COMP2050 Coursework #2

Boon-Giin, Lee; Heng, Yu
boon-giin.lee@nottingham.edu.cn; heng.yu@nottingham.edu.cn

University of Nottingham Ningbo China, School of Computer Science

1 Synopsis

Coursework 2 is about maintaining and extending a modern version of classic retro game (Packman). To get started, download zip file "BestPackmanEverV5.zip". The zip file contains only "src" folder, so you have to set up your own JavaFX project in Eclipse and import the src folder. You will be asked to make some changes and additions to the project.

NOTE This coursework is about maintaining and extending existing code. So, for the maintenance part, you have to use the existing code as a basis, and not write your own Pacman game from scratch.

IMPORTANT Make sure you understand what you are writing in your Javadocs. We reserve the right to briefly interview you if we think that you do not understand what you write about. So, when you write your Javadocs, do not simply copy/paste large portions of text from existing articles or other resources - as this does not demonstrate your understanding of the topic. You might also run into issues with plagiarism.

You should roughly spend 50 hours on this coursework.

Table 1: Coursework Summary

Weight	50%
Format	• Class Diagram (as PDF). • Source Code in Form of Eclipse Project (*.zip, **NOT** *.rar). • Documentation (Javacs) • Video in 2-3 Minutes
Submission Date	9th December 2019, Monday by 4 p.m.
Late Policy	Standard Policy
Submission Method	Electronic Submission via Moodle
Feedback Date	2nd January 2020 (Expected)
Feedback Method	Individual Comments via Turnitin in Moodle page

2 Deliverable

Set up a Git repository on the university's Git server as PRIVATE (link) and use it for version control activities. The project should be name with "FirstnameLASTNAME-StudentID" for consistency, e.g., RuibinBAI-20031168. PLEASE to be NOTE that the university's Git server ONLY can be ACCESSED when CONNECTED to eduroam. Follow the instructions ("Add Users to Group") to add two GitLab members "z2019017" and "z2019078" with role permission as "Maintainer" for your coursework assessment.

Four directories (folders) should be created, namely, "src", "documentation", "diagram" and "video".

- "src" - Contains all the source codes of the project.
- "documentation" - The documentation should be delivered in form of Javadocs. We will mainly look at the Javadocs to find out how you maintained and extended the game. If it is not obvious from there, we might miss it. Also, we have only a limited amount of time to look at each coursework submitted. So, please make sure to provide informative but concise Javadocs.
- "diagram" - Provide a high level class diagram in PDF format (landscape orientation is highly recommended for clearer view) that shows the structure of the final version of your game (considering only classes, interfaces, relationships and multiplicity)
- "video" - Make a video showing your software in action, demonstrating and explaining its functionality. You could highlight 2-3 achievements you are very proud of in your video.

REPORT A maximum of one page PDF report should be submitted in the Moodle page to provide any brief description of software maintenance you have done especially related to refactoring and additions. You can also highlight any important things that you wish us to pay attention to. Be simple but precise. You should name your report as "cw2-StudentID-report.pdf", e.g., "cw2-20031168-report.pdf". A zip file should be created containing your entire project and submitted as supplementary file. Your project must be able to import into Eclipse using the import wizard. Please name your ZIP (NOT *.RAR or other archive types) file as "cw2-StudentID-project.zip", e.g., "cw2-20031168-project.zip". Failure to adhere to these rules may result in our refusal to mark your coursework.

IMPORTANT The mark and feedback of the coursework will be released in the Turnitin under the report section but assessment will be mainly based on your project in university's GitLab server.

3 Plagiarism

You are gently reminded that we are at liberty to use plagiarism detection software on your submission. Plagiarism will not be tolerated, and academic offenses will be dealt with in accordance with UNNC policy and as detailed in the student handbook. This means you may informally discuss the coursework with other students but your submission must be your own. Please also note that it is not permitted for you to copy and paste text from another source without correct referencing.

4 Assessment

This coursework is worth 100% of this module and as such is marked out of 50. The marks will be split as follows:

- **[15%]** Git Use - Version control? Utilization of Git?
- **[30%]** Refactoring - MVC pattern? Design principles?
- **[30%]** Additions - Interesting features? Creative ideas?
- **[15%]** Documentation - Javadoc? High-level class diagrams?
- **[10%]** Video - Interactive? Presentation quality? Explanations?

