CO871 2019: The World of Zuul Framework

This assignment must be submitted through Moodle by **11.55pm, Tuesday Week 6 — 29 October 2019**. Make certain that, when unpacked, your files will compile and run in any environment (e.g. avoid absolute links to files that exist only on your machine).

Motivation for the task

The purpose of this task is to give you an opportunity to demonstrate what you have learned as a reflective, object-oriented programmer. At the heart of the assignment is a very badly constructed piece of code which you have to critique and refactor. You need to apply all you have learned about responsibility driven design. In particular, you should consider what design patterns are appropriate for this program.

Be very clear that this assignment requires a substantial amount of refactoring. *For example, solutions that retain the existing structure of the code will not achieve a pass mark*.

The background brief is deliberately under-specified, as is common with briefs from clients in the real world. *Note that the company has ambitions for future developments that your solution must support*.

- Solutions that permit future developments with only very limited changes to the code are likely to get good marks;
- Solutions that allow such developments *without any changes to the code that you have already written* will obtain excellent marks.

Peer assessment As part of the assessment, after the deadline has passed, you will be given a marking scheme and a set of solutions to mark (one of these will be your own). Peer assessment of other people's code has proved to be a very good way to develop understanding of good programming practices: it forces you to reflect on the code.

One-to-one feedback You will have a one-one feedback meeting with me to discuss your solution.

Note that to obtain credit for this assessment, you must complete *all* components: the critique and solution, peer assessment, and feedback meeting.

Background

For some unknown reason (1980s nostalgia?), the small games company that you work for has decided to produce a new adventure game, *The World of Zuul*. The company hopes that this will be the first of a series of products (some in the Zuul series, others quite different but all sharing the same underlying structure). It is therefore essential that the code produced is *easy to maintain* and to *extend*. They plan to release games in *single-player* and *multi-player* versions. They also intend to have some AI (artificial intelligence) — *computer-controlled characters* than can move through the game.

As team leader, you asked an intern from Parkwood Metropolitan University to produce an initial prototype with a simple command-line interface. The module page contains a link to a jar file of his prototype, `zuul-bad-extended.jar`. You save a copy of this file to your file space, read his code and despair: it is dreadful. It clearly cannot be extended in its present form, yet alone support any of your company's ambitions (see above). Clearly, you are going to have to refactor it before it meets all these goals.

Your tasks

There are four parts to this assignment. First, write a detailed report critiquing the prototype and explaining your improvements. Second, refactor the prototype and implement the missing features

listed below in order to remedy the prototype's shortcomings. Third, peer mark some solutions. Fourth, meet me for feedback. You must complete all of these tasks.

Read these instructions and the background very carefully. In particular, pay attention to the requirements and future plans of the company. Your solution *must* support all of them.

1 Refactor the game (80%)

1.1 Refactoring

Refactor the game to remove the flaws of the prototype. Some of these have been mentioned in the lectures or in *Objects First with Java*: you may implement these. **Your implementation should demonstrate good design throughout** and **your framework must support all the company's plans for future developments**. Your code must be professionally written and will be assessed for:

- correctness
- design (particular consideration will be given to cohesion, coupling, maintainability, adaptability)
- support for the company's ambitions for its products.
- appropriate use of language constructs
- style (commenting, indentation, etc.)

Note that solutions that retain the current structure of the solution will not get a pass mark.

1.2 New features

Add the following features from the company's wish-list.

- Support for more than one player.
- Support for computer-controlled characters.
- A solution that will allow new actions to be added efficiently and with minimal (or no) changes to code that you have already written.
- Support to *allow* different user interfaces to be added . Note that you do not have to *implement* a GUI or other new interface.

Note: You must support the command language in exactly its current form. You don't need to add any new game functionality (you will not get any extra marks) but you may do so if it helps to test your redesign.

