Alvin Tran

CSE 220-50

Assignment #4

Submission Date: 7/12/2021

Problem 1

Problem 1 requires us to create a **class** Course that contains an **ArrayList** representing students, which is derived from a premade student class. We are required to create a **constructor** for the name of the Course, and provide **methods** that add students, in addition to averaging all the test scores of the students. Furthermore, there must be a **roll** method to print out all student names who are in the course.

To solve, we first set up the Student class provided. More importantly, we must modify the Student class to hold testScore values instead of gpa values, and appropriately return that. Afterwards, we create the Course class and create a arrayList using **ArrayList<student> = new ArrayList<student>**, making an array list and defining the type of class. Then we create a constructor simply taking the name String input, and an add student using **ArrayList.add()**. Following, we create a way to get average test score, which is simply made with a **for** loop getting **students :** course and getting sum of test scores, and dividing by **ArrayList.size.** Finally, we provide a way to output all students with **course.roll()** by using a **for** loop and incrementing until we hit **arraylist.size.** Inside, we use **arrayList.get(i)** to get each student and access their name values.

```
Student ralph = new Student("Ralph", "Smith", 90.0);
Student amy = new Student("Amy", "Johnson", 100.0);
Student jerry = new Student("Jerry", "Kim", 80.0);

Course history = new Course("History");
```

Parameters: Simple Average of 90

```
Students in History
                 First name      Last Name
Student 1:       Ralph           Smith
Student 2:       Amy             Johnson
Student 3:       Jerry           Kim
Test average: 90.0
```

Expected Output aligns with parameters.

Problem 2

Problem 2 requires us to create an **Interface** that has a set/get method to a numeric representation of a priority. We then implement the interface using a class named **Task**, be able to compare between priorities, and exercise these functions.

To create the interface, we use Eclipse's **Interface** creator and add it to our package. We then define out **get/set** methods for retrieving and setting our data. Then, we use the **implements** keyword in out class to get our methods. Finally, we create a comparison method by taking an argument of type **Task** and using our **.getPriority()** we defined in our interface. We make appropriate comparisons and output our result.

```
Task homework = new Task("Homework", 1, 2);
Task call = new Task("Call Fred", 2, 6);
Task practice = new Task("Practice", 3, 4);
```

Initial Conditions

```
As of now:
Task Call Fred has a greater priority than Practice at 2
Changing Calling Fred to less urgent (priority 3)
New priority of calling Fred: 3

Moving practice up in priority to 2 from 3
Practice priority is now 2
New priority:
Task Practice has a greater priority with Call Fred at 2

And homework compared to everything else:
Task Homework has a greater priority than Practice at 1
Task Homework has a greater priority than Call Fred at 1
```

Exercised Functions: Compare, Set, and Get Priority

Problem 3

Problem 3 wants us to read an undefined number of integers within 0-50 and count how many of each has occurred. We then output the count of each integer after a value out of range in inputted if we had at least 1 input.

To create an indefinite loop, we utilize a **while** loop that checks if our input is within range. We keep count of all these integers by using **int[]**, and we assign each number 0-50 inclusive to the array matching array index (ie, [0] is representing count of 0's). Each time we successfully input a number in range, we add 1 to the count in the appropriate index, which we can key using our input since they are matching integers. If we ever get an out of range integer, we exit the loop, and use a **for** loop that goes through out entire integer array checking for any values greater than 0. If so, we output the count of that integer.

```
Enter an integer from 0-50, exit by entering out of range integer:
1
Enter another integer:
2
Enter another integer:
2
Enter another integer:
3
Enter another integer:
3
Enter another integer:
3
Enter another integer:
45
Enter another integer:
100
```

Input: 1 1's, 2 2's, 3 3's, a 45

```
Enter another integer:
100
Count of 1's: 1
Count of 2's: 2
Count of 3's: 3
Count of 45's: 1
```

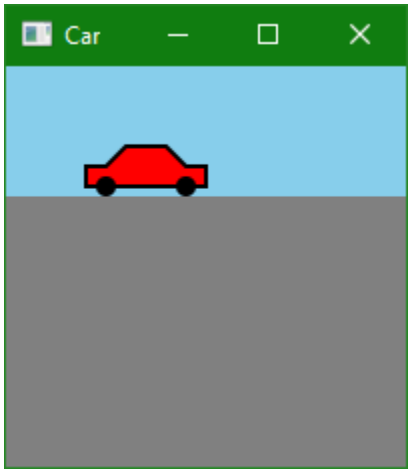
Output Matches: Exits on Out of Range, and output correct count.

Problem 4

Problem 4 wants us to create a class named Car the **extends** the group javaFX class. We do this by utilizing polygons and poly-lines. We then must display the car.

We create the **Car** class using the **extends** keyword, extending the Group class. This allows use to create a car similar to how we create objects and arrays. The polygon and polyline classes that make up our car necessitate the use of integers to designate points. We define those in an **double[]** and use those to create the car body, and a black outline. We then use the **line**, **rectangle, and circle** classes to add correct detailing. After so, we use **getChilden()** to get all members of the **group** and use **.addAll()** to add our shapes to the group.

In our driver function, we use **Car car = new Car()** to create a new car image, put it as our **scene,** and **stage** it.



Car Image Produced