COSC 1046 – Fall 2016 - Assignment 3

This assignment is designed to give you some practice and experience writing Java applications using methods. Each question specifies the name that each file/application should have. Please follow this naming convention. When you have completed the assignment compress all of your files into a single archive using Windows (Right Click folder -> Send To -> Compressed Folder) or OS X (Right Click folder -> Compress). Using a 3rd party compression utility such as WinRAR or 7zip may render your files unreadable and un-markable. Submit a single compressed file to D2L. You can resubmit your files as many times as you would like up to the due date and time.

Be sure to include your name and a brief description of your program (as comments) at the top of each file. Pay attention to using good variable names. If you have any questions, please check or post to the forum.

Submit solutions for these questions. Wherever applicable, do your best to reproduce my output exactly. In my sample **bold** indicates user-input.

If you work in partners, submit only one solution and make sure that both partners' names are in ALL files. If only one name appears, only one person will get the grade. No exceptions.

A word in general on assignment grading (for future assignments): If your program produces the displayed output and meets the criteria specified in the question you should expect to receive full marks. Deductions are taken when there are deviations in the output – small deductions for small deviations (calculation error, improper formatting, etc.) while larger deductions are taken for larger deviations (missing output, substantially incorrect values, etc.). Express your creativity in your code, not in your output. Also, deductions will be taken for:

- lack of comments
- poor variable names
- programs that don't run at all (large deduction), so make sure your program runs, even if it is not complete.

If you have any questions or concerns about the grading scheme before you submit an assignment, please post to the forum and ask for clarification. If you have concerns about the grading you've received on an assignment, please email me and I will review the grading form.

Do all of the questions to ensure that you are practicing all concepts but submit solutions to questions 1, 2 and 3 ONLY for grading. It is assumed that you are doing all of the questions and some of them may be referenced in future assignments. Solutions will be provided only for required questions.

Question 1. (question1.java) (20 marks) Write a single class that has the methods:

```
public static double sphereVolume(double r)
public static double sphereSurface(double r)
public static double coneVolume(double r, double h)
public static double coneSurface(double r, double h)
```

Then write a `main()` method that prompts the user for values for `r` (radius) and `h` (height) and calls all of the other methods to displays the results. Your `main()` method takes care of all input collection and printing of the results – make use of the method arguments and return values. You do not need to do any error checking on the input values.

```
C:\Users\aaron\Desktop\java SphereAndCone
Enter a value for radius:1
Enter a value for height:1
Sphere Volume: 4.1887902047863905
Sphere Surface Area: 12.566370614359172
Cone Volume: 1.0471975511965976
Cone Surface Area: 7.584475591748159
```

```
C:\Users\aaron\Desktop\java SphereAndCone
Enter a value for radius:5.5
Enter a value for height:12.2
Sphere Volume: 696.9099703213358
Sphere Surface Area: 380.132711084365
Cone Volume: 386.46825626910436
Cone Surface Area: 326.26533476656743
```

Question 2. (`question2.java`) (20 marks) Write a class that has the following methods:

isPrime: This method takes a single positive integer and checks to see if it is a prime number. The method returns `true` or `false` (`boolean`, not String). Use your prime number solution from A2 to help you here.

vowels: This method takes a single String argument and returns the number of vowels (a, e, i, o or u) contained in the String.

compoundInterest: This method takes in all of the necessary arguments to compute and return the future value (*futureValue*) of an investment given the initial investment (*initialValue*), an interest rate (*i*) and a number of time periods (*n*).

$$futureValue = initialValue(1 + i)^n$$

factorial: This method takes a single integer argument (say *n*) a single integer. Use a `for` loop to compute the factorial of *n*. Return the computed factorial. Recall:

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

Write a `main()` method that coordinates all of the input and output – prompt the user for the appropriate values, call the appropriate methods and display the appropriate output. Error checking on the values is optional (you may assume the user puts in good values).

(5 marks) Add the following method and include an appropriate call in you main() method.

eEstimate: This method takes a single integer argument (say n) and returns a single double value. Once you have completed your factorial function, use it to compute an estimate of the mathematical constant E according to the following formula. Use a `for` loop that runs *n* times and sums the factors to produce your e-estimate. Try this on paper – the value of your e-estimate will increase in accuracy for a higher value of *n* (the argument passed into your function and used by your loop). Hint: This method should call your factorial method (see above).

