CMSC131 Fall 2021 (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/)

Project #1, Java Basics/Conditionals (Due Wed, Sep 15, 11:55 pm)

Objectives

This project will allow you to practice Java's expressions, input/output, and conditionals. In addition, you will learn how to use course tools (e.g., submit server).

Please, read this project description twice, as it defines several of the class rules we will use this semester.

Overview

You need to implement three programs: **Area, Divisible, and ColorGenerator**. For this project you can work with classmates, but you may not exchange code. Make sure you add the name of classmates you work with to the top of the Area.java file (in comments).

As specified in the syllabus, projects have a good faith attemnpt. A good faith attempt represents the minimum amount of work you have to complete for a project, otherwise you will get an automatic F in the course. If you are having problems completing the good faith attempt, TAs will help you. For this project, the good faith attempt is passing all public tests. Requirements and deadlines for good faith attempts in this class will be posted at Good Faith Attempts (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/goodFaithAttempt.shtml). A good faith attempt has a deadline different than the project deadline (usually a week after the project is due).

Suppose you finished this project and have a project score of 88 pts, but failed one of the public tests. Even though you have a grade of 88/100 in the project, you need to fix your code so you pass all the public tests, otherwise you will get an F in the class. Your score in the project will not change once you fix the code to pass the public tests. If you have questions good faith attempts post a question in Piazza.

Grading

- (52%) Public Tests
- (48%) Release Tests

Which Submission Gets Graded

In this course we grade the highest scoring submission in the submit server after a late penalty has been applied (in case you have a late submission). **The submission selected does not take into account any documentation/additional requirements,** therefore you need to complete documentation/additional requirements as you complete your work. If you don't, you may get a low score. For example, you could have one submission that has 70 pts of tests points and no documentation, and a second one with 60 pts of tests points and documentation (20 pts). We will grade the first one and not the second (you will get a score of 70 instead of 80) as we select from the submit server the one with the highest tests score.

Public, Release and Secret Tests

In this course exercises/projects have three kinds of tests: public, release and secret. Read the information you will find at Test types (http://www.cs.umd.edu/~nelson/classes/resources/testtypes/) before you continue.

Code Distribution

The project's code distribution is available at JavaBasicsCond.zip (https://www.cs.umd.edu/class/fall2021/cmsc131-01XX-03XX/prot/projects/zipFiles/JavaBasicsCond.zip). **You need the same username/password you use to download lecture slides**. Download the code distribution and import it as you have imported our class examples. The code distribution provides you with the following:

- **programs** - A package (folder) where you will find shells for three programs (Area, Divisible, ColorGenerator) that you need to implement.
- **tests** - A package (folder) where you will find public tests.
- **expectedResults** - A folder that has the expected results for each of the public test.
- **results** - A folder that the results generated by your code for each of the public tests.

Specifications

You need to implement three programs: Area, Divisible, and ColorGenerator. We have already provided shells (files with just the main method) in the **programs** package. **For this project you are expected to use good variable names and good indentation, but we will not be grading for style (your score will be based on tests scores)**.

1. **Area Program** - Write a program that will compute the area of a triangle. The program will read the base and height values and display the area. The following is an example of running the program. The example illustrates the messages used to read data and to display the area.

```
Enter base: 7
Enter height: 5
Area is: 17.5
```

2. **Divisible Program** - Write a program that reads two **integer** values x and y, computes the remainder, and determines whether x is divisible by y. The following are examples of running the program. The examples illustrate the messages to use to read data and the messages to display when x is divisible by y, and when it is not divisible by y.

```
Enter x: 8
Enter y: 3
Remainder: 2
8 is NOT divisible by 3

Enter x: 8
Enter y: 4
Remainder: 0
8 is divisible by 4
```

3. **ColorGenerator Program** - Write a program that computes a CSS color. We define a CSS color as a string that starts with a # character and it is followed by three pairs of 2 characters. The possible pairs of two characters will be 00 or FF. A color is defined by two characters for red, two characters for green, and two characters for blue. For example, #FF0000 is just red, whereas #00FFFF is the combination of green and

blue. Your program will ask users whether they want to have red as part of the CSS color. If the user answers "Yes", FF will be used for the red component; otherwise 00 will be used. The program will then ask the user whether they want both green and blue in the CSS color. If the user answers "Yes", FFFF will be used for green and blue; otherwise 0000 will be used. As you can see not all possible color choices can be generated (e.g., #FF00FF). The following are examples of running the program. The examples illustrate the messages to use when reading data and displaying the result. Notice that in addition to use "Yes", users can use "Yeah" to accept a choice. Any other value entered is considered a "No" answer.

```
Do you want red? (Yes/Yeah/No): Yes
Do you want green and blue? (Yes/Yeah/No): No
Final Color: #FF0000

Do you want red? (Yes/Yeah/No): No
Do you want green and blue? (Yes/Yeah/No): Yes
Final Color: #00FFFF
```

