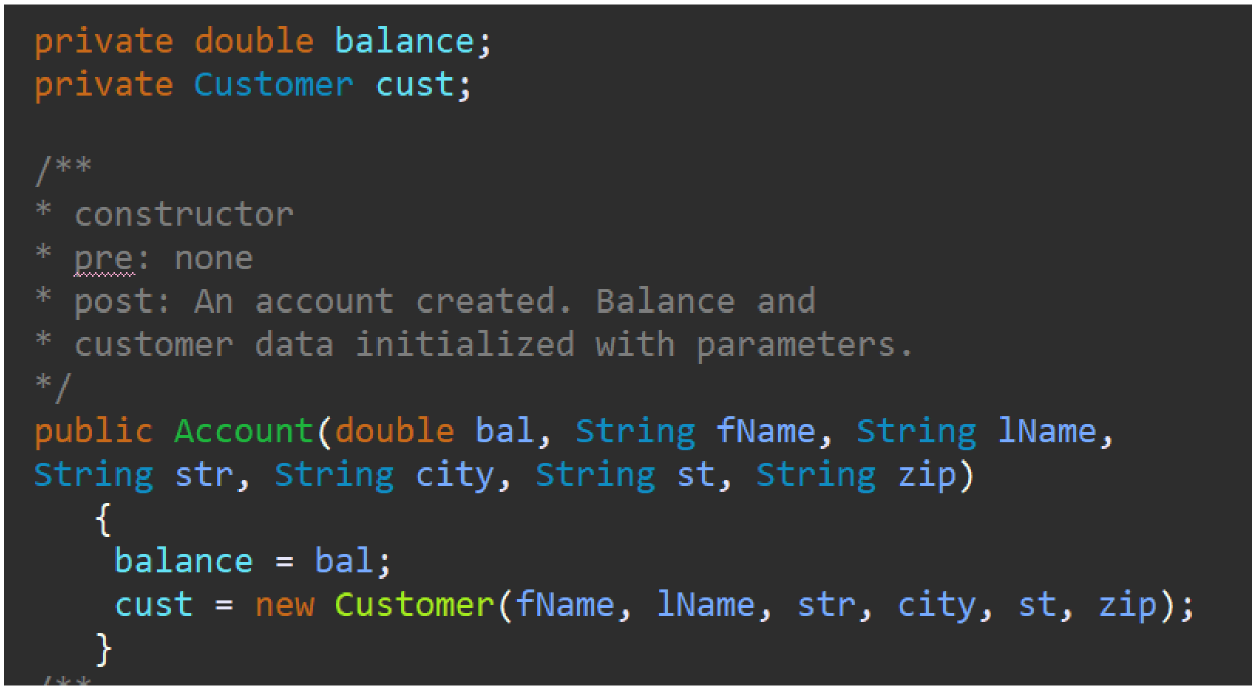
FIrst we made the Customer class. We create class/instance variables for all information to identify a customer, including their name, and the components of their address. Then we form a constructor to create the customer object and initialize each of the declared variables with its respective parameter.



```java
public String toString()
   {
    String custString;
    custString = firstName + " " + lastName + "\n";
    custString += street + "\n";
    custString += city + ", " + state + " " + zip + "\n";
    return(custString);
   }
```

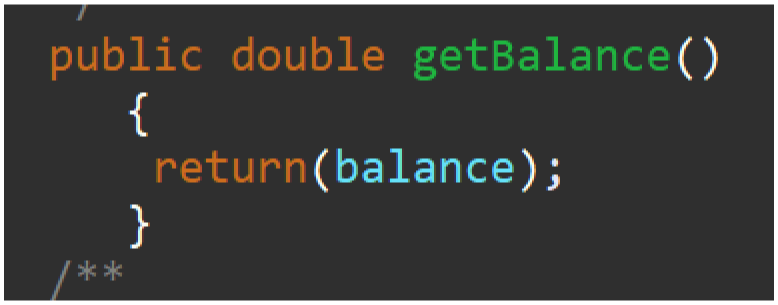
We then have a member method which outputs the customer's details as a string. We use concatenation to create the string.

The next step is creating the Account class



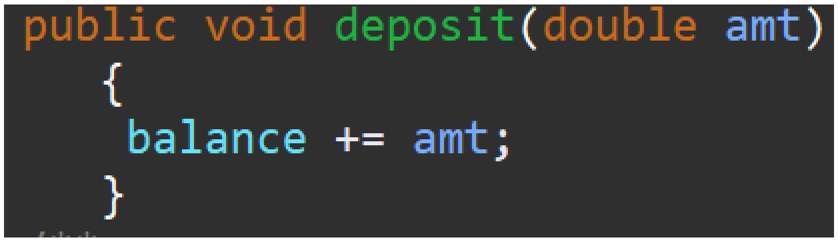
```java
private double balance;
private Customer cust;

/**
* constructor
* pre: none
* post: An account created. Balance and
* customer data initialized with parameters.
*/
public Account(double bal, String fName, String lName,
String str, String city, String st, String zip)
   {
    balance = bal;
    cust = new Customer(fName, lName, str, city, st, zip);
   }
```

Our class variables are balance, which is a double because it refers to money and we also call the customer class we created before; this is because when we make an account, we create a separate file of the information specific to the identity of the customer. Then we form a constructor with the parameters required for both the account and the customer object. Then we initialize both the bank balance and the customer object.



