Mastery EvansAndOdds Reflection Log

Credit Name: CSE 2120 Data Structures 1
Assignment Name: Mastery EvansAndOdds

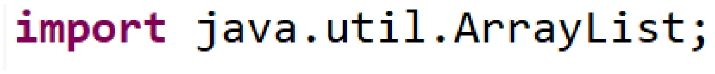
How has your program changed from planning to coding to now? Please explain?

From planning to coding I took the definition of the application from the computer science textbook and started to create each part of code to meet the necessary requirements. Each part became more complex as I made a line of code for the various Arrays/ArrayLists. Although printing the each even and odd number from their respective ArrayList took me a while I eventually got it figured out and was able to properly print these without any errors (explained more in my reflection log and error log)

From coding to now I did not change any of the code besides the error I encountered regarding the printing of the even and odd numbers (explained more in my error log)

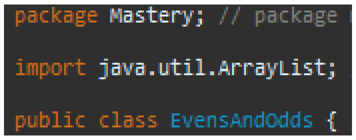
Steps

(the package this file is in is Mastery)
I started off by importing java.util.ArrayList which allows me to create an array that doesn't have a set value (dynamic)



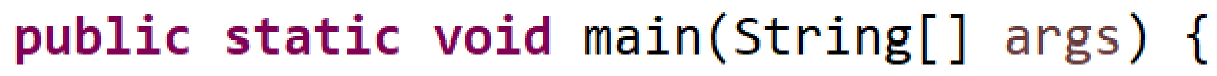
```
import java.util.ArrayList;
```

The public class is declared as DigitsExtractor .



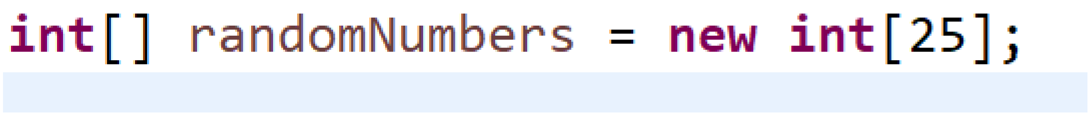
```
package Mastery; // package

import java.util.ArrayList;

public class EvensAndOdds {
```

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).



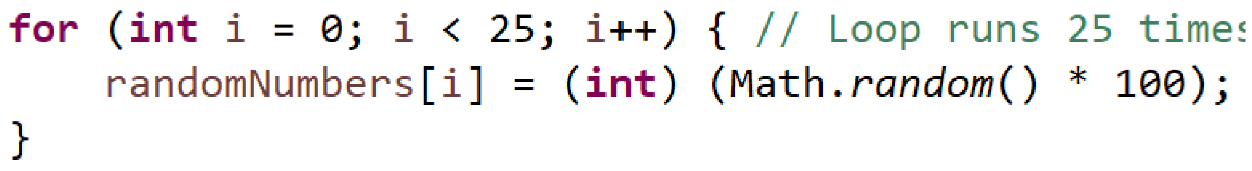
```
public static void main(String[] args) {
```

I then create an array called randomNumbers. The int[] indicates that it will hold integer values, randomNumbers is the array's name, and lastly the new int[25] is what creates the space for all of the random integers to be put into.



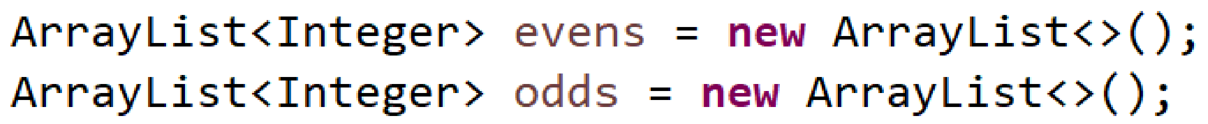
```
int[] randomNumbers = new int[25];
```