NOTE Good programming practice will gain higher marks. Furthermore, nicely presented and easiness of using interfaces will be rewarded. A proportion of the marks will depend on you supplying a working version of your game, and submitting a video of it in use.

IMPORTANT It is always a good practice to refer to the assessment rubric to understand how to secure high mark in the coursework. The rubric is provided for your reference, make sure to check it out at the last page.

5 Task

Do some basic maintenance of the delivered code base, e.g., adding meaningful Javadocs, organizing files in a meaningful way into packages, breaking up large classes in a meaningful way to support the idea of single repository, improving encapsulation etc.

Extend the delivered code base by adding:

- A `START` screen, displaying a picture related to the game and a button that allows going to a `SETUP` screen.
- A `START` screen: Allowing to choose background and wall colour for the game field (allowing a choice of at least 8 colours) and a button to go back to the `START` screen.
- A high score pop-up, appearing at the end of each round, showing the scores from each round, highest at the top.
- Two doors at the side of the play field to allow the pacman and the ghosts to transit between them (leaving at door 1 and immediately re-appearing at door 2 and vice versa) as showed in the Fig. 1.

IMPORTANT You need to use Java 8 for the implementation. The project files you are submitting need to be compatible with Eclipse.

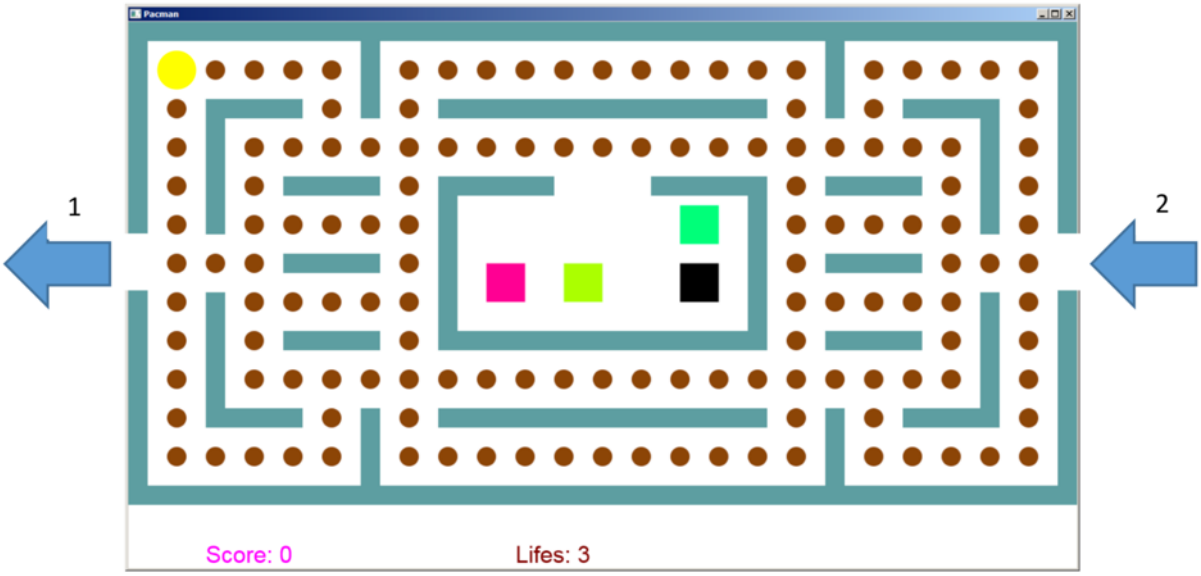

Figure 1: Pacman with Door 1 and 2.

5.1 Tips & Hints

To get higher marks, you can perform some of the tasks as follow in addition to the previous:

- Refactor the code by adding some design patterns to enhance maintainability.
- Considering organizing the code to adhere to the MVC pattern.
- Create a permanent high score list, e.g., using a file to store scores.
- Load level descriptions from file and allow running the game with different layouts.
- Load proper character figure.
- ...

6 References

If you use any text or information from any resources such as books, journals, articles, proceedings etc., you have to provide those references. A guide for references and citation can be found here.

Finally, if you have any questions, you are always welcome to post your questions in the forum under the topic Coursework#2-Q&A.

7 Q & A

If you have any questions, kindly utilize the "Q&A Forum" under the topic Coursework#2-Q&A.

REMINDER Make sure you follow strictly the instructions/guideline provided for posting the questions in the forum.