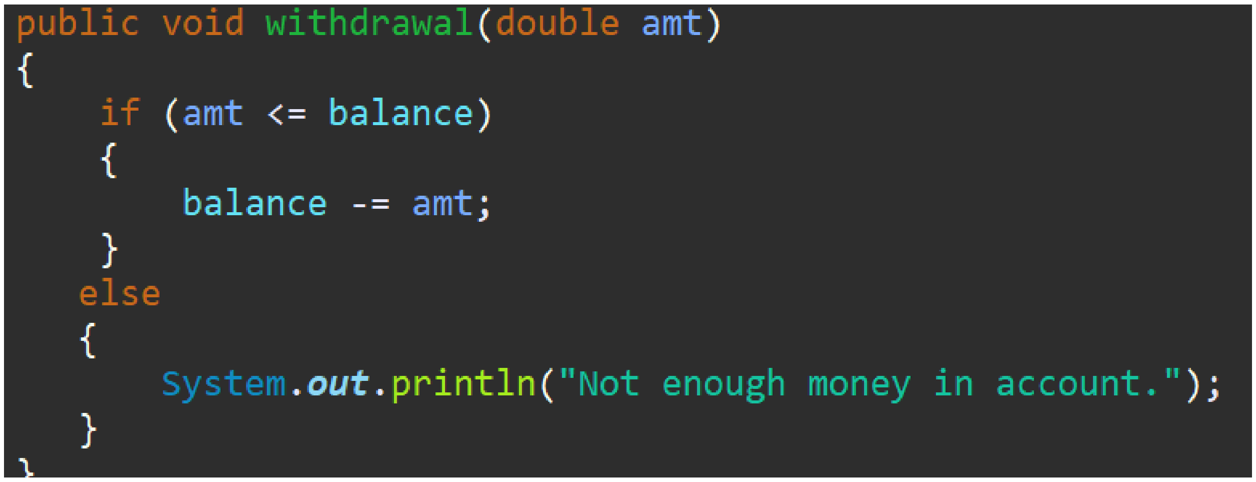
```
 /
public double getBalance()
   {
    return(balance);
   }
 /**
```

We then make a method to return the balance in the account. Our method as such defines the return type to be a double and it can be accessed by other classes so it is public.



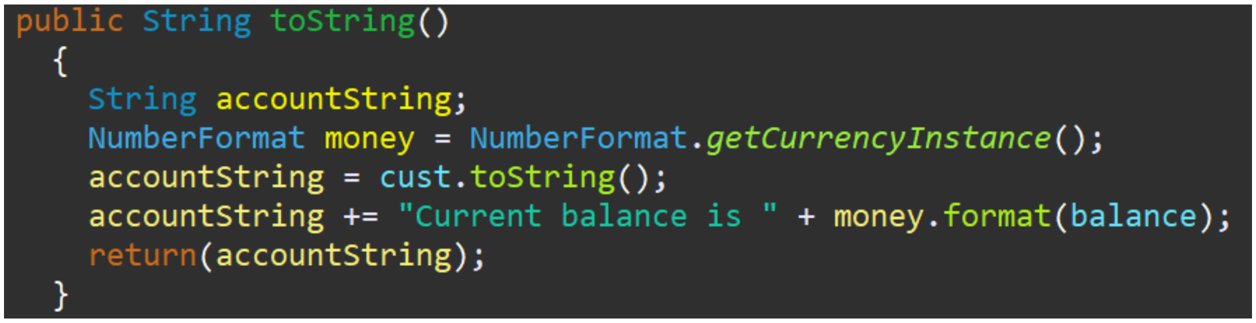
```
public void deposit(double amt)
   {
    balance += amt;
   }
```

Similarly, our next method, deposits/adds money to an account so we add a specified amount(designated in the method parameter) to the existing bank balance.



```
public void withdrawal(double amt)
{
    if (amt <= balance)
    {
        balance -= amt;
    }
   else
   {
       System.out.println("Not enough money in account.");
   }
}
```

