

1 Introduction

As it states on the syllabus, projects are to be done on your own and as with all assignments the sharing of files and code is strictly prohibited and constitutes an act of Academic Misconduct. Furthermore, the use of any electronic medium, such as code repositories, forums, blogs, message boards, email, etc. is strictly prohibited and constitutes an act of Academic Misconduct. You are not to discuss the project with anyone other than myself. You may use your class notes, solutions to the exams, labs, and exercises that are posted on the MyClasses site for this class, and the two textbooks being used for the course.

As usual, you will submit all your work through the MyClasses page for this class, under Project #1. Make sure you do the formatting Shift+Ctrl+F before you submit your work. For this project you will submit the following.

1. The Java code file for the program.
2. A Microsoft Word, LibreOffice Writer, or text file with at least three complete runs of the program.

For this project you will have a choice of program to write, either an arithmetic quiz program or a dice game program. You may do either one but you may not submit both, only one will be graded.

2 An Arithmetic Quiz Program

Write a program that will administer a ten-question arithmetic quiz to the user. There can be addition, subtraction, multiplication, and division problems. The questions should be appropriate for elementary school students who are just beginning to learn arithmetic. For example, all the numbers should be integers. The number of digits for addition problems should be at most two. For multiplication problems, at least one of the numbers should be a one digit number. For subtraction, the answer should not be a negative number. For division, the answer should be an exact integer, so that a problem like $\frac{28}{5}$ would not be possible but one like $\frac{28}{7}$ would be fine.

For each of the ten problems, you should pick the kind of problem — addition, subtraction, multiplication, or division — at random. You should pick the numbers in the problem at random. So, for an addition problem $A + B$, you should choose A and B at random. Present the problem to the user and get the user's answer. Compute the correct answer and check the user's response. Give the user two chances to get each problem correct. If they get it right on the first try, they get full credit (10 points). If they get it right on the second try, they get half credit (5 points). If they don't get the correct answer in two tries, tell them the correct answer, and don't give them any points. At the end of the quiz, tell them how many points they got, out of a possible 100.

The program should have input and output that looks nice and makes it easy for the user to tell what is going on. It should be friendly to the user (This is for kids!). Specifically,

- For addition, the question layout should look like the examples below. The columns of the numbers should line up and be right justified. The question mark should always be below the units column.

$$\begin{array}{r} 58 \\ + 91 \\ \hline ? \end{array}$$

Input your answer:

$$\begin{array}{r} 26 \\ + 9 \\ \hline ? \end{array}$$

Input your answer:

- For subtraction, the question layout should look like the examples below. The columns of the numbers should line up and be right justified. The question mark should always be below the units column. Since we are not allowing negative answers for this, the larger number should be on top.

$$\begin{array}{r} 70 \\ - 25 \\ \hline ? \end{array}$$

Input your answer:

$$\begin{array}{r} 70 \\ - 5 \\ \hline ? \end{array}$$

Input your answer:

- For multiplication, the question layout should look like the examples below. The columns of the numbers should line up and be right justified. The question mark should always be below the units column. If one of the numbers is 2 digits in length and the other is one digit in length (remember, at least one of them must be a single digit) put the two digit number on top.

$$\begin{array}{r} 7 \\ \times 6 \\ \hline ? \end{array}$$

Input your answer:

$$\begin{array}{r} 17 \\ \times 6 \\ \hline ? \end{array}$$

Input your answer:

- For division, the question layout should look like the example below. Use fraction notation, with an equal sign and question mark to the right and vertically centered with the fraction. Remember that the answers are to be integers here, so the numerator must be evenly divisible by the denominator.

$$\begin{array}{r} 54 \\ - 2 \\ \hline 2 \end{array}$$

Input your answer:

A portion of a sample run is below.

Program Run
<pre>70 - 25 ----- ? Input your answer: 45 Correct! 58 + 91 ----- ? Input your answer: 139 Your answer is incorrect, try again. Input your answer: 149 Correct! 7 X 6 ----- ? Input your answer: 42 Correct! 49 X 4 ----- ? Input your answer: 166 Your answer is incorrect, try again. Input your answer: 186 Both of your answers were incorrect, the correct answer is 196 54 ---- = ? 2 Input your answer: 27 Correct! <<< Removed for Handout >>> Your final score is 75 out of 100.</pre>

3 Dice Game

For the dice game, there are two players who will play 5 rounds of dice. In each round, each player will roll 4 dice and they may roll the dice up to 3 times. The player can stop rolling at any point. The program should do the die roll, show the player the score on the roll and ask the player if they would like to roll again. If the player chooses to roll again then the program will do another roll of the 4 die. The player's score for a single round will be the value of the last roll they did. The total score for all 5 rounds is the sum of the roll values for each round. At the end of the game the program should display the final scores for the two players and determine the winner or if the game was a draw.

