• [15%] Git Use (e.g. push, branch, merge, providing .gitignore).   
• [30%] Refactoring.   
• [30%] Additions.  
• [15%] Documentation (e.g. README.md, Javadocs and class diagrams). • [10%] Video (explaining improved activities and additions).

**TODO**

**√**

**√**

**√**

**√**

* A “README.md” file (with maximum 5000 words), documenting the work you conducted (highlight- ing the key changes you made for maintenance and extension, where you made them, and why you made them). You should clearly stated all the applied OO design patterns, GUI design patterns, refactoring methodologies with detail text explanations and proper references to the code in the README.md file. Otherwise, they will be treated as non-implemented.

**√**

* High level class diagram in diagram folder that shows the structure of the final version of your game which consider only classes (exclude fields and methods, unless they are relevant for under- standing design principles/patterns), interfaces, relationships and multiplicity. (Visual paradigm)

**√**

* The source code documentation (Javadocs) should be delivered in form of docs folder inside your project.

要求：

**√**

* 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:

**√**

* + TWO BUTTON: a button to start game and a button to provide information of the game.

**√**

* + –  A HIGH SCORE pop-up, appearing at the end of each round, showing the scores from each

round, highest at the top.

Diagram

Description automatically generated

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

* Refactor the code by adding some design patterns to enhance maintainability.

**√**

* Organise 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 Sokoban game levels or come up with

**√**

your own ideas).

**√**

* Add meaningful JUnit tests.

**√**

* Use build files (Ant or Maven or Gradle).

**√**

* Other additions of game feature or other implementations that innovative and beyond

creativity - surprise us! **Reset Level(Hotkey : space)**

**√**

* 加背景音乐

Good programming practice will gain higher marks. Furthermore, nicely presented and easiness of using interfaces will be rewarded. High quality additions of game features are expected to gain higher mark, e.g. implementation of 3 basic and simple features (i.e. START screen, HIGH SCORE screen, sound effects etc.) will not gain you the “Good” performance, however, implementation of 1 or 2 creative features (at least half of the total (3) implemented features) allow you to gain the “Good” performance.

**√**

**√**

You need to use Java 15 and JavaFX 15 for the implementation. The project files you are submitting need to be compatible with IntelliJ.

**√**

.