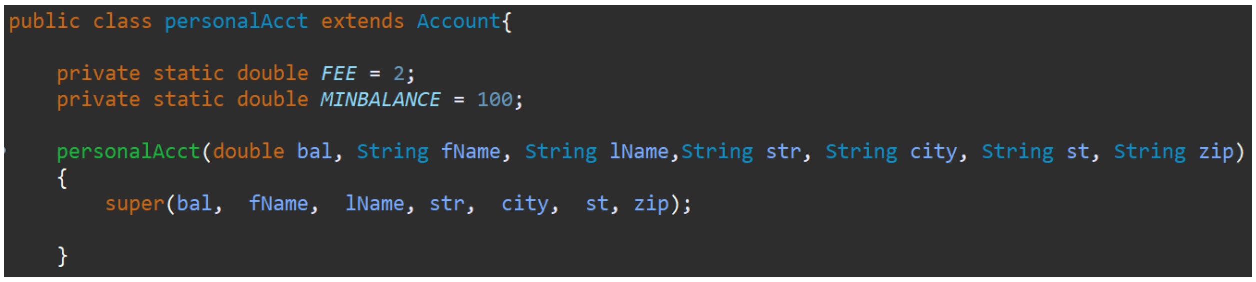
To withdraw money, we can only do so if there are sufficient funds in the account. To account for this we make an if else statement that only deducts the specified amount amount, if that value of cash is less than the bank balance. Otherwise an error message shows indicating insufficient funds.



```java
public String toString()
  {
    String accountString;
    NumberFormat money = NumberFormat.getCurrencyInstance();
    accountString = cust.toString();
    accountString += "Current balance is " + money.format(balance);
    return(accountString);
  }
```

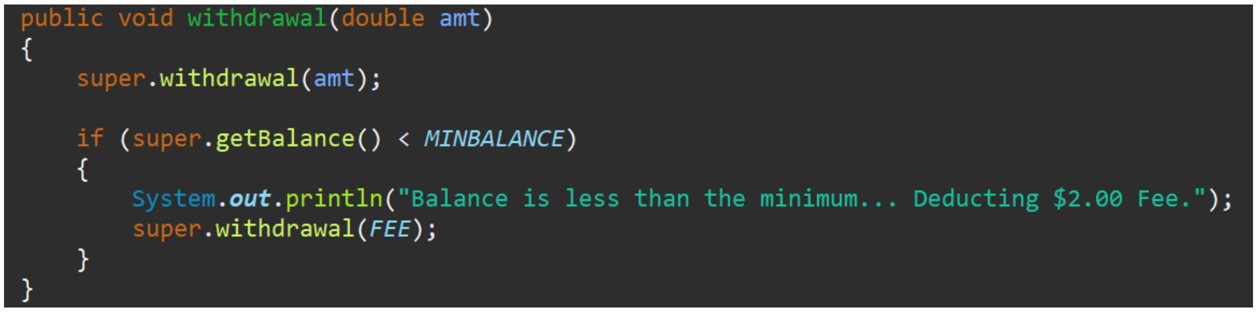
Lastly we make a similar toString() method as done in Customer, which displays the our personal information, as well as our current balance. We also format the balance as a form of money which rounds the value to 2 decimal places and makes it into a string.

From here is where we create our subclasses: personalAcct, and businessAcct.



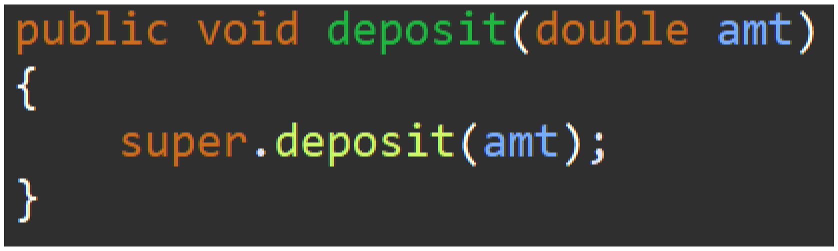
```java
public class personalAcct extends Account{

    private static double FEE = 2;
    private static double MINBALANCE = 100;

    personalAcct(double bal, String fName, String lName,String str, String city, String st, String zip)
    {
        super(bal,  fName,  lName, str,  city,  st, zip);

    }
```

The personal account inherits our Account class and has constants for the deductible fee of $2.00 when a transaction is completed below a min balance(the other constant) of $100.00. Besides that there are no additional parameters so we create the constructor with the same parameters as in the account class. We then create an object from the superclass.



