RMIT UNIVERSITY

School of Science

COSC1284 Programming Techniques

Assignment 3

	Assessment Type: **Individual assignment; no group work.** *Submit online via Canvas→Assignments→Programming Assignment #3. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.*
	Due date: 9 am, Thursday 22nd October 2020. Deadlines will not be advanced, but they may be extended **under exceptional circumstances**. Please check Canvas→Syllabus or via *Canvas→Assignments→Programming Assignment #3* for the most up to date information.
	Weighting: **30 marks**

1. Overview

A local game developer, **The League of Über Nerds**, has commissioned the prototype of a random character generator for their latest computer role-playing game, **World of Kitschcraft**. This early prototype will only model a subset of attributes for the game characters, namely:

- Name (The game character's name)
- Hit points (The game character's vitality/energy level)
- Strength (The game character's strength)
- Intelligence (The game character's intelligence)
- Wisdom (The game character's wisdom)

There are exactly three game character types, namely:

- Fighter
- Wizard
- Cleric

Those character types are implemented in a **Fighter**, **Wizard** and **Cleric** class, derived from a **GameCharacter** base class. Each derived class overrides two methods:

- A **void randGen(boolean areHitPointsMaxed)** method. This method randomly generates the values of the game character's attributes. However, when the boolean parameter maxHitPoints is set to **true**, it assigns a predetermined maximum value of hit points for any game character type.

- A **String display()** method. This method returns a game character's attributes similarly to the output below:

```
Name:         Felonious
Class:        Wizard
Hit Points:   6
Strength:     12
Intelligence: 17
Wisdom:       16
```

Question 1.1 [5 marks]

Implement a console-based menu in the main method of the class **GameCharGen**, included in the skeleton code provided to you. The menu will carry out the commands illustrated below:

```
*** WoK GameCharGen Menu***
 Create Wizard          WZ
 Create Fighter         FG
 Create Cleric          CL
 Exit Program           EX
 Enter selection:
```

You must implement the menu as shown. Do not change the options or the inputs for selecting them!

The implementation of this question must perform the following tasks:

CC1. Create a character based on the user's selection.
CC2. Randomly generate the values for the character by utilising the provided method 'randGen'.
The maxHitPoints should be set to false if the character is a Fighter, otherwise the maxHitPoints should be set to true.
CC3. Add the character to the programs array of characters as declared in the class GameCharGenModel class.
CC4. Invoke the display method implemented in the GameCharGenModel class and print its return value to the console.

The **Exit Program** option should simply exit the program while, if the user types an input that does not correspond to any of the menu options listed above, the program should display an error message (e.g., **"Unrecognised command."**) without exiting.

Note: The letters on the right represent what the user must type to use that feature. For instance, to create a new Wizard, the user must input 'WZ'. The solution should be case insensitive therefore the user should be able to also enter either 'wz', 'Wz' or 'wZ', etc

Question 2: Wisest Character [10 Marks]

Now that you have a new array of randomly generated game characters, implement the method specified below to return a summary of the wisest of all the characters.

Implement the **public String[] wisestOfThemAll()** method provided in the class **GameCharGenModel**.
This method returns a summary of the character with the most wisdom points.

The implementation of this method must abide by these constraints:

WC1. If no characters are in the array the algorithm is not executed and a NoGameCharacterException is generated and thrown. This exception class has already been implemented in the skeleton code provided to you.
WC2. The index of the wisest character is found. If two or more characters share the wisest value then the first character in the array should be selected.
WC3. The name of the selected character is identified.
WC4. The wisdom value of the selected character is identified.
WC5. The average of all the characters wisdom values is calculated.
WC6. The four data points from WC2 - WC5 are returned as a String array

The summary data should be returned as an array of String values in the following format.

Index 0: holds the index value where that character can be found in the game characters array of the GameCharGenModel class.
Index 1: holds the name of the character.
Index 2: holds the wisdom points for that character
Index 3: holds the average wisdom points for all the characters.
This value should contain exactly two decimal places.

Question 3: GameCharGen - Wiser than strong or Strong as Wise [10 Marks]

Implement the **void wiserThanStrongOrStrongAsWise()** method provided in the class **GameCharGenModel**, which **filters and sorts** the original array of game characters based on the characteristic that the character is wiser than they are strong or just as wise as they are strong. Note: this is a void method and does not return any data. After executing this function you will need to execute the display characters function to see the results of this operation.

The implementation of this method must abide by these constraints:

WS1. If no characters are in the array the algorithm is not executed and a NoGameCharacterException is generated and thrown. This exception class has already been implemented in the skeleton code provided to you.
WS2. All characters that have a strength value less than their wisdom value are removed from the original array.
WS3. All characters that have a wisdom value greater than or equal to their strength value remain in the original array.
WS4. The remaining characters in the original array are sorted by the wisest first (descending order).

