CW Software Maintenance Spec Sheet

Academic Year 2023/2024

This coursework is about maintaining and extending a re-implementation of a classic retro game (Brick Breaker). This version of the game isn't finished, but it works once you set it up correctly. The primary objective of Brick Breaker is to break all the bricks on the screen using a bouncing ball and a paddle. You control a horizontal paddle at the bottom of the screen using the left and right arrow keys. The paddle's purpose is to deflect the bouncing ball and prevent it from falling off the screen. Points are awarded for each brick that is successfully broken. If you want to learn more about the original Brick Breaker game and its history, you can check out this link: https://heroconcept.com/a-brief-history-of-brick-breaker-video-games/. To start, download the game's source code from this GitHub repository: https://github.com/kooitt/CourseworkGame.

Number of Credits: 75% of a 20-credit module. This coursework contributes 75% to your overall grade.

Deadline: December 11th, 2023, 11:59 PM

Assessment: The marks will be split as follows:

- 10% for git use (show your screenshot of GitHub and history on your version control)
- 30% for refactoring
- 30% for additions
- 15% for documentation (Javadocs + class diagram)
- 15% for a video, explaining your refactoring activities and additions

Note: We would like to kindly remind you that our institution employs plagiarism detection software to review your coursework submissions. Plagiarism is strictly prohibited, and any academic misconduct will be addressed following university policies and the guidelines outlined in the student handbook. You are encouraged to engage in informal discussions with fellow students regarding the coursework. However, it is important that your submission reflects your own original work. Copying and pasting text from external sources without proper referencing is not permitted. If you have any uncertainties or questions about the referencing process, please don't hesitate to reach out to the module convenors during our lab sessions or at the conclusion of a teaching session.

Requirement Specification

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 game from scratch.

Assignment Submission and Organization

In this section, we will outline the specific files that must be included in your coursework submission and how they should be organized. Please be aware that due to the large number of students in the class, the organization of your submission is crucial to ensure a smooth and efficient review process.

Assignment Report: A **Readme.md** file, documenting the work you've done for the coursework, including key changes made for maintenance and extension, along with their locations and reasons. WARNING: If you do not mention it here, do not blame us later if we miss it.

Here's how to structure your Readme.md:

- **Compilation Instructions**: Provide a clear, step-by-step guide on how to compile the code to produce the application. Include any dependencies or special settings required.
- **Implemented and Working Properly**: List the features that have been successfully implemented and are functioning as expected. Provide a brief description of each.
- **Implemented but Not Working Properly**: List any features that have been implemented but are not working correctly. Explain the issues you encountered, and if possible, the steps you took to address them.
- **Features Not Implemented**: Identify any features that you were unable to implement and provide a clear explanation for why they were left out.
- **New Java Classes**: Enumerate any new Java classes that you introduced for the assignment. Include a brief description of each class's purpose and its location in the code.
- **Modified Java Classes**: List the Java classes you modified from the provided code base. Describe the changes you made and explain why these modifications were necessary.
- **Unexpected Problems**: Communicate any unexpected challenges or issues you encountered during the assignment. Describe how you addressed or attempted to resolve them.

Design Diagram: A file called **Design.pdf** contains a high-level class diagram that shows the structure of the final version of your game This class diagram should focus solely on classes (excluding fields and methods, unless they are pertinent for understanding design principles/patterns), interfaces, relationships, and multiplicity. If you're using software to reverse engineer the class diagram, it is important to ensure that the delivered diagram is accurate and aligns with the specified requirements. Make sure the class diagram is well-labeled and easy to interpret, as it will help us understand the structural aspects of your game's code.

Source Code Documentation: You need to generate and include a copy of the Javadoc documentation for your coursework. Javadoc creates linked HTML web pages to make it easier to browse through your project's implementation. The HTML web pages produced by Javadoc should be placed in a folder named "**Javadoc**".