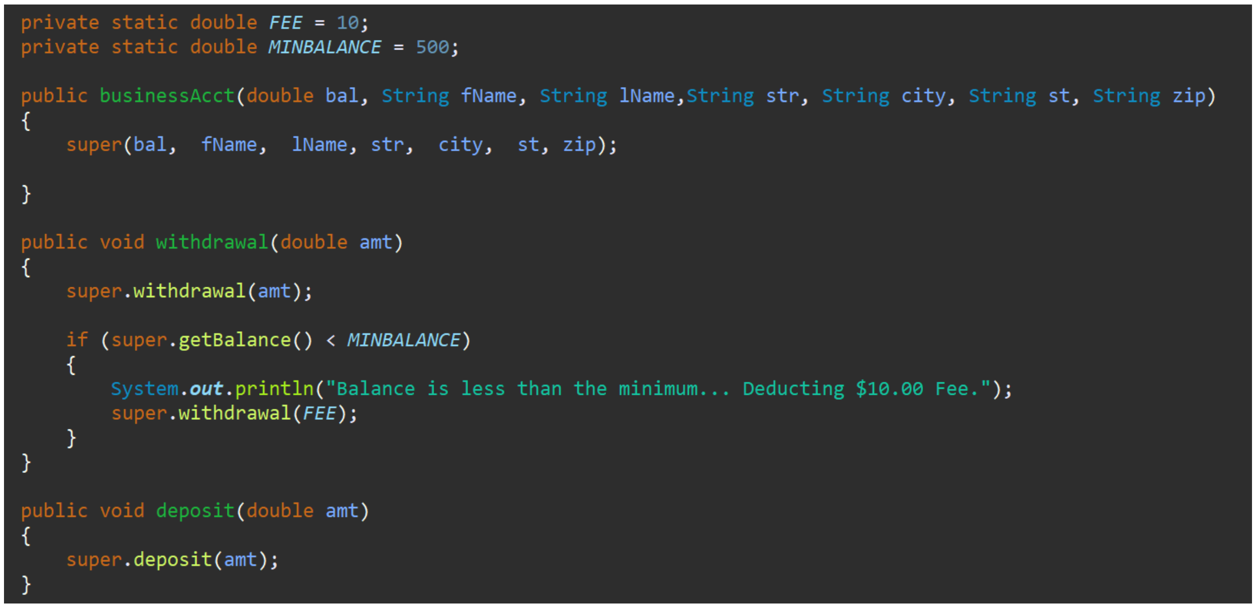
```java
public void withdrawal(double amt)
{
    super.withdrawal(amt);

    if (super.getBalance() < MINBALANCE)
    {
        System.out.println("Balance is less than the minimum... Deducting $2.00 Fee.");
        super.withdrawal(FEE);
    }
}
```

To implement our deductible fee, we override the withdrawal method in the account class: firstly we call the withdrawal method from our superclass, Account() and then we impose the conditional if statement to deduct $2.00 if our balance is below the minimum value. We print a message to highlight why additional money was deducted and how much.



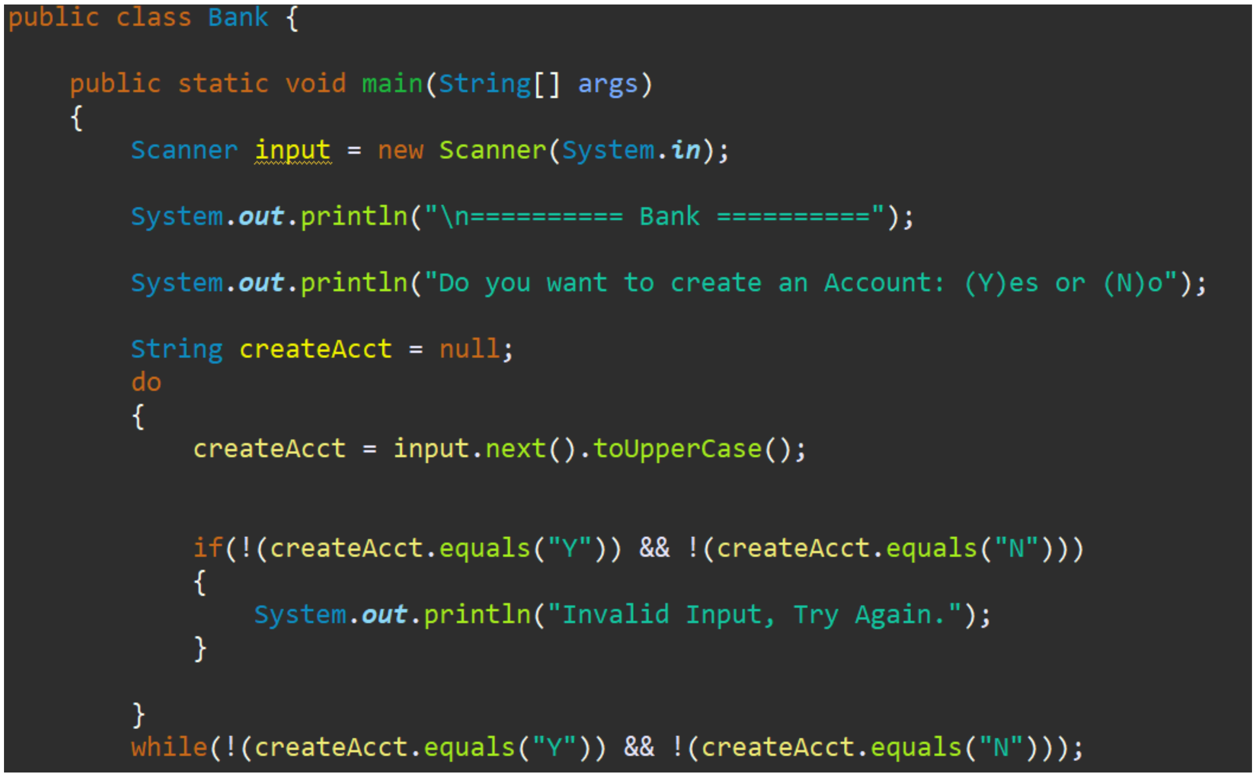
```java
public void deposit(double amt)
{
     super.deposit(amt);
}
```

Our deposit method requires no changes and so we simply call the superclass' method.



