Mastery MySavings Reflection Log

Credit Name: CSE 3120 Object-Oriented Programming 1
Assignment Name: Mastery MySavings

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

From planning to coding I took the UML diagram that I made and started to flush out each variable and method in their respective class (MySavings and PiggyBank) into code. Each part became more complex as I made a line of code for the various methods and variables spread out between the two classes. Adding the object also took much longer than the simple diagram in the UML diagram; however after watching some videos I figured out how to properly transfer information between classes, leading to a successful program.

From coding to now I did not change any of the code besides adding the missing switch statement in the PiggyBank class for removing the number of coins in the user's bank account (explained in my error log), along with this I completed the error log and the reflection log after finishing coding.

Steps

MySavings

(the package this file is in is mastery)
I started off by importing java.util.scanner which allows me to get different data types from user inputs. The public class is declared as MySavings (It's declared as public so the PiggyBank class can access its information and return information).

```
package mastery;

import java.util.Scanner;

public class MySavings { /
```

Below it shows the public (accessible from other classes) method "main" is static (you can access it without creating an object), void (method doesn't return any value), main (method name), and String[ ] args (String represents the data type [ ] indicates that it will be an array, and args (arguments) is the name of the array).

```
public static void main(String[] args) {
```

I then created the piggyBank object.
-PiggyBank is the type of object created (is the PiggyBank class which contains the methods and variables related to the user's bank).

-piggyBank is the name of the object that the PiggyBank class above is assigned to (PiggyBank classes(and now objects) values will become piggyBank objects values as well)
-new is used to create memory(space) to store the piggyBank object
-PiggyBank() this is a constructor that initializes the new object (filling the memory(space) with information for the object from the class PiggyBank)

```
PiggyBank piggyBank = new PiggyBank();
```

The Scanner object is created below to take in inputs from the user. (I named mine scanner)

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

Next, I declared the integer variable choice. I do this outside of the do-while loop, because you need to keep track of the user's input across multiple iterations(loops). If you did it inside of the loop you would lose the previous value every time the loop ran again. Also the while part of the loop needs choice outside of the loop to check its restrictions on choice.

```
int choice;
```

After this I created a welcome statement for the user, this is outside the loop because it doesn't make sense to constantly welcome the user. Doing outside ensures this only prints at the start.

```
System.out.println("Welcome to Hielan's Bank!");
```

I then began the do-while loop after declaring and printing the two previous things (int choice and welcome message)

```
do {
```

Next, I made a print statement that said Menu, this just indicated to the user that what they're looking at is the menu options.

```
System.out.println("\nMenu:");
```

After this I created the first option (1), which showed the total in the user's bank (this is done in the PiggyBank class).

```
System.out.println("1. Show total in bank");
```

I then created the four coin adding options, penny (option 2), nickel (option 3), dime (option 4), and quarter (option 5). These would add their respective value to the user's bank when chosen (explained more in PiggyBank reflection section)

```
System.out.println("2. Add a penny"); /
System.out.println("3. Add a nickel");
System.out.println("4. Add a dime"); //
System.out.println("5. Add a quarter");
```

Next, I made the option (6) for removing all of the coins(money) in the user's bank (this is done in the PiggyBank class as well)

```
System.out.println("6. Remove all coins");
```

Lastly, I made the option to quit (0) (stop the program) for when the user wants to stop using/adjusting things in their bank account.

```
System.out.println("0. Quit");
```

After all of the options I made a message to prompt the user to enter their choice.

```
System.out.print("Enter your choice: ");
```

I then saved their response, and made this value the value of the integer variable choice that was declared outside of the do-while loop. Note: the name of the Scanner object has to match here hence why I put scanner.nextInt().

```
choice = scanner.nextInt();
```

I then made a print statement that would output a response based on the user's choice. It got the response from the PiggyBank objects variable piggyBank, which got the response from the PiggyBank class method Options. (PiggyBank gets this value by

executing code based on the variable choice (choice is passed to Options as a parameter) as seen below this is highlighted further in the PiggyBank reflection section)

```
System.out.println(piggyBank.Options(choice));
```

After this, I closed the do-while loop and made the while restriction (choice != 0), this means as long as the user doesn't enter choice 0 from the menu the program will continue to run (0 is the quit option).