Question 4: GameCharGen - UpdatedMenu [5 Marks]

Add three menu options to the class **GameCharGen** as below:

```
*** WoK GameCharGen Menu***
 Create Wizard              WZ
 Create Fighter             FG
 Create Cleric              CL
 WisestOfThemAll            WC
 WiserThanStrongAsWise      WS
 Display Characters         DC
 Exit Program               EX
 Enter selection:
```

Note: The letters on the right represent what the user must type to use that feature. Once again, the solution should be case insensitive: the user should be able to also enter either 'wz', 'Wz' or 'wZ', etc.

The implementation of this question must abide by these constraints:

UM1. Execute the WisestOfThemAll method and print out a summary of the wisest character including all the details from the array returned.
UM 2. Execute the WiserThanStrongAsWise method.
UM 3. Execute the display method and print out the return value of this method to display all the current characters in the array.

This program will demonstrate the following key skills:

1. Creating a small program to demonstrate what you have learned as a developer.
2. Analysing a problem and developing an algorithm to solve the problem.
3. Converting an algorithm to computer code.
4. Debugging your code on test data sets.

Read the requirements thoroughly!

There are specific constraints that have been placed on you for this assignment to force you to work within defined parameters.
The ability to work within a pre-defined set of parameters is a fundamental skill required by any software developer.
You will also need to debug code on your own.
You are given marks on your ability to fulfil all requirements of this document.
Develop this assignment in an iterative fashion, as opposed to completing it in one sitting.
Any questions regarding this assignment must be asked via the relevant Canvas discussion forums **only** (no emails to teaching staff, please) in a general manner.
Questions about clarifying the specifications can be asked, but questions about how to write the code cannot be answered.

2. Assessment Criteria

This assignment will assess several skills:

1. Following coding conventions and behavioural requirements provided in this document and in the lessons.
2. Independently solving a problem by using programming concepts taught over the first eight weeks of the course.
3. Writing and debugging Java code independently.
4. Meeting deadlines.
5. Seeking clarifications from your teaching team, when needed, **via discussion forums**.
6. Creating a program by recalling concepts taught in class, understanding, and applying concepts relevant to the solution, analysing components of the problem, and evaluating different approaches.

3. Learning Outcomes

This assessment is relevant to the following Course Learning Outcomes (CLOs):

CLO1. Demonstrate (through small programming exercises) knowledge and skills with concepts of program design and acceptable coding standards.
CLO2. Use Java programming language as a vehicle to demonstrate good software development practices.
CLO3. Use arrays and control structures to demonstrate skills of basic algorithms and data structures.
CLO4. Apply knowledge of the basic principles of the object-oriented development process to the analysis and design of solutions for small scale problems.
CLO8. Demonstrate skills for self-directed learning.

4. Assessment details

Note: Please ensure that you have read sections 1-3 of this document before going further.
Your code must meet the requirements listed in the marking rubric in **Section 8**, also available on Canvas.

5. Submission format

Follow the instructions below to complete this assignment:

1. Download skeleton code for **Assignment 3 from Canvas**.
2. Unzip the archive you have downloaded to a folder on your computer.
3. Open the folder in **Visual Studio Code** selecting **File->Open Folder** (**File->Open** on Mac OS) and then picking your folder.
4. You will need to complete the code, in response to **Questions 1-4** by the due date**.**
5. Follow the instructions outlined in the comments of files **GameCharGen.java** and **GameCharGenModel.java** to complete the questions. These files are included in the skeleton code assigned to you.
6. Use Visual Studio Code as **your only editor** to complete the program as per the specification provided to you in the questions.
7. Submit your entire project, as a **.zip archive**, using the submission link on Canvas for this assignment.
8. It is **your responsibility** to download and check your submission before completing your assignment.
9. Marking will be entirely based on **your last submission in Canvas before the deadline**.
10. **Late submission penalties apply.**
    Please ensure that you submit in progress submissions during the assessment period to ensure that you have a valid submission in the event that your last submission is not successful.

    - For each 2 hour period after the submission deadline there is a 3 mark penalty.
    - If you submit after 09:00 a.m. on Thursday then you will receive a 3 mark penalty.
    - If you submit after 11:00 a.m. on Thursday then you will receive a 6 mark penalty.
    - No submissions will be accepted after 09:00 p.m. Thursday 22nd October.
    - Any extensions must go through the formal special consideration process.

    For example, if the last submission is made after 09:00 a.m. and before 11:00 a.m. on Thursday 22nd October, 2020 then a 3 mark deduction will apply.

    If the last submission is made after 011:00 a.m. and before 01:00 p.m. on Thursday 22nd October, 2020 then a 6 mark deduction will apply.