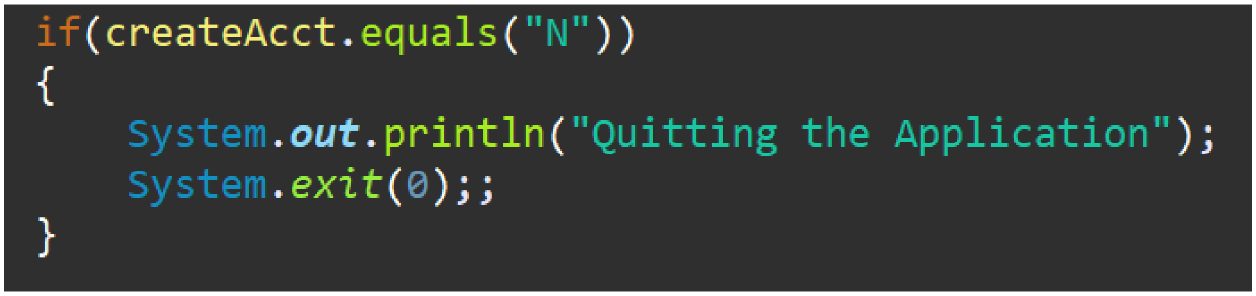
```java
private static double FEE = 10;
private static double MINBALANCE = 500;

public businessAcct(double bal, String fName, String lName,String str, String city, String st, String zip)
{
    super(bal,  fName,  lName, str,  city,  st, zip);

}

public void withdrawal(double amt)
{
    super.withdrawal(amt);

    if (super.getBalance() < MINBALANCE)
    {
        System.out.println("Balance is less than the minimum... Deducting $10.00 Fee.");
        super.withdrawal(FEE);
    }
}

public void deposit(double amt)
{
    super.deposit(amt);
}
```

Our businessAcct is set up almost identically to the personalAcct class, inheriting the Account class, and the only change is the values of our constants, as we have an increased fee and minimum balance of $10.00 and $500.00 respectively.



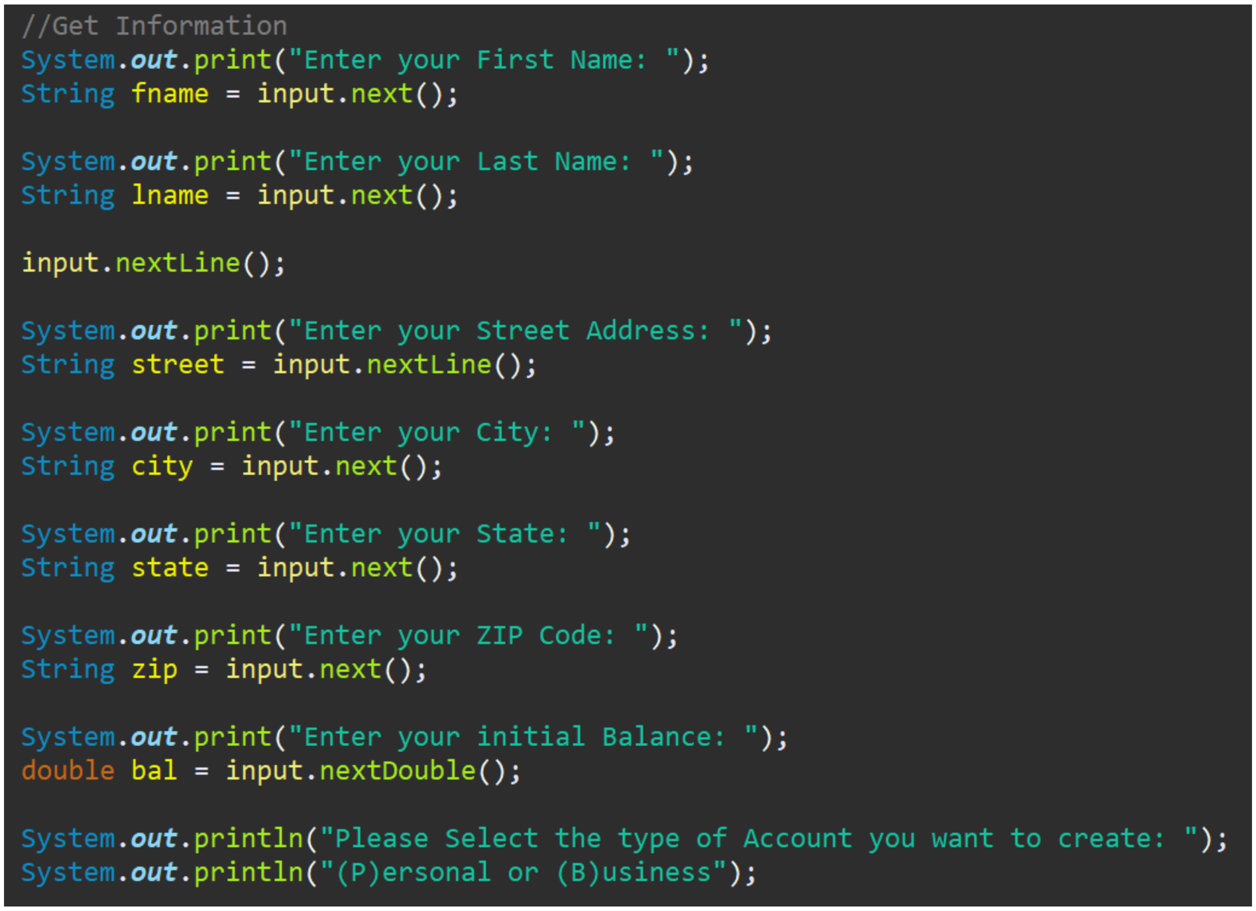
```java
public class Bank {

    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);

        System.out.println("\n========== Bank ==========");

        System.out.println("Do you want to create an Account: (Y)es or (N)o");

        String createAcct = null;
        do
        {
            createAcct = input.next().toUpperCase();


            if(!(createAcct.equals("Y")) && !(createAcct.equals("N")))
            {
                System.out.println("Invalid Input, Try Again.");
            }

        }
        while(!(createAcct.equals("Y")) && !(createAcct.equals("N")));
```

