CW Software Maintenance Spec Sheet

Academic Year 2020/2021

CW is about maintaining and extending a re-implementation of a classic retro game (Frogger). This version has never been completed, but at least it runs, once it is set up properly.

More information about the original Frogger game and its history is available here. You can also pick up some ideas for creating additional levels.
https://www.arcadeclassics.net/80s-game-videos/frogger

To get started download the original re-implementation from the following GitHub repository.
https://github.com/hirish99/Frogger-Arcade-Game

Set up a project in your preferred IDE (either Eclipse or IntelliJ) and embed the files that you just downloaded. Note that the GitHub repository mentioned above only provides source code and resources but no project files, as it is good practice to ignore IDE-specific generated files for source control. Set up a remote git repository at GitHub. Your remote repository needs to be named COMP2042_CW_CompSciUserName (e.g. in my case it would be COMP2042_CW_neoh). Now you are ready for coding with version control (i.e. do an initial push to upload files from your local to your remote repository).

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

What to produce:

- A README.md file (max. 500 words), documenting the work you conducted (highlighting the key changes you made for maintenance and extension, where you made them, and why you made them)
- High level class diagram that shows the structure of the final version of your game (considering only classes (excluding fields and methods, unless they are relevant for understanding design principles/patterns), interfaces, relationships, and multiplicity). If you use software to reverse engineer your class diagram, make sure the delivered diagram is correct and follows the above requirements.
- The source code documentation (Javadocs) should be delivered in form of a Javadoc folder inside your project folder. Besides reading your README.md file we will 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. So, please make sure to provide informative but concise Javadocs.
- Finally, you have to make a video (very briefly) showing your software running and then (for the main part) explaining your refactoring activities and additions. You should also highlight two achievements you are most proud of.

Important:

- 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 Frogger game from scratch.

For moderate marks:

- Set up a git repository in GitHub and use it actively for version control activities
- Do some basic maintenance of the delivered code base (e.g. adding meaningful Javadocs, organising files in a meaningful way into packages, breaking up large classes in a meaningful way to support the idea of single responsibility, improving encapsulation, etc.)
- Extend the delivered code base by adding:
    - A START screen, displaying a picture related to the game and a button that provides access to a INFO screen explaining the game operation.
    - A HIGH SCORE popup, appearing at the end of each round, showing the scores from each round, highest at the top

For higher marks: In addition to the previous, do some of the following...

- Refactor the code by adding some design patterns to enhance maintainability
- Organize the code to adhere to the MVC pattern
- Create a permanent high score list (using a file to store scores)
- Add interesting levels to the game (either follow the original Frogger game levels or come up with your own ideas)
- Add meaningful JUnit tests
- Use build files (Ant or Maven or Gradle)

You have to use Java 10 or higher and JavaFX 10 or higher for the implementation. The project files you are submitting need to be either compatible with Eclipse or IntelliJ.

Deliverables: Three separate files, uploaded to Moodle:

- A zip file containing your ENTIRE LOCAL PROJECT FOLDER (including a copy of your README.md and Javadocs files). It needs to be possible to IMPORT (or OPEN) and RUN your project in either Eclipse or IntelliJ. To avoid disappointment later, test your final version on a different computer. This should help to uncover hardcoded path dependencies, which was a major issue in previous years. Name your zip file: "Surname_FirstName_IDE_JavaVersion.zip", where IDE represents the name of the IDE you used and JavaVersion the Java version you used. Here is an example: "Neoh_SiewChin_IntelliJ_12.zip".
- Your final class diagram as pdf. Please name your pdf as follows: "Surname_FirstName.pdf". Here is an example: " Neoh_SiewChin.pdf"
- A video of up to 3 minutes, as described above, in a common format. Please test your video before submitting it, to avoid disappointment. Please name your video as follows: "Surname_FirstName.EXT", where EXT represents the extension related to your video format (e.g. mp4; mpg; avi). Here is an example: "Neoh_SiewChin.mp4".

Note: You are gently reminded that we are at liberty to use plagiarism detection software on your submission. Plagiarism will not be tolerated, and academic offences will be dealt with in accordance with university 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 work. Please also note that it is not permitted for you to copy and paste text from another source without correct referencing. If you are unclear about this process, please discuss with the module convenors during one of our lab sessions or at the end of a teaching session.

Assessment Details:

- This coursework is worth 75% of your module mark.