The Javadoc documentation should contain the following elements:

- **Brief Class Descriptions**: Each class should have a brief description that summarizes its purpose and functionality.
- **Method Documentation**: For each method, you should provide documentation that includes a description of the method's purpose, input parameters, and return values.
- **Original Source Code**: The Javadoc should include links or references to the original source code for both new Java classes and modified Java classes.

The Javadoc documentation should be informative but concise. This documentation will be an essential resource for understanding how you maintained and extended the game. It's crucial to provide clear and detailed explanations within Javadoc, as it will help us assess your work, especially when we have limited time to evaluate each coursework submission.

Project Implementation Files and Folders Description: A zip file containing your entire local project folder. This zip file should be configured so that it can be easily imported and run in either Eclipse or IntelliJ. To ensure there are no hardcoded path dependencies, please test your final version on a different computer before submission. This should help to uncover hardcoded path dependencies, which was a major issue in previous years. Name your zip file **SurnameFirstName_IDE_JavaVersion.zip**, where IDE represents the name of the IDE you used, and Java Version represents the Java version you used. Here is an example: "TanChyeCheah_IntelliJ_12.zip".

Version Control: Set up a PRIVATE remote git repository at GitHub and add my account (kooitt) into your collaborator list. Your remote repository needs to be named **COMP2042_CW_OWAUserName** (e.g., in my case it would be COMP2042_CW_kcztcc). Then follow the setup instructions provided to "Push an existing folder" (i.e. do an initial push to upload files from your local to your remote repository). Now you are ready for coding with version control.

Demo via Screen Capture: Use screen capturing software to demonstrate the features of your application. The video should be approximately 3 minutes in length. The video file needs to be saved in MPEG or MP4 format. Also, you can use MPEG2 and MPEG4 compression formats. Your screen capture demo has to be called **surnameFirstnameDemo.mp4** (or .mp2 or .mpg).

Note that the demo video will be the primary way in which we assess whether your software is working. The demo video should show your software running and then explains your refactoring activities and additions. You also must highlight the **TWO** achievements you are most proud of.

Folder and File Organisation

You are required to create a root folder called **COMP2042surnameFirstname**, where "surname" is replaced with your surname, likewise for "firstname", The COMP2042surnameFirstname folder contains (and organises) digital copies of all of the files that compose the assignment. Thus, when completing the submission of the assignment, you have a directory structure in your **COMP2042surnameFirstname** folder that looks like this:

README.md
Design.pdf
javadoc
surnameFirstnameDemo.mp4
project/SurnameFirstName_JavaVersion.zip

For this coursework, Moodle limits uploads to a single file of up to 250MB. Before you upload your coursework, please create a zip archive of your COMP2042surnameFirstname folder and then upload that zip archive to Moodle:
https://moodle.nottingham.ac.uk/course/view.php?id=137675

MARKING CRITERIA/RUBRIC

Assessment Criteria and Marking Overview Tasks	Marks given	Marks awarded
1. Git • Project exists • Commits • Regularity • Commit messages • Branching/merging • Meaningful .gitignore	10	
2. Refactoring • Meaningful package naming/organisation • Basic maintenance (e.g. renaming classes; encapsulation; deleting unused resources) • Supporting single responsibility by splitting up classes • MVC • Other patterns • Meaningful JUnit tests • Correct use of build tools (Maven or Gradle) • Complete translation from Swing to JavaFX • module-info.java file (compulsory)	30	
3. Additions • Additional (playable) levels • Anything else exciting (reward) or bad (penalty)	30	
4. Documentation • Readme.md: highlighting the key changes (maintenance + extensions) + where + why • Javadocs: Created (and deposited in the correct location) • Javadocs: New ones added (half marks if only comments added) • Javadocs: Complete (half marks for substantial amount) • Javadocs: Informative and concise • Class diagram: Something meaningful present • Class diagram: High level • Class diagram: Conforms with code (of the final program)	15	
5. Demonstration video • Showing software running • Explaining refactoring activities and extensions • Highlighted two achievements most proud of • Timing (approx. 3 minutes)	15	
Total marks	**100**	
Total	**75%**	