Our tester class is the bank application. We first implement the scanner for user inputs and then ask the user if they want to create an account or not, requiring a non-case sensitive "y" to create an account and a "n" to not create an account and stop the program. We encase our response, labelled createAcct in a do- while loop to ensure invalid inputs do not crash the application. If the user types an invalid response, then an if statement recognizes this an prints an error message. This loop will continue while the createAcct variable is neither "y" nor "n".



```java
if(createAcct.equals("N"))
{
    System.out.println("Quitting the Application");
    System.exit(0);;
}
```

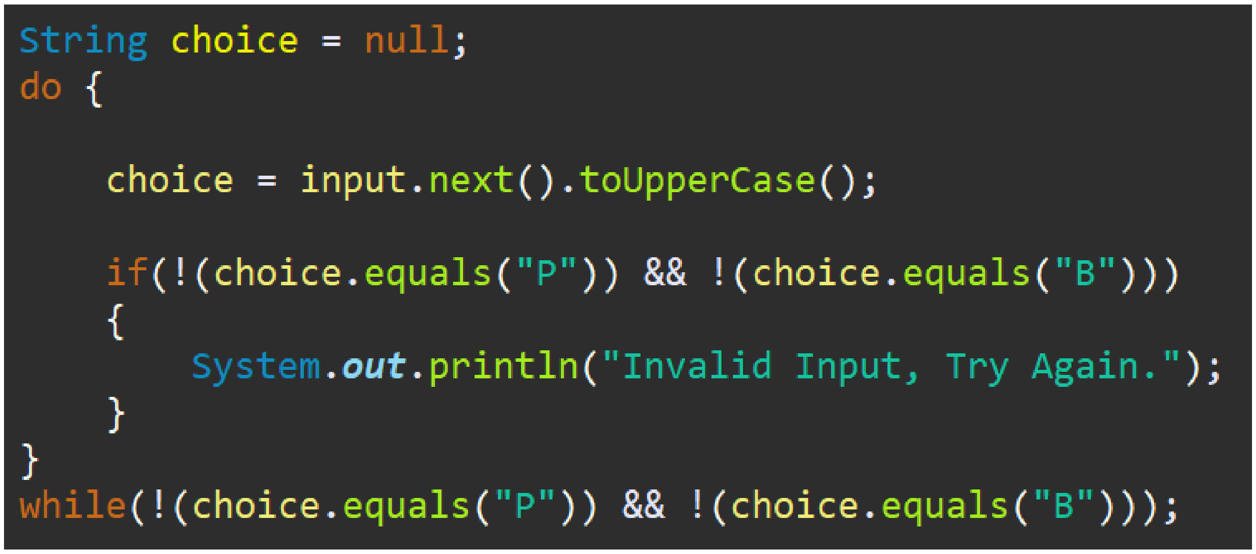
If the user typed in "n",we quit the program.



```java
//Get Information
System.out.print("Enter your First Name: ");
String fname = input.next();

System.out.print("Enter your Last Name: ");
String lname = input.next();

input.nextLine();

System.out.print("Enter your Street Address: ");
String street = input.nextLine();

System.out.print("Enter your City: ");
String city = input.next();

System.out.print("Enter your State: ");
String state = input.next();

System.out.print("Enter your ZIP Code: ");
String zip = input.next();

System.out.print("Enter your initial Balance: ");
double bal = input.nextDouble();

System.out.println("Please Select the type of Account you want to create: ");
System.out.println("(P)ersonal or (B)usiness");
```

