University of Wolverhampton
Faculty of Science and Engineering
School of Mathematics and Computer Science

Module Assessment

Module	6CS002
Module Leader	Dr John Kanyaru
Semester	2
Year	2020-21
Assessment	Portfolio task 2
% of module mark	30%
Due Date	14 May 2021
Hand-in – what? Program code should be pushed to the GitLab repository that you were given access to in the first lecture. The work should be documented in Git commit comments.	Program code and refactoring checklist should be pushed to the GitLab repository that you were given access to in the first lecture. The refactoring work should be documented in Git commit comments.
Hand-in- where?	Program code should be pushed to the GitLab repository that you were given access to in the first lecture. The refactoring work should be documented in Git commit comments.
Pass mark	40% is required overall to pass the module. Each portfolio task has an indicative weighting. However, the final grade will not be calculated entirely mechanically. A degree of academic judgement will be used to assess how well you have met the learning outcomes overall. You must attempt and submit all portfolio tasks.
Method of retrieval	Resit portfolio tasks in the summer resit period. These will not necessarily be the same as the original tasks, but variants of them.
Feedback	Available during scheduled classes
Collection of marked work	From module tutors in scheduled classes.

Notes:

Assignment details: Task 2 - Refactoring

You must use the Git version control system to manage your source code. You should frequently check your work into your repository using explanatory comments to document your progress with the work. Your complete collection of commit logs will effectively form a learning journal that documents your progress with the module. It must be submitted as evidence of your learning. Each commit comment should describe what you have done since the last commit.

You should try to tackle as many refactorings as possible from the supplied checklist, but not necessarily in the order presented. At the end of each refactoring you should commit your code with an explanation of what you have done, which code is effected and relate it to underpinning theory, e.g. smells and principles. Before attempting complex refactorings you should put unit tests in place to enable the recognition of any regression bugs. To achieve an excellent grade you will need to decouple the user interface from the game logic, so that you can test the logic fully, i.e. you should write unit tests for extracted methods.

By regularly committing your code to your repository, it will not only document your work but also act as a backup so that you may rollback if you introduce defects into the program. For larger refactorings you may need to restructure your tests, which should be documented in your commit logs. You should ideally address all the problems documented in Task 1.

Submission of work

Your completed work for assignments must be handed in on or before the due date. ***You must keep a copy or backup of any assessed work that you submit. Failure to do so may result in your having to repeat that piece of work.***

Penalties for late submission of coursework
ANY late submission (without valid cause) will result in the grade 0% being allocated to the coursework.

Procedure for requesting extensions and mitigating circumstances
If you are unable to meet a deadline or attend an examination, and you have a valid reason, then you will need to make a request for an extension or mitigating circumstances through e-vision. Please contact the Students union or check the following web page for more details: http://www.wolvesunion.org/main/advice/academic/extenuating

Retrieval of Failure
A pass of 40% or above must be obtained in each of the components (but not necessarily in the elements). Compensation between elements within a component is allowed, but compensation between components is not.
Where a student fails a module they have the right to attempt the failed assessment(s) once, at the next resit opportunity (normally July resit period). If a student fails assessment for a second time they have a right to repeat the module.
Return of assignments
Assignments will be normally returned within three working weeks. You normally have **two working weeks** from the date you receive your returned assessment and/or written feedback or receive your exam results to contact and discuss the matter with your lecturer. See the Student's Union advice page http://www.wolvesunion.org/home/content/pages/advice/ for more details.

Registration

Please ensure that you are registered on the module. You can check your module registrations via e:Vision You should see your year tutor or the Programmes Advisor if you are unsure about your programme of study. The fact that you are attending module lectures and classes does not mean that you are necessarily registered. A grade may not be given if you are not registered.

Cheating
Cheating is any attempt to gain unfair advantage by dishonest means and includes **plagiarism** and **collusion.** Cheating is a serious offence. You are advised to check the nature of each assessment. You must work individually unless it is a group assessment.

Cheating is defined as any attempt by a candidate to gain unfair advantage in an assessment by dishonest means, and includes e.g. all breaches of examination room rules, impersonating another candidate, falsifying data, and obtaining an examination paper in advance of its authorised release.

Plagiarism is defined as incorporating a significant amount of un-attributed direct quotation from, or un-attributed substantial paraphrasing of, the work of another.

Collusion occurs when two or more students collaborate to produce a piece of work to be submitted (in whole or part) for assessment and the work is presented as the work of one student alone.

Task 2 – This portfolio item contributes 30% to the overall module grade

Assesses Learning Outcome 1 “Evaluate computer applications and their architecture with respect to the principles, practice and theories of Software Engineering” and Learning Outcome 2 “Apply appropriate theory, tools and techniques to the software development processes of synthesizing computer software enabling graduate employment in Software Engineering”.

80-100%	10 different checklist refactorings correctly applied. Each refactoring is fully described in an individual git log comment. User interface code has been decoupled from game logic exposing the logic for testing. Descriptions relate refactorings to bad smells and principles. JUnit tests have been included for extracted methods.
60-79%	8 different checklist refactorings correctly applied. Each refactoring is fully described in an individual git log comment. Descriptions relate refactorings to bad smells and principles. JUnit tests have been included for extracted methods.
40-59%	6 different checklist refactorings correctly applied. Each refactoring is fully described in an individual git log comment. Descriptions relate refactorings to bad smells and principles. Competence with unit testing has been demonstrated.
20-39%	4 different checklist refactorings correctly applied. Refactorings are described in git log comments but not easy to follow. Large steps between commits. Descriptions only partially relate refactorings to bad smells and principles. Competence with unit testing has been demonstrated.
0-19%	Less than different checklist refactorings correctly applied. Each refactoring is fully described in an individual git log comment. Descriptions do not relate well to bad smells and principles. JUnit tests have not been included for extracted methods. Source code has not been included or is problematic

Comments