Requirements

- **Do not post any kind of code in Piazza.**
- **Before you post a question in Piazza, make sure you read the project description again. Often students post questions we have already addressed in the description.**
- Every project will have a Piazza folder associated with it. The first message in that folder will be a message that summarizes any clarifications we have made about the project. **You need to check these clarifications often**. If you have a question about the project, post a message in the folder we have created for the project.
- **Verify that your project passes the submit server public and release tests (https://submit.cs.umd.edu/ (https://submit.cs.umd.edu/fall2021)). Your score on the project will be based on results from the submit server and NOT from your results in Eclipse.**
- You can see results of public tests in the submit server or by executing in Eclipse the tests in the file **PublicTests.java**. To run public tests, right-click on the file PublicTests.java and select "Run As" → "JUnit Test". On the left side of the Eclipse window you will see the results. If you see a green bar, you have passed all tests; otherwise you are failing at least one test.
- **You can only release test your project once you have passed all public tests.** For this project you have 3 tokens in a 22-hour period. To release test your project:
    - In the submit server, select the "view" option for the project.
    - Select "perform release".
    - If you have any tokens left you will be able to click OK and see the results.
    - If you pass a test you will see a green square; otherwise you will see a brown square.
- Do not use System.exit(0) in your programs.

Comparing Files In Eclipse

In this project, when you run a public test, a file (with the suffix "Results.txt") that has the results of running the test with your code, will be generated in the **results** folder. For example, for the first public test, the file will be named pub1AreaResults.txt. We have left the expected results for tests in files with the suffix "ExpectedResults.txt" (e.g., pub1AreaExpectedResults.txt) in the **expectedResults** folder. If you are not passing a test, you can compare the results of your code (e.g., pub1AreaResults.txt) against the expected results (e.g., pub1AreaExpectedResults.txt). In Eclipse you can compare two files by selecting the two files, right-clicking and selecting "Compare With"→"Each Other". Practice comparing files as it might help you in this and future projects.

Submission

- Submit your project from Eclipse (within the Java perspective) by right-clicking the project folder and selecting "Submit Project". When it asks for your ID and password, be sure to use your University of Maryland "Directory ID" (e.g., terps, NOT your number (e.g., 12346789)) and the corresponding password. You may submit as many times as you want. We will always grade the submission that scores the highest on the submit server as defined in the syllabus. **If you do not see the "Submit Project" option is because you have not installed the CS Course Management Plugin as described in the installation section of the Eclipse tutorial (Eclipse Tutorial (http://www.cs.umd.edu/~nelson/eclipse/))**.
- The project's deadline is 11:55 pm, even though you will see 11:56 pm in the submit server. We need to use 11:56 pm as a project submitted exactly at 11:55 pm will be considered late by the submit server.
- You can submit your project 24-hours late, with a penalty of 12 pts. After 24-hours late, you will not receive any credit for the project (0 grade), but you still need to satisfy the good faith attempt.
- It takes time for a project to be tested in the submit server, especially the day the project is due. Keep this mind.
- **You need to submit often so you have a backup copy in the submit server.**
- We do not use CVS in this course. If you don't know what CVS is, don't worry about it.
- **If you write the project from scratch, without using the provided zip file, you will not be able to submit your work.**
- You should be submitting your project using Eclipse's "Submit Project" option. Uploading a zip file to the submit server should only used for emergencies.

Other

- When a file is created in Eclipse, you may need to refresh the Eclipse project to see the file. You should enable automatic file refresh in Eclipse. Additional information can be found at http://www.cs.umd.edu/~nelson/eclipse/other/#automatic-refresh (http://www.cs.umd.edu/~nelson/eclipse/other/#automatic-refresh)
- Only create one Scanner object in a program (e.g., one new Scanner(System.in) in Area.java
- We cannot provide additional information about release tests.
- If you are not passing release tests:
  - **Make sure you don't have spelling errors in messages the program generates. Even a small typo will make a test fail.** Spaces in a line are not an issue; extra blank lines may generate problems.
  - Carefully read the description, making sure your code handles the cases described in the description.
  - Read the project clarifications we have posted. Some of them can help you pass the test(s).
  - There is nothing tricky about release tests; they are based on the information we have provided in the project description and in the clarifications we have posted in Piazza.
- **If you hard code (i.e., use System.out.println() statements to generate expected output for a test) you will not receive credit for this project. During grading, TAs will look at your code and if you have hard coded any of your code, you will receive a 0 in the project.**
- You should have received a message indicating you are part of Piazza. If not, contact your lab / discussion TA.

Videos

The following videos illustrates several tasks associated with this project.

- Running Eclipse Public Tests (https://umd.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=c3ad9183-e7e0-4b61-9dbc-ad9d0141f69c)
- Submitting a Project (https://umd.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=28d9cf9e-9cd5-49ad-ab4d-ad9d0142f713)

- Submit Server Tests (Release-Testing) (https://umd.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=62efed1d-714a-448a-ae56-ad9d01450dc7)
- Refreshing Eclipse Project (https://umd.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=ef5a4117-2bb3-486d-9d5c-ad9d0149084b)

Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the University's Office of Student Conduct (https://www.studentconduct.umd.edu/).

Web Accessibility (https://www.umd.edu/web-accessibility/)