Requirements analysis:

The purpose of the project is to develop a fully working (see below for the definition) 2048 game. This game should be playable on a regular desktop computer, regardless of Operating System. Furthermore, it should work on any modern computer. Users should be able to save their highscores. Users should also be able to close the game and reopen it, starting where they left of. In case users want to start over, they should be able to do so. Users should be able to see their current score and their highscore. It would also be nice if users could see the highest value they have ever reached in the game.

The game should also be visually appealing: when a move is made, the tiles should visually move into that direction. Also, when a tile spawns, it should do so with a "jojo" animation. When the user reaches a tile of 2048, there should be a dialog with the message that the user has won asking whether or not to continue.

A basic, working version will be presented in two weeks. For tracking purposes, git will be used. Maven and DevHub should be used to supply project management and continuous integration. To deliver to the cross-platform requirement, the game shall be written in Java.

Requirements specification:

As mentioned, the purpose is to develop a fully working 2048 clone. Here it will be explained what is defined under "fully working", together with other requirements taken from the requirements analysis:

Game rules (equal, high priority):

- The game is won when two tiles of value 1024 merge into a single 2048 tile.
- A score should be kept of the progress of the game. Merging two tiles adds the value of the merged tile to the score.
- Only tiles of the same value can be merged.
- Two tiles of the same value are merged into one single tile when they collide. The value of the resulting tile will be the sum of the two tiles.
- Two tiles cannot merge into one if one of them has already been merged this move.
- Every move, all tiles should move into that direction, for the greatest distance possible.
- Initially, the grid should contain only two tiles with either value 2 or 4.
- The player can move around tiles on the grid using the four arrow keys on the keyboard.
- Moves can go into the upper, lower, left and right direction.
- Tiles can not go off-grid
- Moves are only valid when adjacent squares in the direction of the movement have the same value, or are empty.
- After every valid move a new tile should appear, with a value of either 2 or 4.

- The chance of a new tile appearing with the value 2 is 90%, in contrast to a chance of 10% on a value of 4.
- When no more moves are possible before the game is won, the game is lost.
- The game ends when the player loses.
- After winning the game, the user can decide to keep playing or restart the game.
- After the user has won the game and keeps playing, the user can only increase the highscore, and is thus playing in endless mode.
- In endless mode, the player can still lose the game for the above mentioned criteria.

As for the user interface, these requirements have been made (sorted from high to low priority):

- The interface should display a board to the user with a 4x4 grid on it.
- There should be an indication of the current score, the highscore and the highest tile ever reached.
- There should be a replay button.
- Tiles of different values should have different colors.
- When the game is won, a dialog will pop up telling the player he/she won and asking whether or not to continue.
- When moving tiles, there should be a visual indication of this movement.
- When a tile spawns, it should do so with a "jojo" animation.

Other functional requirements are (sorted from high to low priority):

- The game should be able to be both started and restarted by the user.
- After closing a game, upon restart, the game should start where it left of before being closed.

Non-functional requirements are:

- The game should run on a regular desktop computer, regardless of Operating System.
- The game should run on a modern computer.
- The use of Java.
- The use of Maven.
- The use of Git.
- The use of DevHub.
- A working version due in two weeks.