Next, I made a for loop that generates all of the random numbers to fill the array with. I start by declaring “i” as an integer = to 0 (note that “i” is an index, basically since the array needs 25 integers 25 indexes will be generated each holding a value for example index 0 = 36 while index 1 = 23). As long as “i” is less than 25 the for loop will add 1 to “i”’s value indicating it’s on the next index (I stop it at 25 because 0 counts as its own index since that’s what “i” is initiated as so it stop at 24 instead of 25 to ensure there are only 25 random numbers made). The next part (randomNumbers[i]) is saying to get the element at index “i” (basically if i = 0 then it’s talking about the first element in the array, if i = 1 it’s talking about the second element in the array, this goes all the way to i = 24 for the 25 indexes). This element's value in the randomNumbers array then becomes equal to the randomly generated number. The number is generated by using the Math.Random() method, this method generates a random number between 0.0 (includes 0.0) and 1.0 (doesn’t include 1.0) this means when multiplied by 100 you always get a number between or equal to 0-99 (because 1.0 isn’t included). Lastly the int part at the start is to convert the double value (Math.Random() generates doubles Ex. 0.1234…) into an integer value so that it meets the integer requirements in the textbook. Ex. Math.random() generates 0.567… which when multiplied by 100 becomes 56.7… and then as an integer becomes 56 (doesn’t become 57 because java automatically round down, this works in my favour because if I got for instance 99.99999 and java didn’t automatically round down it would become 100 which isn’t in restrictions). Once this number is generated the loop continues until the 24th index and then stops because of the restriction i < 25 giving us 25 random numbers. Lastly, I close the for loop with a brace.



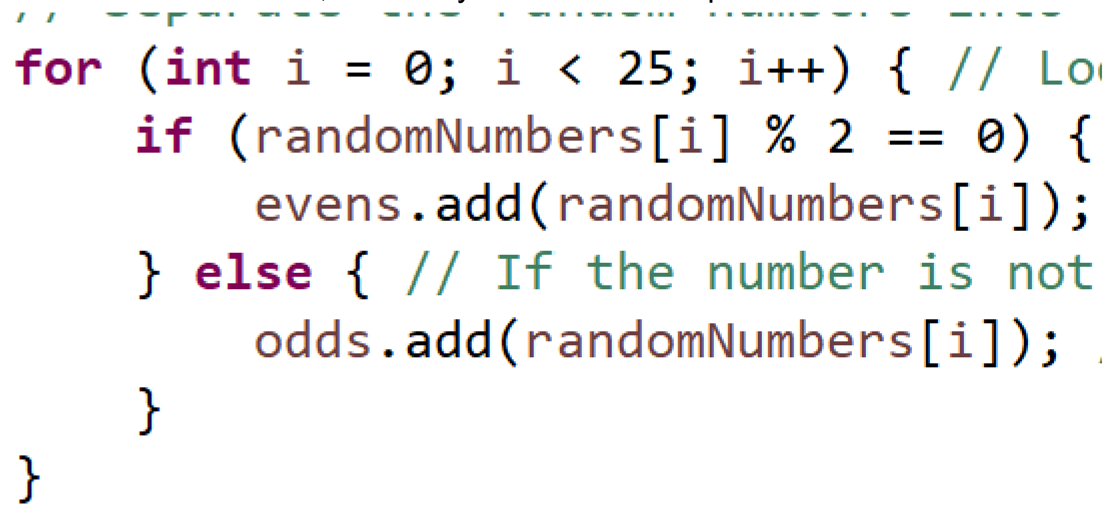
```
for (int i = 0; i < 25; i++) { // Loop runs 25 time
    randomNumbers[i] = (int) (Math.random() * 100);
}
```

After this I created two ArrayLists (this is where importing java.util.ArrayList comes in becomes useful) one list is for even numbers, and the other is for odd numbers. This is indicated by the names I gave them: evens for even numbers and odds for odd numbers. The ArrayList is the class that allows the arrays to be dynamic (this means that their value isn’t set like the first array I made, this is useful because you don’t know how many even, or odd numbers will be generated into each list because it’s random, so even if you have for instance 5 even numbers and 20 odd numbers the array’s will adjust accordingly to this because there value is not set). The <Integer> part indicates that it will only be working with integer numbers. Then I wrote their names as indicated evens and odds for the two different number values. Lastly, the new ArrayList<> code creates an empty array list (where all of the numbers will be sorted into), and the () indicate the default size of the ArrayList which is usually 10 elements, this is before the numbers are added to each ArrayList, so once they are added the ArrayList will adjust its value accordingly.



