Group 10 Phase 2 Deliverables

– describe your overall approach to implementing the game

We divided our work among all the members. We needed to create all the classes after referring to our UML diagram so each team member had been assigned to different classes. We decided to create and code our classes individually. Then we discussed how the classes would merge and integrate with each other. Once everyone was done creating all the methods and classes, then we would bring everything together. This would allow us to use the different methods and fields from the entire program.

– state and justify the adjustments and modifications to the initial design of the project (shown in class diagrams and use cases from Phase 1)

There were a few changes in the fields from individual classes. New fields were added and some were removed. We implemented a few changes in how every class interacts with other classes and how the classes are connected to each other.

– explain the management process of this phase and the division of roles and responsibilities

Dates	Meeting notes
Feb. 28th	First meeting ● Get familiar with Maven and Javadoc ● Use assignment 2 to test and get used to design-patterns
Mar. 04th	Second meeting Separating features and working on different branches ● Board (Karim) ● GameManager(Herbert) ● Entities(Maggie) ● EnemiesMovements (Tony)
Mar. 12th	Third meeting Updates on process Adding updates to the report Tony ● pom.xml ● AIPathManager class with implementation of A* pathfinding Maggie ● Implements draft classes (punishment, rewards and bonus rewards) Karim ● Implements classes for board and cell

	Herb ● Implements GameManager and UI **There were short meetings along the way to help each other out with clarifying different requirements and implementing the different functionality and classes.*
Mar. 20th	Writing deliverables as a group Finalizing the report Creating merge requests

– list external libraries you used, for instance for the GUI and briefly justify the reason(s) for choosing the libraries

Java.awt.event - Used to handle key inputs and issue suitable output
Java.awt.Image - Used to load images and animation

– describe the measures you took to enhance the quality of your code

We pushed our individual work to branches, which are centred around specific features, such as the grid/board, or movable entities. Once a feature is complete, we review it with at least one other teammate for feedback and to check compatibility with existing code. The feature branch is rebased onto the master branch and merge conflicts are resolved, tested for errors, and any bugs are fixed. Then the branch is pushed onto the master branch, ensuring the best possible code for everyone to work with.

We peer-reviewed each other's code to come to a consensus on the best and most clean implementation of the code. Since we are aware of each other's code, we were able to discuss and remove redundant classes and functions.

– and discuss the biggest challenges you faced during this phase

1. It was a little unclear to make the initial decisions on how to work on the coding part, and it was hard to see a clear picture of the implementation of the game.
2. Getting everyone to work through the same GitLab repository, using branches.
3. All the codes needed to be merged and integrated with each other.
4. Merge requests were a new concept for half of us.