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 1
% of module mark	10%
Due Date	14 May 2021
Hand-in – what?	Line numbered code in Word 365 format, containing annotated program listings. Accompanying checklist that references specific class/methods names and line numbers in the code.
Hand-in- where?	Push the solution to your GitLab repository.
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 tutors in scheduled classes

Notes:

Assignment details:

Goal

Abominodo is a text-based computer game influenced by the popular dice game *Dominos*, but converted to a puzzle form as published in *Beyond Sudoku*. The rules can be found on Canvas.

The game was originally written for computers that had text-only consoles, thus it looks quite primitive today. However, it is based on sound algorithms, which should be able to power a state-of-the-art graphical game. There has been some progress towards this in the latest version in that the player may see the dominoes in a graphical window. However, all interaction remains console-based.

The source code for the game will be shared with you through a GitLab repository. It can be compiled and run on any Java 11 development system. If you experience compiler errors it is likely that your system is not Oracle Java 11 compatible. Java 11 enforces the use of modules so Eclipse will report problems with your module permissions if you do not include a module descriptor (module-info.java) in your project.

The ultimate aim of this work is to *refactor* the application so that it is in a form that is easily modified. The reason for doing this is because there are plans to incorporate better graphics and a competitive 2 player mode, but at present the code is not structured well enough to facilitate this. It is *not* your goal to write the graphics or two player code, as this will be the focus of another project. You should concentrate strictly on improving the internal design of the software and meeting the requirements specified in this document.

There will be a series of stages that you should take towards refactoring the application. You should attempt them systematically in the order given here and not move on to later stages until you have at least made an attempt at earlier ones. The attached marking grid outlines the breakdown of marks and will be used during marking to give you feedback. You are encouraged to complete the grid yourself so that you may gauge your own progress. If you feel that you are falling behind you must make the module leader aware immediately. Note that programming accounts for around two thirds of the marks. If you are not that confident with programming you should try to maximise your grades for the written parts, but it is essential that you make some attempt at all the practical parts - if you make no attempt you cannot be given any credit, but a reasonable attempt, no matter how imperfect, may attract some credit that may make the difference in passing the assignment.

THIS IS AN INDIVIDUAL PIECE OF WORK. You may discuss ideas with staff and class mates but the artefacts submitted must be your own work. Software submitted must be derived solely by you from the original system and from no other source including partially or wholly completed work of your peers. You must not make your solution available to others either via the Internet or any other means. Using the module's Canvas forum for discussing the work with your peers and tutors is recommended, but you must not distribute substantial contributions or solutions there.

Task 1 - Evaluation of the Existing System

To line number your code you may use "cat -n", e.g. "cat -n x.java > x.txt" to create a line numbered version, x.txt, of x.java. You should make a line numbered Word 365 document adjusting the margins and fonts to maximise neatness. You should then work through the code annotating it with review comments making notes that explain how it works and explanations relating to the theory covered in the module.

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 1 – This portfolio item contributes 10% 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”

80-100%	At least 12 different Fowler bad smells or poor coding style correctly identified. Annotations demonstrate excellent or outstanding understanding of the code. Informative checklists are included.
60-79%	8-11 different Fowler bad smells or poor coding style correctly identified. Annotations demonstrate good or very good understanding of the code. Informative checklists are included.
40-59%	6-7 different Fowler bad smells or poor coding style correctly identified. Annotations demonstrate competent understanding of the code. Informative checklists are included.
20-39%	4-5 different Fowler bad smells or poor coding style correctly identified. Annotations demonstrate insufficient understanding of the code.
0-19%	Less than 4 different Fowler bad smells or poor coding style correctly identified. Annotations demonstrate little or no understanding of the code.

Comments