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

If the user does select the quit option (0), then this farewell message will be printed before ending the program.

```
System.out.println("Thank you for using Hielan's Bank!");
```

This brace then closes the main method.

```
}
```

And this brace closes the public class MySavings.

```
}
```

Together they look like this.

```
    }
} // c
```

PiggyBank

(The package this file is in is mastery)
I start off by declaring the PiggyBank class as public so that the MySavings class can access and send information to it.

```
package mastery;

//PiggyBank class is set
public class PiggyBank {
```

I then declare the four coin types as private int variables because here they are only used in the PiggyBank class. I set their value to 0 to initialize each variable outside of the Options method.

```
private int pennies = 0;
private int nickels = 0;
private int dimes = 0; //
private int quarters = 0;
```

Next, I created the Options method. It was public (accessible by MySavings), and I declared its type as string so that it could return the string output messages (allows me to take in the int variable choice while also outputting string messages all in one method). Lastly, I gave it the integer parameter choice from the MySavings class (it's the user's selection 0-6 from the menu)

```
public String Options(int choice) {
```

After this, I created the switch statement that would perform an action based on the user's choice, hence why I passed it the choice variable for the cases.

```
switch (choice) {
```

I then created the first case which correlated to if the user chose option 1 form the menu. (this was show total in bank)

```
case 1:
```

In this case I created a double variable called total. I multiplied the amount of each coin by its respective monetary value and then added all of them together; this value became the value of total.

```
double total = pennies * 0.01 + nickels * 0.05 + dimes * 0.10 + quarters * 0.25;
```

I then made a return message with the total in the user's bank formatted to 2 decimal places as seen below using %.2f. This message was sent to MySavings to be outputted

```
return String.format("Total amount in piggy bank: $%.2f", total);
```

The next case created (case 2) was for pennies. If the user selected the option to add a penny then this case would add 1 to the variable pennies as indicated by ++. I then returned the message “Added a penny.” to MySavings to be printed, confirming that the user’s choice was executed.

```
case 2:// Add a penny (user s
    pennies++; // Adds a penr
    return "Added a penny.";
```

The next case created (case 3) was for nickels. If the user selected the option to add a nickel then this case would add 1 to the variable nickels as indicated by ++. I then returned the message “Added a nickel.” to MySavings to be printed, confirming that the user’s choice was executed.

```
case 3: // Add a nickel (user
    nickels++; // Adds a nicke
    return "Added a nickel.";
```

The next case created (case 4) was for dimes. If the user selected the option to add a dime then this case would add 1 to the variable dimes as indicated by ++. I then returned the message “Added a dime.” to MySavings to be printed, confirming that the user’s choice was executed.

```
case 4: // Add a dime (user
    dimes++; // Adds a dime
    return "Added a dime.";
```

The next case created (case 5) was for quarters. If the user selected the option to add a quarter then this case would add 1 to the variable dimes as indicated by ++. I then

returned the message “Added a quarter.” to MySavings to be printed, confirming that the user’s choice was executed.

```
case 5: // Add a quarter (user
    quarters++; // Adds a quar
    return "Added a quarter.";
```

After adding the four cases for coins I added the sixth case (remove all coins).

```
case 6:
```

To do this I made each coin variable equal to 0, effectively removing every coin.

```
pennies = 0;
nickels = 0;
dimes = 0; //
quarters = 0;
```

I then made a return statement saying “All coins have been removed from your piggy bank.”. This was sent to MySavings to print.

```
return "All coins have been removed from your piggy bank.";
```

The next case (0) I made was for if the user wanted to quit the program (exit their piggy bank account). If the user entered 0 then case 0 would return “Exiting your bank account…” to MySavings to print before finishing/stopping the program.

```
case 0: // Stops the program (this is done
    return "Exiting your bank account...";
```

The last case I made was the default case, this was in case the user entered a non valid option/choice (if a number 0-6 wasn’t selected). This case would then return an error message to MySavings to be printed, that would prompt the user to enter a valid option.

```
default: // If a number 0-6 wasn't selected (for invalid options)
    return "Invalid choice. Please enter a number from the menu options.";
```

I then added a brace to close the switch statement.

```
}
```

A brace to close the Options method.

```
}
```

And a brace to close the public class PiggyBank.

```
}
```

Together they looked like this.

```
        }
    } // C
} // Close
```