Credit Name: File Structures and Exception Handling
Assignment Name: MySavings

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

Credit Name: Object Oriented Programming
Assignment Name: My Savings

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

```
public PiggyBank()
{

}



public void addPenny()
{

}


public void addNickle()
{

}

public void addDime()
{

}

public void addQuarter()
{

}
```

I started out by creating the constructor method and 4 methods for adding different types of coins into the piggy bank. They were all empty at first because I just wanted to get started and see what I would need for the program first.

```
public void removePenny()
{

}
```

I also realized that I needed to add methods for removing coins as well so I created those.

I started by going to last year's reflection log for the PiggyBank code and implementing the code I used for that project. I also added slight changes to the original code to

remove methods I wouldn't need as well as checking if the user had enough money to remove a coin inside of the methods instead of in MySavvings.

```
//Taking out methods
public void removePenny()
{
    if(totalMoney >= 0.01)
    {
        totalMoney -= 0.01;
        System.out.println("A penny was removed.\n");
    }
    else
    {
        System.out.println("There is not enough money to remove a penny.\n");
    }
}
```

```
public static void main(String[] args) {
 //Declaration area
 File dataFile;
 FileOutputStream fileOut;
 ObjectOutputStream objectOut;
 FileInputStream fileIn;
 ObjectInputStream objectIn;
 int choice;


 Scanner input = new Scanner(System.in);
```

I started working on MySavings by implementing the variables I would need as well as setting up Scanner for user input.

```
//Load the saved piggy bank
if(loadChoice.toLowerCase().equals("y"))
{
    try
    {
    dataFile = new File("C:\\Users\\logan\\Downloads\\savedBank.txt");
    fileIn = new FileInputStream(dataFile);
    objectIn = new ObjectInputStream(fileIn);



    userBank = (PiggyBank) objectIn.readObject();


    System.out.println(userBank);
    System.out.println("Saved bank has loaded.\n");


    objectIn.close();
    fileIn.close();
    }
```

Next I worked on creating the code for allowing the user to load a previous piggybank object if one existed.

```
}
//Create a new bank
else
{
    PiggyBank userBank = new PiggyBank();
}
```

I then added the code that would load the last saved bank from a file or create a new bank if they decided not to load the saved one.

```
//Start the main program
while(quit == false)
{
    //Menu
    System.out.println
    (
    "Please enter a number that corresponds to one of the possible choices.\n" +
    "1. Show total money in the bank.\n" +
    "2. Add money to the bank.\n" +
    "3. Take out money from the bank.\n" +
    "4. End the program."
    );
    optionChoice = input.nextInt();
```

Outside of the if conditional I started to work on the menu and choices that the user could make for MySavings.

```
switch(optionChoice)
{
    case 1:
        System.out.println(userBank);
    break;



    case 2:

    break;



    case 3:

    break;



    case 4:
        quit = true;
    break;
}
```

I started working on a switch conditional to take the user's choice and process what they wanted. I left case 2 and 3 for last because they would be the hardest. When working on the switch conditional I had to go back and initialize userBank outside of the initial if/else statement as well as remove the else because it was no longer needed.

```
//Add a coin
case 2:
    System.out.println
    (
    "What would you like to add?\n" +
    "1. Penny.\n" +
    "2. Nickle.\n" +
    "3. Dime.\n" +
    "4. Quarter."
    );
    addedAmount = input.nextInt();


    switch(addedAmount)
    {
        case 1:
            userBank.addPenny();
        break;
        case 2:
            userBank.addNickle();
        break;
        case 3:
            userBank.addDime();
        break;
        case 4:
            userBank.addQuarter();
        break;
        default:
            System.out.println("Invalid option. Please try again.\n");
        break;
    }
break;
```

I filled in both case 1 and 2 with a secondary menu that would ask the user which coin to add or remove.

```
System.out.println("Thank you for using this program. Would you like to save this bank? [Y]es or [N]o: ");
input.nextLine();
saveChoice = input.nextLine();
```

Finally I had to work on the serialization of the object when the user quits the program. I started this by asking the user if they would like to save the bank or not.

```
//Check if the user wants to save
if(saveChoice.toLowerCase().equals("y"))
{
    try
    {
        dataFile = new File("C:\\Users\\logan\\Downloads\\savedBank.txt");
        fileOut = new FileOutputStream(dataFile);
        objectOut = new ObjectOutputStream(fileOut);


        objectOut.writeObject(userBank);


        System.out.print("\nBank saved.");


        objectOut.close();
        fileOut.close();
    }
```

I then ended it with an if statement to check if they wanted it to save and if they did, the program will save the file in a try statement.