Mastery DigitsExtractor Reflection Log

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

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 (DigitsExtractor and Num) 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. Along with this, thinking about how to do the calculations to return the positive value of each place took me a while, but after thinking about it thoroughly, I created a code that executes the proper calculations while keeping the digits place value positive (explained more in the actual reflection log).

From coding to now I did not change any of the code besides the misplaced choice variable in my DigitsExtractor class(explained in error log), along with this I completed the error log and the reflection log after finishing coding.

Note:My DigitsExtractor UML diagram is mostly blank because I didn't declare new variables in the Num class, but instead used the user's input from DigitsExtractor and returned that after doing calculations on it.

Steps

Digits Extractor

(the package this file is in is Mastery (here it says digits because I haven't connected my github at home yet))

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 DigitsExtractor (It's declared as public so the Num class can access its information and return information).

```
package digits;

import java.util.Scanner; // Im

public class DigitsExtractor {
```

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 num object.
-Num is the type of object created (is the Num class which contains the methods and updated variables related to the user's number).
-num is the name of the object that the Num class above is assigned to (Num classes(and now objects) values will become num objects values as well)
-new is used to create memory(space) to store the Num object
-Num() this is a constructor that initializes the new object (filling the memory(space) with information for the object from the class Num)

```
Num num = new Num();
```

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 prompted the user to "Enter an integer" (no decimal places), and I stored their input as an integer variable called number. (note that the Scanner objects name has to match here, hence why I put scanner.nextInt())

```
System.out.print("Enter an integer: ");
int number = scanner.nextInt(); // Save
```

After this, I declared the integer variable choice. I do this outside of the do-while loop, because the while part of the loop needs it outside of the loop to check its restrictions on choice. (explained in my error log)

```
int choice;
```

I then began the do-while loop after declaring and printing/getting the two previous things (int choice and getting the user's number)

```
do {
```

After this I made a message to prompt the user to "Choose an option:"

```
System.out.println("\nChoose an option:");
```

Next, I wrote the four choices pertaining to the user's entered number. This included displaying the whole number, displaying the ones place of the user's number, displaying the tens place of the user's number, and displaying the hundreds, place of the user's number. (note: if the user entered a two-digit number for instance and asked for the hundred place it would print 0 can be seen in my test cases)

```
System.out.println("1. Whole number"); /
System.out.println("2. Ones place");   /
System.out.println("3. Tens place");   /
System.out.println("4. Hundreds place");
```

The next option I added was the quit option (0) for if the user no longer wanted/needed to use the DigitsExtractor program.

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

Lastly, I prompted the user to enter their choice

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

I saved whatever choice they made (0-4) in the integer variable choice (declared outside of the loop) (note: the Scanner objects name has to match here, hence why I put scannernextInt())

```
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 Num objects variable num, which got the response from the Num class method Options. (Num gets this value by executing code based on the variables choice and number (choice and number is passed to Options as a parameter) as seen below this is highlighted further in the Num reflection section)

```
System.out.println(num.Options(number, 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);
```

I then added a brace to close the main method.

```
}
```

And a brace to close the public class DigitsExtractor

```
}
```

Together they looked like this.

```
    }
} //
```

Num

(The package this file is in is mastery(here it says digits because I haven't connected my github at home yet))
I start off by declaring the Num class as public so that the DigitsExtractor class can access and send information to it.

```
package digits;

//Num class is set t
public class Num {
```

Next, I created the Options method. It was public (accessible by DigitsExtractor), and I declared its type as string so that it could return the string output messages (allows me to take in the int variables choice and number while also outputting string messages all in one method). Lastly, I gave it the integer parameters choice and number from the DigitsExtractor class (it's the user's selection 0-4 from the menu, and the number they entered initially)

```
public String Options(int number, 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 the whole number)

```
case 1:
```

All I did for this case was have a return message plus the number variable (just the number that the user entered), I returned this to DigitsExtractor to be printed for the user to see.

```
return "Whole number: " + number;
```

The next case (2) I made was for finding the ones place of the user's number. I wrote the return message and then to find the ones place of the user's number I did Math.abs(number) % 10. The calculation for the ones places is simply number (the entered user number) modulo 10. This divides the number by 10 and leaves the remainder which would be the ones place. I use Math.abs here to make sure a positive

number is returned, this is for incase the user enters a negative number. I do this because the ones place would not be for example -7 in -237 but 7, (-) is simply the indication of whether the number is positive or negative. All of this information (message plus ones place gets sent to DigitsExtractor to be printed for the user to see)

```
case 2: // Ones place (user selects choice 2)
    return "Ones place: " + Math.abs(number) % 10;
```

The next case (3) I made was for finding the tens place of the user's number. I wrote the return message and then to find the tens place of the user's number I did Math.abs(number / 10) % 10. To get the tens place we first divide the user's number by 10 to get rid of the ones place (Ex. 235/10 = 23.5 but since we're working with an integer variable this is actually 23), and then I do modulo 10 to get the remainder of this number which will be the tens place number (Ex. 23 % 10 = 3 since the ones are gone this is the tens place). Lastly, as explained before I use Math.abs to ensure the result is positive. I then return this to DigitsExtractor to be printed for the user to see.

```
case 3: // Tens place (user selects choice 3)
    return "Tens place: " + Math.abs(number / 10) % 10;
```

The next case (4) I made was for finding the hundreds place of the user's number. I wrote the return message and then to find the hundreds place of the user's number I did Math.abs(number / 100) % 10. To get the hundreds place we first divide the user's number by 100 to get rid of the tens and ones places (Ex. 2305/10 = 23.05 but since we're working with an integer variable this is actually 23), and then I do modulo 10 to get the remainder of this number which will be the hundreds place number (Ex. 23 % 10 = 3 since the ones and tens are gone this is the hundreds place). Lastly, as explained before I use Math.abs to ensure the result is positive. I then return this to DigitsExtractor to be printed for the user to see.

```
case 4: // Hundreds place (user selects choice 4)
    return "Hundreds place: " + Math.abs(number / 100) % 10;
```

The next case (0) is for if the user wants to exit the program, this will simply return an exiting message to DigitsExtractor to be printed so the user knows the program is stopping.

```
case 0: // Stops the program (this is dor
    return "Exiting DigitsExtractor...";
```

The last case I made was the default case, this was in case the user entered a non valid option/choice (if a number 0-4 wasn't selected). This case would then return an error message to DigitsExtractor to be printed, that would prompt the user to enter a valid option.

```
default: // If a number 0-4 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 Num.

```
}
```

Together they looked like this.

```
        }
    } //
} // Clc
```