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} \ldots + \frac{1}{n!}$$

Check the A3 discussion thread for sample output (sometime before the assignment is due).

Question 3. (20 marks) Write a simple press-your-luck dice game. The game works as follows:

- Player 1 starts with a score of zero and rolls two dice.
  - If the player rolls 7, 2, or 12 their turn is over and they lose.
  - If the player rolls any other sum it is added to their total score.
  - The player can stop rolling at any time – remember if their last roll was a 7, 2 or 12 they lose and Player 2 wins.
- After Player 1 stops rolling, Player 2 rolls the two dice
  - Player 2 continues to roll the dice until they roll a 7, 2 or 12 (Player 2 loses) or they have a total higher than Player 1 (Player 2 wins).

Your solution should make use of methods with the following headers:

- `public static int rollDie()`
  - return a random valid die value between 1 and 6
- `public static int newScore(int oldScore, int rollValue)`
  - determine if the current roll should be added to the score. If the player rolled 7, 2 or 12 return -1 to show that the player just lost. Otherwise return the updated score.
- `public static int whoGoesFirst()`
  - randomly return 1 if player 1 goes first or 2 if player 2 goes first
- `public static String winner(int playerNumber)`
  - print a message indicating who won the game
- `public static char keepRolling()`
  - A method that prompts the player to ask if they want to keep rolling. Return y or n as needed. Note: Player 2 ~could~ stop rolling if they wanted to but if they haven't achieved a higher score they will lose.
- `main()`
  - this is your main method that coordinates all of the other methods as necessary. Collect all input and produce all output from this method.

Your game should be able to play a complete human vs. human game.

No sample output. Be creative (but kind to your users).

(5 marks) You should be able to play human vs. computer AND play multiple games without quitting the program AND keep track of how many games each player has won.

Question 4. Hey, a science question! The resistance in a strand of wire can be computed using:

$$\boldsymbol{R = \frac{4\rho L}{\pi d^2}}$$

where ρ is the resistivity of the wire, L is the length and d is the diameter. The diameter of wire can be determined given the American wire gauge number:

$$\boldsymbol{d = 0.127 \times 92^{\frac{36-}{39}}}$$

where n is the AWG number.

An appropriate header for the first method might be:

```
public static double resistance(double resistivity, double length, double diameter)
```

and for the second method:

```
public static double diameter(int n)
```

Write a program that prompts the user for the AWG number (integer) and the length of a piece of wire (double) and the wire's resistivity. First, compute the e diameter of the wire from the AWG number using the appropriate method. Save this value and pass it, along with the collected length and resistivity to the resistance method to compute the desired result. Test your program with the values shown below.

Note: copper wire a resistivity (ρ) of 1.68 × 10^{-8} Ω/m) and aluminum wire has a resistivity (ρ) of 2.82 × 10^{-8} Ω/m

Remember, write a `main()` method to collect the necessary input, to call the other methods and collect their returned values and then to display the output.

```
C:\cosc1046\a3\>java question2
java question1
Enter wire gauge:22
Enter wire length (in m.):10
Enter wire resistivity:1.68e-8
Diameter of the wire is: 0.6438032984904795
Resistance in copper is: 5.160752189379868E-7 ohms
Resistance in aluminum is: 8.662691175030492E-7 ohms

C:\cosc1046\a3\>java question2
Enter wire gauge:2
Enter wire length (in m.):1825.23
Enter wire resistivity:2.82e-8
Diameter of the wire is: 6.543707496202785
Resistance in copper is: 9.117782800853958E-7 ohms
Resistance in aluminum is: 1.5304849701433429E-6 ohms
```

(extra) Instead of calling the two methods separately, change the header of the first method to:

```
public static double resistance(double resistivity, double length, int n)
```

and call the diameter method using method nesting. Also, include a method that continually prompts the user to enter positive values for the collected input. Hint: One method should be enough and you should check each input separately as opposed to trying to check all three at once.