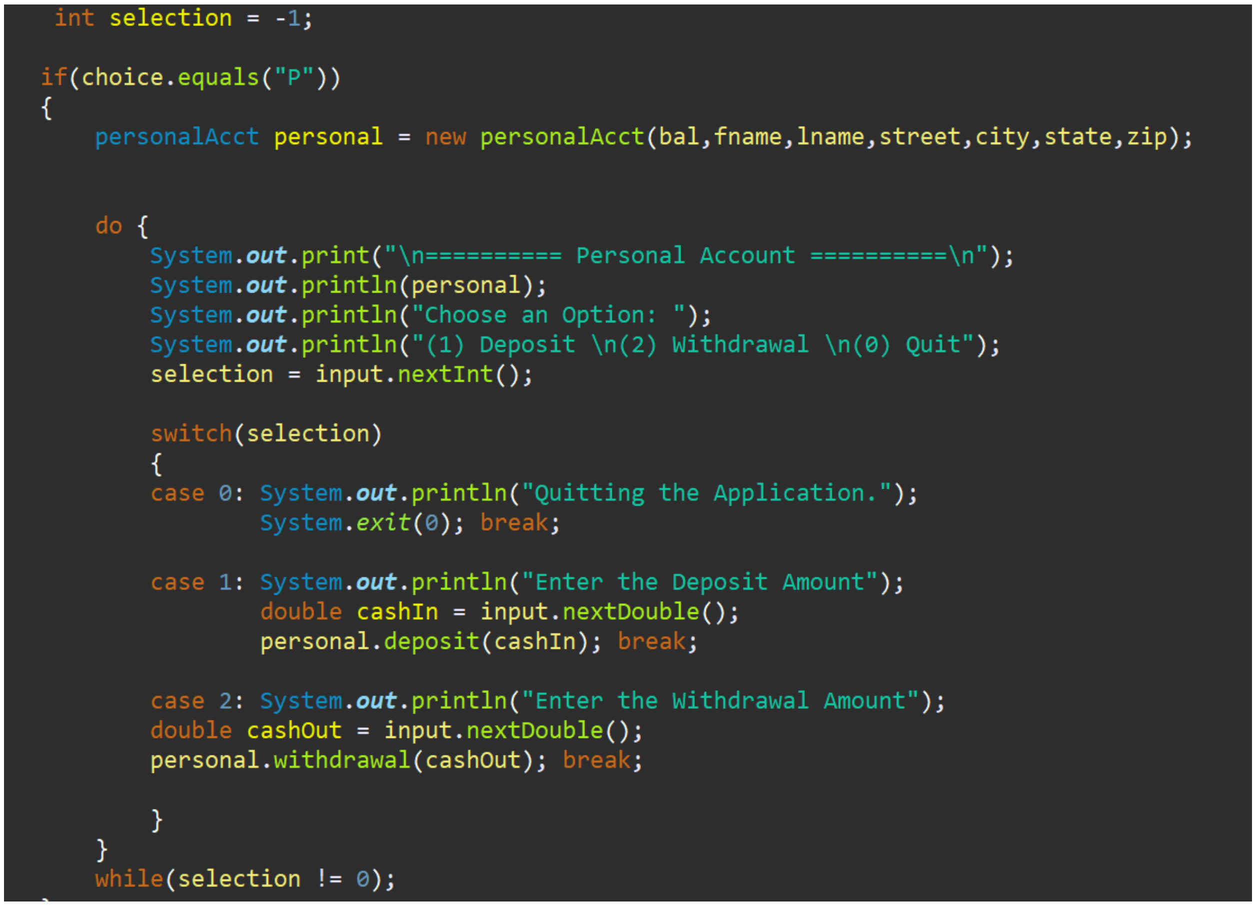
We then get the relevant information from the user. There is an input.nextLine() before the street input to consume the left over newline from the last name input so that the street variable doesn't return a null value.

Then we ask the, the type of account they want to make, business or personal.



```java
String choice = null;
do {

     choice = input.next().toUpperCase();

     if(!(choice.equals("P")) && !(choice.equals("B")))
     {
          System.out.println("Invalid Input, Try Again.");
     }
}
while(!(choice.equals("P")) && !(choice.equals("B")));
```

Then to get their choice, we create the same do while statement as before, to only get valid inputs.

