Mastery Prime Numbers Reflection Log

Credit Name: CSE 2110 Procedural Programming 1
Assignment Name: Mastery Prime Numbers

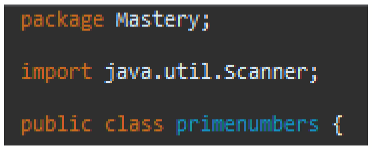
(Note: the screenshots color changed because I used snipping for the latter half of the code at home and screenshotted the code fro github)

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

From planning to coding I took the IPO chart that I made and started to flush out each line into code. Each part became more complex as I made a line of code in the main method to prompt the user for a number, and created a calculation in a new method isPrime using the square root of the user input and a variable called i to check if a number is prime, this was then sent to the main method to be printed for the user.

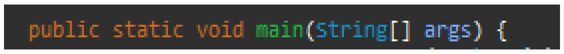
From coding to now I did not change too much, the two things I did were adding an if statement to for the numbers 1 and below (to output as not prime) and then adding this error/change to my error log.

Steps
(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 inputs. The public class is declared as primenumbers (public makes it accessible from anywhere).



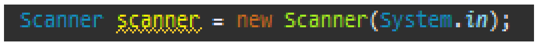
```
package Mastery;

import java.util.Scanner;

public class primenumbers {
```

Below it shows the public 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).



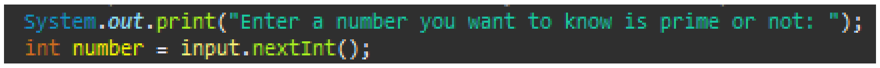
```
public static void main(String[] args) {
```

The scanner object input is created below to take in inputs from the user.



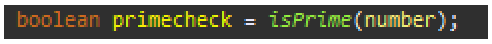
```
Scanner scanner = new Scanner(System.in);
```

After this I created a print statement to prompt the user for the number they want to know is prime or not. I saved this in an integer variable called number.



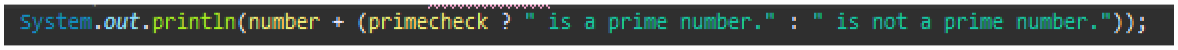
```
System.out.print("Enter a number you want to know is prime or not: ");
int number = input.nextInt();
```

Then I made a boolean variable called primecheck which will be used to determine what the output statement will print. (if primecheck is false it will print (user's number) not a prime number, if it's true then it will print (user's number) is a prime number) (note that it will get its true or false value from the method isPrime, the variable number in brackets is passed to isPrime and used to determine whether the number is true or false, number then gets returned to the main method as a true or false and is used to determine whether primecheck is true or false Ex. if number returns false primecheck becomes false) (this becomes clearer after going through the method isPrime)



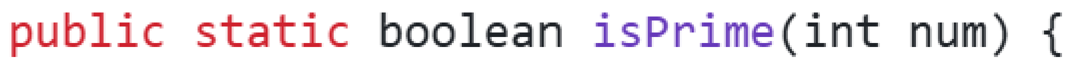
```java
boolean primecheck = isPrime(number);
```

Next, I made an output statement that would print if the number is prime or not based on the boolean variable primecheck. If primecheck returned true from the method isPrime then it would print (user's number) is a prime number, but if primecheck returned false from the method isPrime then it would print (user's number) is not a prime number. (I do the variable number plus the sentence so that it displays the number that they wanted to know is prime or not) (also ? : just serves as an if-else statement



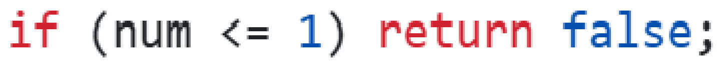
```java
System.out.println(number + (primecheck ? " is a prime number." : " is not a prime number."));
```

After I created a public static boolean method called isPrime that took in the integer variable num as its parameter.



```java
public static boolean isPrime(int num) {
```

Then I added an if statement that would return false if the number was equal to or below 1, this ensures that the program would display that the numbers 1 and below is not a prime number. If you don't have this if statement these numbers come out as prime from the calculations but this isn't true. (my error log has a more in-depth explanation of this)



```java
if (num <= 1) return false;
```

Next, I took the for loop from my original prime numbers code and altered it. First, it makes the integer variable i (declared in the loop) equal to 2. After this, it checks if i is equal to or less than the square root of num (this is because I found out you don't need to divide by everything until the number but only until its square root since num = a * b one of either a or b has to be less then the square root of num; this means you will have divided by one of either a or b by the time you get to the square root of num meaning if there is a number you can divide by you will get to it before or at the square root, so if i is less then or equal to the square root it will continue the loop.). After this, I added 1 to i so the loop can check again (explained more in the next line)

```
for (int i = 2; i <= Math.sqrt(num); i++) {
```

These next lines of code say that if the remainder of num divided by i is equal to 0 to returns false. This is because that means num is evenly divisible by another number other than 1 and and itself, hence it's not prime. This will be sent to the main method to be outputted for the user to see as explained in the output line code.

```
 if (num % i == 0) {
     return false; //
```

After this, I used two braces to close the if statement.

```
      }
 }
```

If the loop goes through all the integer numbers up to the square root of num and doesn't find a number that evenly divides into num then it will return true (the number is prime / only divisible by 1 and itself). This will be sent to the main method to outputted for the user to see as explained in the output line of code.

```
return true;
```

I then add a brace to close the isPrime method code.

```
 }
```

And a brace to close the public class primenumbers.

```
 }
```

Together they look like this.

```
        }
    }
```