The scoring for a single roll is as follows:

1. A four of a kind, that is, all four dice have the same value (e.g. 5 5 5 5) the player receives 50 points.
2. A three of a kind, that is, three dice have the same value (e.g. 5 2 5 5) the player receives 40 points.
3. Two pair, that is, two dice have the same value and another two dice have the same value (e.g. 5 2 2 5) the player receives 40 points.
4. One pair, that is, two dice have the same value (e.g. 5 1 5 3) the player receives 20 points.
5. No matches, (e.g. 5 1 4 3) the player receives the sum of the values of the dice, so in this example 13 points.

When the user is asked if they wish to roll again the program will roll again if the response is either *y* or *Y* and it will not roll again if the response is either *n* or *N*. If the user inputs something else then the program should tell the user that it was an invalid input and ask for another input. For example,

```
Player 1 Roll: 2 5 2 2      Value: 40
Would you like to roll again? (Y/N): q
Invalid Selection
Would you like to roll again? (Y/N): n
```

If the user inputs more than one character then the program should take the first character that is not a space. For example, if the user inputs Yes the program should take this as the input *Y* and if the user inputs

...I_do_not_know_what_to_do...

the program should take this as the input *I*.

Below is an example run of an entire game, make your program output the same format as in the example.

Program Run

```
Round #1
-----
Player 1 Roll: 4 3 2 4      Value: 20
Would you like to roll again? (Y/N): y
Player 1 Roll: 6 5 1 6      Value: 20
Would you like to roll again? (Y/N): y
Player 1 Roll: 3 1 2 1      Value: 20

Player 2 Roll: 1 2 4 6      Value: 13
Would you like to roll again? (Y/N): y
Player 2 Roll: 2 5 4 2      Value: 20
Would you like to roll again? (Y/N): y
Player 2 Roll: 4 2 6 3      Value: 15

Round #2
-----
Player 1 Roll: 4 5 2 6      Value: 17
Would you like to roll again? (Y/N): y
Player 1 Roll: 6 6 4 5      Value: 20
Would you like to roll again? (Y/N): n

Player 2 Roll: 3 5 3 3      Value: 40
Would you like to roll again? (Y/N): n

Round #3
-----
Player 1 Roll: 6 4 2 6      Value: 20
Would you like to roll again? (Y/N): y
Player 1 Roll: 4 4 6 3      Value: 20
Would you like to roll again? (Y/N): y
Player 1 Roll: 5 2 4 1      Value: 12

Player 2 Roll: 3 5 1 6      Value: 15
Would you like to roll again? (Y/N): y
Player 2 Roll: 4 1 6 3      Value: 14
Would you like to roll again? (Y/N): y
Player 2 Roll: 6 2 5 2      Value: 20

Round #4
-----
Player 1 Roll: 6 3 4 2      Value: 15
Would you like to roll again? (Y/N): y
Player 1 Roll: 6 6 3 2      Value: 20
Would you like to roll again? (Y/N): n

Player 2 Roll: 5 2 1 3      Value: 11
Would you like to roll again? (Y/N): y
Player 2 Roll: 3 5 4 2      Value: 14
Would you like to roll again? (Y/N): y
Player 2 Roll: 5 3 1 6      Value: 15

Round #5
-----
Player 1 Roll: 1 4 3 5      Value: 13
Would you like to roll again? (Y/N): y
Player 1 Roll: 4 1 3 1      Value: 20
Would you like to roll again? (Y/N): y
Player 1 Roll: 5 1 3 5      Value: 20

Player 2 Roll: 3 6 2 2      Value: 20
Would you like to roll again? (Y/N): y
Player 2 Roll: 6 6 1 6      Value: 40
Would you like to roll again? (Y/N): n

Final Score
-----
Player 1: 92
Player 2: 130

Player 2 Wins
```

4 Grading

The program itself should, of course, be nicely formatted and commented and should follow all the other rules of good programming style. Use the built-in formatting tool in Eclipse and put in some vertical white space to aid in the readability but don't over do it. Variable names should be representative of their purpose. As always, there must be our standard header comment. All variables must have a comment to their use and major blocks of code should contain brief but descriptive comments to their function.

The grading of the project will take two forms, a sample run and an inspection of the code. If the program does not run you will receive a zero for that portion. So even if the program is not complete you will get a better grade for a partial program that runs verses a program that does not run. The run portion of the grading will test the user interface for usability and conforming to the specifications I have outlined above. The code inspection portion of the grade will involve commenting, readability, correct indentation, variable names, structure and style, correctness, and conforming to specifications.