Criteria	F - 0	D - 40	C2 - 50	C1 - 60	B2 - 70	B1 - 80	A - 100
Git Use **[15%]**	No evidence of git repository set up on this project.	Remote Git repository is set up but low usage activities recorded. Evidence of local Git repository is found but little-to-no knowledge of git use evidence presented.	Remote Git repository is set up but low usage activities recorded. Evidence of local git repository is found but little-to-no evidence of commitments and purpose use of Git are presented.	Remote Git repository is set up with basic usage evidenced. Local repository is found with basic commitment and poor explanation. The purpose use of git is not sufficiently presented.	Remote Git repository is set up with basic usage evidenced. Local repository operation is found with moderate commitment and with proper and detailed explanations. Branching feature is incorporated.	Remote Git repository is set up with extensive usage evidenced. Local repository operation is found with high commitment and with informative and clear explanations. Principal use of branching is greatly illustrated.	Remote Git repository is set up with rich set of usage evidenced. Local repository operation is found with high commitment and stunning explanation, branching is used extensively. Git work flow is clearly explained, demonstrated professional way of exercising git tool for software development and maintenance, satisfying the aim of this module.
Refactoring **[30%]**	No evidence of code refactoring presented in this project.	There is little evidence that code refactoring is presented with no improvement to this project.	There is some evidence that relevant code refactoring principles have been incorporated into this project. At best, there are loose principles.	Your code refactoring refers to couple of refactoring principles with at least one OO design patterns documented, but largely lacks precision and citations.	Your code refactoring shows evidence that aligned with MVC GUI design pattern and at least two OO design patterns, but your terminology may be imprecise. You may be missing some key concepts or some of your refactoring may be incorrect or your references may be incorrect.	Your code refactoring shows significant evidence that were supported by the MVC GUI design pattern and at least two OO design patterns. Your terminology may is precise and correct. Appropriate references have been used. You made improvement to the ghost AI algorithm.	The majority code refactoring principles are justified by appropriate citations with MVC GUI design pattern and three or more OO design patterns. Those patterns are clearly articulated, use precise terminology, and cite appropriate source. You successfully implemented the original behavior of ghost AI.
Additions **[30%]**	None additions documented or presented in this project.	Some detailing that hint towards additional features or ideas, but not explained.	Some detail and explanation of very small additional features, ideas and tasks. Poorly explained.	2+ tasks are implemented with clear additional features and ideas implemented. Briefly explained.	3+ tasks are implemented with clear additional features, ideas and tasks implemented. Clearly explained but poorly reasoned/justified.	All tasks are completed. 5+ additional features and ideas implemented and well justified.	All tasks are completed. Several additional features and ideas are included. All are well explained and justified. The implementation demonstrates creativity beyond expectation.
Documentation **[15%]**	No documentation provided in this project.	Little-to-no high level diagrams provided and package information is missing. Javadoc folder created, but with low commitment, quality, lack of justified explanations.	High level class diagrams provided, but insufficient package information. Javadoc folder created with moderate commitment but a number of quality issues exists, includes poor explanations of parameters and methods.	High level class diagrams provided with package information, but with some errors. Javadoc folder created with high commitment, explanations of parameters and methods, but does not comply with OO design considerations.	High level class diagrams provided with organized and justified package information, but with some minor errors. Javadoc folder created with high commitment, well explained and justified, comply with some OO design considerations but with minor errors.	High level class diagrams provided with clear explanations of package information and presented in appropriate layout. Javadoc folder created with high commitment, all codes are with clear explanations, justified and show consistency in the project. Also, comply with OO design concepts.	The high level class diagrams provided in professional, clear and considered manner with high quality package information includes relationship between classes. Javadoc presented with stunning explanations, correct format and identical quality with Java online libraries documentation.
Video **[10%]**	No video submitted on this project.	Video showed with no explanations.	Insufficient and lack of informative contents presented in the video.	Meaningful information and small references of improvements presented, but with errors in the video.	Informative contents with relevant references of improvements presented, but with minor errors in the video.	A number of high quality, impactful and relevant references to your OO designs improvements and refactoring efforts, supported with texts explanations in the video.	A wealth of high quality, impactful and relevant references to your OO design improvements and refactoring efforts, supported with text and audio explanations. Video contents not lengthy, straight to the points with clear and appropriate pronunciation.