    Only the last submission will be marked.
    Previous submissions cannot be marked to avoid late penalties.

6. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods,
Provided a reference list of the publication details so your reader can locate the source if necessary.
This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students

For further information on our policies and procedures, please refer to the University website.
https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

7. Assessment declaration

When you submit work electronically, you agree to the assessment declaration.
https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration

Rubric on following pages in section 8

RMIT UNIVERSITY

8. Rubric/assessment criteria for marking

Code must be valid, runnable Java to be given a mark (code that cannot be compiled, pseudocode, incomplete Java code cannot be marked).
Run-time errors will incur up to a 50% penalty (run-time errors due to data type mismatches in inputs are acceptable).

Criteria	Ratings							Pts
C1 - Character Creation Menu CC1. Create a character based on the users selection. CC2. Randomly generate the values for the character by utilising the provided method 'randGen'. The maxHitPoints should be set to false if the character is a Fighter, otherwise the maxHitPoints should be set to true. CC3. Add the character to the program's array of characters as declared in the class GameCharGenModel. CC4. Invoke the display method implemented in the GameCharGenModel class and print its return value to the console.	**5.0 to >3.74 pts** **4 items met** 4 of the items listed in this criterion are functionally correct and no improvements were identified.	**3.74 to >2.5 pts** **3 items met** 3 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**2.5 to >1.75 pts** **2 items met** 2 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**1.75 to >0.0 pts** **1 item met** 1 of the items listed in this criterion is functionally correct and/or some improvements were identified.	**0.0 pts** **0 items met** 0 of the items listed in this criterion are functionally correct.			5.0 pts
Q2 - Wisest Character Summary WC1. If no characters are in the array the algorithm is not executed and a NoGameCharacterException is generated and thrown. This exception class has already been implemented in the skeleton code provided to you. WC2. The index of the wisest character is found. If two or more characters share the wisest value then the first character in the array should be selected. WC3. The name of the selected character is identified. WC4. The wisdom value of the selected character is identified. WC5. The average of all the characters wisdom values is calculated. WC6. The four data points from SW2 - SW5 are returned as a String array	**10.0 to >8.0 pts** **6 items met** 6 of the items listed in this criterion are functionally correct and no improvements were identified.	**8.0 to >6.0 pts** **5 items met** 5 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**6.0 to >4.0 pts** **4 items met** 4 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**4.0 to >2.0 pts** **3 items met** 3 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**2.0 to >1.0 pts** **2 items met** 2 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**1.0 to >0.0 pts** **1 item met** 1 of the items listed in this criterion is functionally correct and/or some improvements were identified.	**0.0 pts** **Did not complete** 0 of the items listed in this criterion are functionally correct.	10.0 pts

RMIT UNIVERSITY

C3 - WiserThanStrongAsWise WS1. If no characters are in the array the algorithm is not executed and a NoGameCharacterException is generated and thrown. This exception class has already been implemented in the skeleton code provided to you. WS2. All characters that have a strength value less than their wisdom value are removed from the array. WS3. All characters that have a wisdom value greater than or equal to their strength value remain in the array. WS4. The remaining characters in the array are sorted by the wisest first (descending order).	**10.0 to >8.0 pts** **4 items met** 4 of the items listed in this criterion are functionally correct.	**8.0 to >6.0 pts** **3 items met** 3 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**6.0 to >4.0 pts** **2 items met** 2 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**4.0 to >0.0 pts** **1 item met** 1 of the items listed in this criterion is functionally correct and/or some improvements were identified.	**0.0 pts** **Did not complete** 0 of the items listed in this criterion are functionally correct.	10.0 pts
C4 - Character Creation Menu UM1. Execute the WisestOfThemAll method and print out a summary of the wisest character including all the details from the array returned. UM 2. Execute the WiserThanStrongAsWise method. UM 3. Execute the display method and print out the return value of this method to display all the current characters in the array.	**5.0 to >3.0 pts** **3 items met** 3 of the items listed in this criterion are functionally correct.	**3.0 to >1.5 pts** **2 items met** 2 of the items listed in this criterion are functionally correct and/or some improvements were identified.	**1.5 to >0.0 pts** **1 item met** 1 of the items listed in this criterion is functionally correct and/or some improvements were identified.	**0.0 pts** **Did not complete** 0 of the items listed in this criterion are functionally correct.		5.0 pts
Penalties The following penalties will be applied. a) 6 marks for any compilation errors You must remove any compilation errors prior to submission. b) 3 marks for each runtime error that occurs such as ArrayIndexOutOfBounds, NullPointer or other unhandled exceptions.	**0.0 pts** **Full Marks**		**0.0 pts** **No Marks**			0.0 pts
						Total Points: 30.0