Junit Lab 1A:

What I learned:

I refreshed my memory on how to create a Junit class with setUp(), tearDown(), and associated methods from a class. I learned that the setUp() method is called before every test method, @before; while @beforeClass is called before all the test methods. TearDown() is called after each test method, thus @after.

I refreshed myself on assertEquals along with assertTrue. Both allow us to verify and check that we are getting the expected outputs.

Issues:

I kept having a small debugging issue with the toString method. Turns out that I forgot to add the + to the +=. Not proud to say how long that took to figure out.

What I would have done differently:

I would have added more scores along with negative scores as well. It would be smart to test all scenarios. I would have liked to test different cases where there might be a 1 instead of 1.0 and whether a negative score affects the end result etc.

How can I apply this concept in the future?

Junit testing will be very key in the future. It will allow us to test our end results given specific requirements and situations. It allows us to make sure that we don't miss any type of situation where the code would produce incorrect results.

Anything else:

I was very comfortable with Junits since we used them a lot in 203. We used them for pretty much every assignment and allowed us to make sure we were headed on the right track.

Exceptions Lab 1B:

What I learned:

I learned that in the try block of a driver, we expect something inside to throw an exception. From your sample write up, I did learn/refresh myself about the next() and nextlLine() and which does which. I always used nextLine() since it covered all my ends but never really understood why. Also, I learned how to properly throw exceptions along with creating an exception class that can accept different error messages.

What I would have done differently:

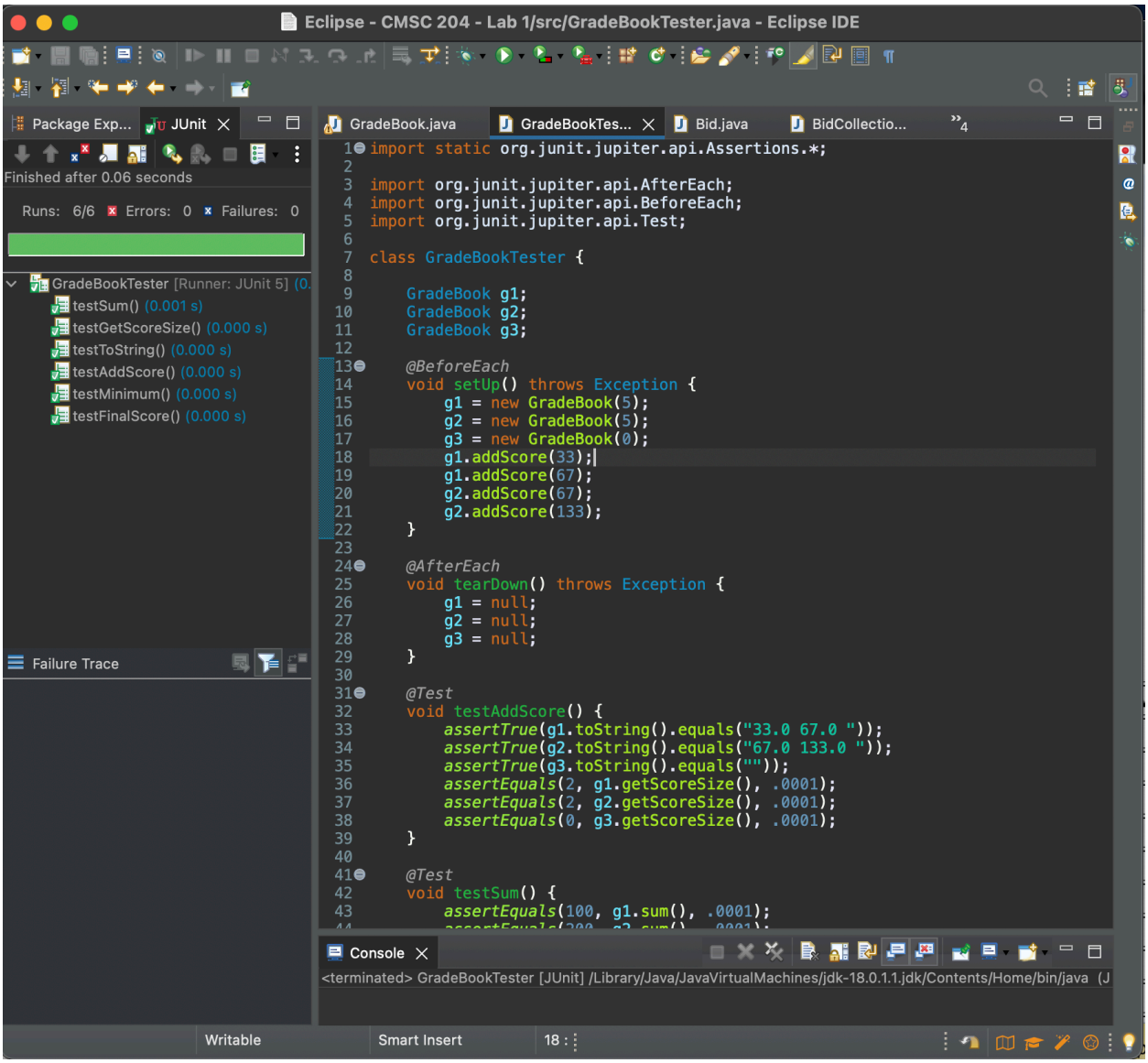
I agree with what you said about not needing to create a separate class. Although we really don't need the separate class, it's nice to break things up to simplify them. I'm glad we did it this way since I didn't every think to create an exception class where you can have different message per each scenario.