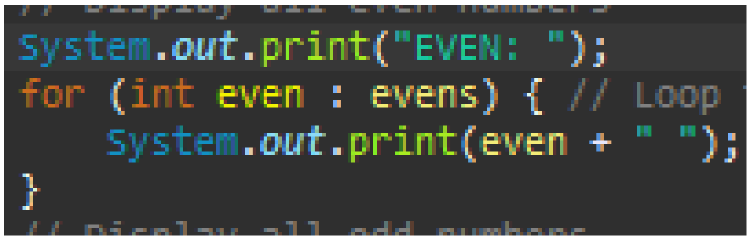
```
ArrayList<Integer> evens = new ArrayList<>();
ArrayList<Integer> odds = new ArrayList<>();
```

I then created another for loop, this one for organizing the numbers into their respective ArrayList. I begin by declaring “i” as an integer equal to 0, “i” will again serve as an index for accessing the elements in the randomNumbers array. I < 25 ensures that the loop runs 25 times for each of the 25 random elements in the randomNumbers array (i = 0 to i = 24), once “i” equals 25 this loop will stop. Next 1 gets added to “i” so that it can move onto the next index and check if the random element in the randomNumbers array is even or odd, this will keep happening until “i” = 25 so that all of the elements are declared even or odd. Below this I start an if-else statement that will actually sort the number into the evens or odds ArrayList. It’s saying if the number(element) in the randomNumbers ArrayList is divisible(modulo) by 2 while leaving no remainder (0) then it is even, hence why it’s then added to the evens ArrayList through the line evens.add(randomNumbers[i]) (this adds the specific element “i” from randomNumbers into the evens list). The else part is for if the number is not divisible by 2 leaving no remainder (0) (it does leave a remainder), this would mean the number is odd so I add it to the odds list through the line odds.add(randomNumbers[i]) (this adds the specific element “i” from randomNumbers into the odds list). I then close the else part of the statement with a brace, and lastly close the for loop with a brace.



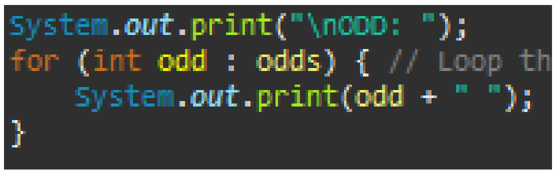
```
for (int i = 0; i < 25; i++) { // Lo
    if (randomNumbers[i] % 2 == 0) {
        evens.add(randomNumbers[i]);
    } else { // If the number is not
        odds.add(randomNumbers[i]);
    }
}
```

After this I create the code to print out the even number from the evens ArrayList. First I print the word “EVEN:” to indicate these are the even numbers. This is followed by a for-each loop that is used to loop through each element in the evens ArrayList. I start by saying that the elements names will be even (this is so it's more straightforward to read/understandable) and are int variables. These variables are from the evens ArrayList that I created before with all of the randomly generated even numbers. I then write the print statement of the for loop that will print the even variable from the evens ArrayList and a space so that the output is organized neatly. This loop will continue printing FOR EACH (hence the for-each loop) element in the ArrayList evens.



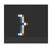
```
System.out.print("EVEN: ");
for (int even : evens) { // Loop
    System.out.print(even + " ");
}
```

I then create the code to print out the odd number from the odds ArrayList. First I print the word “ODD:” to indicate these are the odd numbers. This is followed by a for-each loop that is used to loop through each element in the odds ArrayList. I start by saying that the elements names will be odd (this is so it's more straightforward to read/understandable) and are int variables. These variables are from the odds ArrayList that I created before with all of the randomly generated odd numbers. I then write the print statement of the for loop that will print the odd variable from the odds ArrayList and a space so that the output is organized neatly. This loop will continue printing FOR EACH (hence the for-each loop) element in the ArrayList evens.



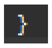
```
System.out.print("\nODD: ");
for (int odd : odds) { // Loop th
    System.out.print(odd + " ");
}
```

Then I add a brace to close the main method.



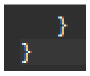
```
}
```

And a brace to close the public EvensAndOdds class.



```
}
```

Together they look like this.



```
    }
}
```