2 Report (20%)

Reading and critiquing other people's code is a useful exercise. Your first task is to read the existing code, understand what it does, and where and why it does it badly. You may consult Chapter 7 of *Objects First with Java*, Barnes and Kölling, for ideas about good design. Bearing in mind the goals of the company as well as good software engineering practice, write a detailed report explaining how your refactoring has addressed the design and implementation flaws in the prototype and meets the company's goals. Ensure that you not only list each flaw but explain why it is problematic and how you have resolved it.

- Structure your report as a table, identifying the flaw, the bad consequences of the flaw, and how you have remedied it. You need only make a brief note for each case. E.g.

Flaw	Consequences	Solution

- Your report should be 1 or 2 pages of A4. Very short reports are unlikely to get full marks.

Suggested approach

Here is a suggestion for how you might tackle this problem. It is only a hint and you don't have to follow these steps or in this order. I strongly suggest that you write your report as you read and refactor the code in order to ensure that you do not omit anything from the report.

- Basic steps
    1. Read the code and include in your report a table of all the flaws you find and their consequences. Thoroughly test the existing code to find bugs. Add these to your report.
    2. Fix any 'bad smells' or other bad practice in the code — here you will be thinking about cohesion,
    3. coupling, maintainability, good programming practice, etc. Add these to your report.
    4. Fix the bugs you have found. Add these to your report no matter how trivial you may consider them.
- At this point, you should have some code that you can start to work with to meet the company's goals.
    5. Refactor the code to make it easy to add new actions. Explain this in your report.
    6. Refactor the code to support the company's other goals. Explain all your changes in your report.
- At this point, you should have a solution that is good enough to get at least a pass mark.
    7. Refactor the code to make it simple to provide different user interfaces (e.g. a GUI as well as a Command Line interface) in the future. Explain how you have done this in your report
    8. Can you write the code so that no changes at all need to be made to existing code in order to add a new action?

Submission and Assessment

You must submit the project by **11.55pm, Monday Week 7 — Tuesday Week 6 — 29 October 2019**. You must include:

1. Your PDF critique of the prototype, `critique.pdf`
2. The source code and Javadoc documentation for all your code. If you submit a Netbeans project, make sure that it has included the source code, and not just the .class files.

You must adhere to the submission requirements. Adhere to the user interface specified, i.e. do not change command names, or add a graphical user interface. However, you can make any other changes you deem appropriate, including adding, removing or renaming classes (but don't rename Main.java). **Submit the report as PDF using the name specified above.** If you ignore these requirements, it is likely that my test and printing scripts will not work and you will lose marks.

Late or non submission of coursework

The penalty for late or non submission of coursework is normally that a mark of zero is awarded for the missing piece of work and the final mark for the module is calculated accordingly.

Plagiarism and Duplication of Material

Senate has agreed the following definition of plagiarism: "Plagiarism is the act of repeating the ideas or discoveries of another as one's own. To copy sentences, phrases or even striking expressions without acknowledgement in a manner that may deceive the reader as to the source is plagiarism; to paraphrase in a manner that may deceive the reader is likewise plagiarism. Where such copying or close paraphrase has occurred the mere mention of the source in a bibliography will not be deemed sufficient acknowledgement; in each such instance it must be referred specifically to its source. Verbatim quotations must be directly acknowledged either in inverted commas or by indenting." The work you submit must be your own, except where its original author is clearly referenced. We reserve the right to run checks on all submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism. When you use other peoples' material, you must clearly indicate the source of the material using the Harvard style (see http://www.kent.ac.uk/uelt/ai/styleguides.html).

In addition, substantial amounts of verbatim or near verbatim cut-and-paste from web-based sources, course material and other resources will not be considered as evidence of your own understanding of the topics being examined.

The School publishes an on-line Plagiarism and Collaboration Frequently Asked Questions (FAQ) which is available at: http://www.cs.ukc.ac.uk/teaching/student/assessment/plagiarism.local

Work may be submitted to Turnitin for the identification of possible plagiarism. You can find out more about Turnitin at the following page:
http://www.kent.ac.uk/uelt/ai/students/usingturnitinsts.html#whatisTurnitin

Richard Jones, September 2019