How could I apply this concept in the future?

Exceptions allow us to catch mistakes and errors that happen in our code. It allows us to catch user mistakes and allows the program to adjust and react accordingly. Combined with Junits, both allow us to make sure we are writing correct, concise code.

Anything else:

This was a nice refresher on char searching and using while loops. It was also a nice refresher on try and catch along with throwing exceptions.



Eclipse - CMSC 204 - Lab 1B/src/SocSecProcessor.java - Eclipse IDE

Package Exp... | JUnit | GradeBookTes... | SocSecProces... | SocSecExcept...

- CMSC 204 - Lab 1
  - src
    - (default package)
      - GradeBook.java
      - GradeBookTester.java
  - JRE System Library [Java SE 18.0.1.
  - JUnit 5
- CMSC 204 - Lab 1B
  - JRE System Library [Java SE 18.0.1.
  - src
    - (default package)
      - SocSecException.java
      - SocSecProcessor.java
- Test

```java
import java.util.Scanner;

public class SocSecProcessor {

    public static void main(String[] args) {
        Scanner entry = new Scanner(System.in);
        String userName = "";
        String userSSN = "";
        String userReply = "";
        char charReply = 'y';

        while (charReply == 'y' || charReply == 'Y') {
            try {
                System.out.print("Name? ");
                userName = entry.nextLine();
                System.out.print("SSN? ");
                userSSN = entry.nextLine();

                if (isValid(userSSN))
                    System.out.println(userName + " " + userSSN + " is valid")
            }

            catch (SocSecException e) {
                System.out.println(e.getMessage());
            }
            System.out.print("Continue? ");
```

Console

```
<terminated> SocSecProcessor [Java Application] /Library/Java/JavaVirtualMachines/jdk-18.0.1.1.jdk/Contents/Home
Name? Sam Sly
SSN? 333-00-999
Invalid the social security number, wrong number of characters
Continue? y
Name? George Washington
SSN? 123-45-6789
George Washington  123-45-6789 is valid
Continue? Y
Name? Dudley Doright
SSN? 222-00-999o
Invalid the social security number, contains a character that is not a digit
Continue? y
Name? Jane Doe
SSN? 333-333-333
Invalid the social security number, dashes at wrong positions
Continue? n
```