Question 5. Write a class that has the following methods:

power: This method takes two integers as arguments, one representing the base and another representing the exponent. The method then uses a `for` loop to compute and return the result of base^{exponent}. Eg. $2^3=8$.

passwordCheck: This method takes a single String argument and returns a boolean value. The purpose of this method is to test the strength of a password. To be considered strong a password must have the following properties:

-at least 8 characters in length
-at least one uppercase and one lowercase letter
-at least one digit

If the string parameter passed to the method passes all of these tests, return true, else return false. Use your main method to report whether or not the password is strong.

fix: Matlab™ has a great method called *fix* which returns the decimal portion of a floating point number. Write a method that takes in a double and returns only the decimal portion. Eg. 3.14159 passed in would return 0.14159. There may be some rounding issues as you test your method. Don't worry about that too much but if you want to try and solve that, that's ok too.

Finally, write a `main()` method that prompts the user to enter inputs appropriate for testing your methods. Remember, to do all input and output in your `main()` method. Pass the input to your methods and collect the results using assignment statements. You don't need to do any input checking. Assume the user enters appropriate values. I suggest your write and test your methods one at a time. You can get part marks for the methods that work.

halfLife: Radioactive decay of materials can be modeled by the equation shown below. Write a method that takes in all the necessary arguments to compute and then return the amount of material (A_t) after it has decayed for a period of time (t) given a half-life value (h).

$$\boldsymbol{A_t = A_0 e^{-t\left(\frac{ln2}{h}\right)}}$$

Finally, write a `main()` method that prompts the user to enter inputs appropriate for testing your methods. Remember, to do all input and output in your `main()` method. Pass the input to your methods and collect the results using assignment statements. You don't need to do any input checking. Assume the user enters appropriate values. I suggest your write and test your methods one at a time. You can get part marks for the methods that work.

Question 6. Write a game called Cursed Gold. The object of the game is to make your opponent select the last gold coin from the pile. There are initially 16 gold coins in the pile and you and your opponent take turns selecting 1,2 or 3 pieces of gold. The person who has to take the last piece from the pile loses.

The idea here is to create three functions that represent different strategies. Two of your methods (named playerSimple() and playerSmart()) are completely automated – think artificial intelligence. They take a single argument – the number of coins currently in the pile. Your functions use this information to decide how to strategize. One of them can be very simple and do the same thing every time but the other function should be a bit more … intelligent. Random selections are also valid but will only work to a point. These functions print (yes, printing in a function is allowed here) how many coins were removed and they return how many coins are left – this is important as this number is needed by the next "player".

The third function (playerHuman()) is a human player. This means that it still takes a single argument – the number of coins remaining but it then prompts the user for input (1,2 and 3 are the only valid inputs) and returns the number of gold pieces remaining.

All three of the functions can work in different ways but ultimately, all three will have to deal with the pile of coins in a special way when it is down to three or less – keep this in mind.

You'll need a main function that initializes the pile to 16 gold pieces. It will also be responsible for looping over the player turns, for printing out the number of pieces remaining after each turn and for determining who won. It also determines whether or not the second player is human and calls the appropriate functions.

A few hints from my solution (take them or leave them):

-I have a function called checkWinner() that determines if a winner exists (only 1 coin left!!) and stops the loop.
-In the output below, my player 1 is my smarter computer player while player 2 is my simpler computer player – only slightly but there is a difference.
-My solution has a lot of error checking – making sure I don't take too many coins for example – even in the computer players.

Output from the two computer players playing (no user input):

```
Welcome to Cursed Gold!.  Here be the rules:
-Ye can only take 1, 2 or 3 gold coins on yer turn.
-The poor soul that takes the last coin walks the plank (loses).
---------------------------------------------

Be playarr 1 human?: (y/n)n
Be playarr 2 human?: (y/n)n

Arrr.  I be stacking up a pile of coins 16 high.

Player 1:Taking 1 gold coin(s)
Thar be 15 gold left matey.
Player 2:Taking 1 gold coin(s)
Thar be 14 gold left matey.
-------------------------------
Player 1:Taking 3 gold coin(s)
Thar be 11 gold left matey.
Player 2:Taking 1 gold coin(s)
Thar be 10 gold left matey.
-------------------------------
Player 1:Taking 3 gold coin(s)
Thar be 7 gold left matey.
Player 2:Taking 1 gold coin(s)
Thar be 6 gold left matey.
-------------------------------
Player 1:Taking 1 gold coin(s)
Thar be 5 gold left matey.
Player 2:Taking 1 gold coin(s)
Thar be 4 gold left matey.
-------------------------------
Player 1:Taking 3 gold coin(s)
Thar be 1 gold left matey.
Winner is player 1
```