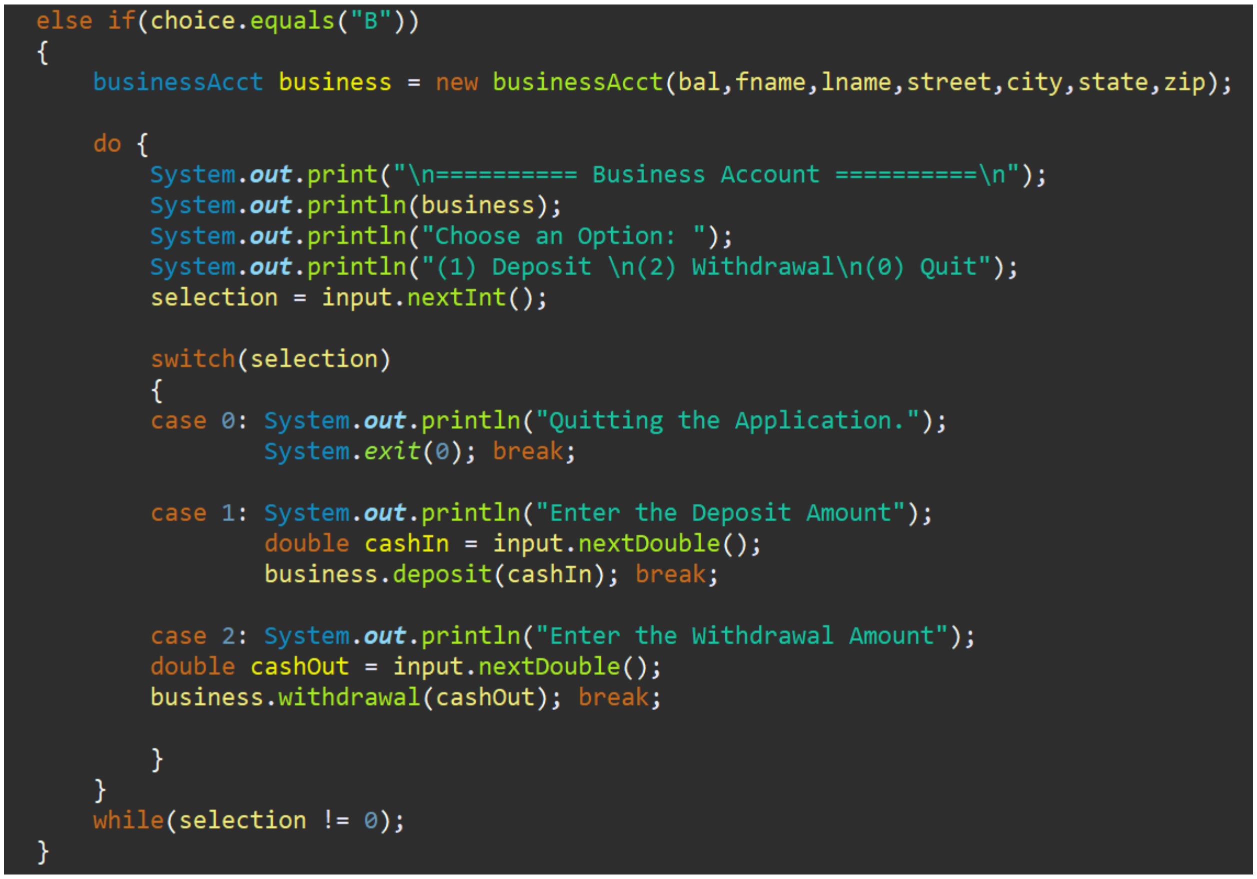


```
 int selection = -1;

if(choice.equals("P"))
{
    personalAcct personal = new personalAcct(bal,fname,lname,street,city,state,zip);


    do {
        System.out.print("\n========== Personal Account ==========\n");
        System.out.println(personal);
        System.out.println("Choose an Option: ");
        System.out.println("(1) Deposit \n(2) Withdrawal \n(0) Quit");
        selection = input.nextInt();

        switch(selection)
        {
        case 0: System.out.println("Quitting the Application.");
                System.exit(0); break;

        case 1: System.out.println("Enter the Deposit Amount");
                double cashIn = input.nextDouble();
                personal.deposit(cashIn); break;

        case 2: System.out.println("Enter the Withdrawal Amount");
        double cashOut = input.nextDouble();
        personal.withdrawal(cashOut); break;

        }
    }
    while(selection != 0);
```

We create an int variable called selection which indicates what action they want to do on their account, we initialize it to a value not interfering with the next do-while and also which doesn't choose one of the listed options. Then we indicate that we are in the personal account and ask the user to choose an option as to what they would like to do in their account.

We then make a switch statement to handle the different cases:
If they choose 0 , then we quit the application.
If they choose 1, then we are depositing cash, so we prompt the user to enter the amount to add, and then we call the deposit method from personalAcct using their specified amount as a parameter.
If they choose 2, then we are withdrawing money, so we use the code from the deposit statement but instead call the withdrawal method instead.

The do while loop is set while the program is not quit so the user can perform multiple actions.



```
else if(choice.equals("B"))
{
    businessAcct business = new businessAcct(bal,fname,lname,street,city,state,zip);

    do {
        System.out.print("\n========== Business Account ==========\n");
        System.out.println(business);
        System.out.println("Choose an Option: ");
        System.out.println("(1) Deposit \n(2) Withdrawal\n(0) Quit");
        selection = input.nextInt();

        switch(selection)
        {
        case 0: System.out.println("Quitting the Application.");
                System.exit(0); break;

        case 1: System.out.println("Enter the Deposit Amount");
                double cashIn = input.nextDouble();
                business.deposit(cashIn); break;

        case 2: System.out.println("Enter the Withdrawal Amount");
        double cashOut = input.nextDouble();
        business.withdrawal(cashOut); break;

        }
    }
    while(selection != 0);
}
```

If the user selected the business account instead we create a businessAcct object instead but the rest of the code remains the same. The only observable difference between the codes is the difference in the deduction fee.