CW Software Maintenance Spec Sheet

Academic Year 2025/2026

This coursework is about maintaining and extending a re-implementation of a classic retro game (Tetris). This version of the game isn't fully complete, but it becomes functional once set up correctly. The primary objective of this game is to arrange falling blocks (known as Tetrominoes) to form complete horizontal lines. When a line is completed, it disappears, and the player earns points. You control the falling blocks using the arrow keys: left and right to move sideways, down to accelerate the fall, and up to rotate the piece. The challenge is to prevent the blocks from stacking up to the top of the screen.

To get started, download the game's source code from this GitHub repository: https://github.com/kooitt/CW2025. If you want to learn more about the history and impact of Tetris, you can check out this link: https://en.wikipedia.org/wiki/Tetris.

Number of Credits: 100% of a 20-credit module. This coursework contributes 100% to your overall grade.

Deadline: December 8th, 2025, 11:59 PM

Assessment: The marks will be split as follows:

- 10% for git use
- 35% for refactoring
- 25% for additions
- 15% for documentation (Javadocs + class diagram + README)
- 15% for a video, explaining your refactoring activities and additions

Note:
Please be reminded that all coursework submissions are reviewed using plagiarism detection software. Plagiarism is strictly prohibited, and any cases of academic misconduct will be handled in accordance with university policies and the student handbook. While you are welcome to have informal discussions with peers, your submitted work must be your own original contribution. Copying text from external sources without proper referencing is not allowed.

According to Section 2.2.2 of the University of Nottingham's Quality Manual, unauthorised use of AI in assessed work is considered False Authorship and is treated in the same way as plagiarism. For this coursework, the use of generative AI is permitted but optional. If you choose to use it, you must be transparent and detailed in acknowledging this through the COMP2042 AI Use Declaration Form. If you are unsure about the referencing process or AI use declaration, please consult the module convenors during lab sessions or after a teaching session.

Requirement Specification

This coursework is about maintaining and extending an existing codebase, emphasizing your ability to understand, modify, and improve existing software rather than writing from scratch.

For the **maintenance** component, you are required to work directly with the code provided. You should analyze how the current system operates, identify potential issues or inefficiencies, and make targeted improvements to enhance stability, performance, and readability. This may involve fixing bugs, refactoring messy code to improve structure and maintainability, or updating documentation to make the project easier to understand. It is important that you demonstrate a clear understanding of how the code works and show evidence of thoughtful, purposeful maintenance rather than random edits or changes.

The **extension** component requires you to implement substantive new features that meaningfully expand or improve the gameplay experience. Examples include designing and adding new playable levels, introducing fresh game mechanics or challenges, refining player controls to make movement and interactions smoother, or incorporating dynamic systems such as adaptive difficulty or scoring. You are also encouraged to think creatively and consider unique gameplay features such as power-ups, penalties, or innovative modes that make your game unique. Minor or superficial edits such as changing a single character, adjusting a variable, or replacing an image do not qualify as meaningful extensions and will receive little to no marks.

Finally, the **quality of your work** is very important for your marks. Your new features should fit smoothly into the existing code, be well-designed, and work properly without causing new problems. You should also test your work carefully to make sure everything runs as expected. Including clear comments and showing proof of your testing will make your project stronger and show that you built your features with care and professionalism.

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.

Documentation: You are required to prepare a **Readme.md** file that documents the work completed for this coursework. The file should clearly describe the key modifications made for system maintenance and extension, including their specific locations within the codebase and the rationale behind each change. Clear and detailed documentation is essential, as it allows us to assess your work effectively within the limited time available for each submission. Your Readme.md will serve as a vital reference for understanding how you maintained and extended the game.

Here's how to structure your Readme.md:

- **GitHub**: Provide the link to your GitHub repository.
- **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 only show the classes, interfaces, relationships, and multiplicity. Attributes and methods should not be included unless they are necessary to illustrate important design principles or patterns. Ensure the diagram is clearly labeled and easy to read, as it will be used to evaluate the structural design 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 be informative but concise, and 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.

Project Implementation Files and Folders Description: A zip file containing your entire local project folder. 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 **StudentName_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_22.zip".

Version Control: Begin by forking the game's source code from the provided GitHub repository: https://github.com/kooitt/CW2025. Forking creates your own copy of the repository under your GitHub account, allowing you to make changes without affecting the original project. After forking, open your chosen Integrated Development Environment (IDE), such as IntelliJ IDEA or Eclipse. Import the cloned Git repository into your IDE. This process may vary slightly between IDEs, but you typically can find an option like "Import Project" or "Open Existing Project." Select the folder where you cloned the repository. Now you are ready for coding with version control.

Important: Remember to add the link to your GitHub repository in your README file. Without this link, I cannot access your work or grade the GitHub component.

Demo video: Use any screen capturing software to create a demonstration video showcasing the features of your application. The video should **NOT EXCEED** 3 minutes long and saved as **MPEG, MP4, MPEG2, or MPEG4**, named **Demo.mp4** (or .mpg / .mp2). It should clearly show your program in action while you explain your refactoring activities and any additional features you've implemented. Additionally, make sure to highlight the **TWO ACHIEVEMENTS** you are most proud of during the demonstration. Please ensure that your explanations are recorded in your own voice, rather than using AI-generated speech.

Folder and File Organisation

You are required to create a root folder called **COMP2042StudentName**, where "StudentName" is to be replaced with your full name. This folder must contain digital copies of all files that compose the assignment. Accordingly, upon submission, the directory structure within your **COMP2042StudentName** folder should appear as follows:

README.md
Design.pdf
Javadoc
Demo.mp4
StudentName_IDE_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 COMP2042StudentName folder and then upload that zip archive to Moodle:
https://moodle.nottingham.ac.uk/mod/assign/view.php?id=8665168

MARKING CRITERIA/RUBRIC

Assessment Criteria and Marking Overview Tasks	Marks given	Marks awarded
1. Git • Project exists • Frequent, meaningful commits • Descriptive commit messages • Branching/merging • Clear history showing logical progression	10	
2. Refactoring • Meaningful package naming/organisation • Basic maintenance (e.g. renaming classes; encapsulation; deleting unused resources) • Supporting single responsibility by splitting up classes • Design patterns • Meaningful JUnit tests • Fixing the bugs	35	
3. Additions • Implement new playable levels • Gameplay enhancement (improve the overall gaming experience) • Innovative feature design (Incorporate unique and creative features that set the game apart from other) • Quality of implementation	25	
4. Documentation • Readme.md: highlighting the key changes • Javadocs: Informative and concise • Class diagram: consistent with final code.	15	
5. Demonstration video • Showing game running • Explaining refactoring activities and extensions • Highlight two proud achievements • Timing (No more than three minutes)	15	
Total marks	**100**	