- Describe your overall approach to implementing the game
    - Our team decided to divide the work among our team by having each member implement a specific module designed in our UML diagram in Phase 1. We also decided to write code for these modules independently, however, we had frequent meetings and utilized a discord channel to ask any questions.
- state and justify the adjustments and modifications to the initial design of the project (shown in class diagrams and use cases from Phase 1)
    - We implemented a new class called collisions, which handles collisions between players and entities.
    - Added a Map class which creates a level based on the map representation in a text file.
    - Changing a part of structure of UML, make the relation between pattern be easier to understand and implement.
    - We redesign our board implementation, added a sqaure class as a basement.
    - Split Trap from enemy, move it to barrier
- explain the management process of this phase and the division of roles and responsibilities
    - For managing the workload in this phase, we decided to split the work based on different modules which needed to be implemented. We also had consistent meetings to see if any member needed help.
    - We agreed to meet at least 1-2 times a week and also utilized the discord server to ask any questions.
    - Roles:
        - James (UI Development)
        - Harmeet (Player and Collision Logic)
        - Huntley (Enemy Design)
        - Jesse (Maven and overall Structure)
- list external libraries you used, for instance for the GUI, and briefly justify the reason(s) for choosing the libraries
    - java.util.ArrayList - Arraylist of entities
    - java.util.List - List of entities, squares
    - Java.awt - Used to create graphical user interfaces for Java programs
    - java.awt.Image - Open images
    - java.awt.Graphics2D - Render images
    - javax.swing.JFrame - Develop Java interface applications
    - java.awt.EventQueue - causes the runnable to call its run method in the dispatch thread of the system event queue. This will happen after all pending events have been processed
    - java.awtcribe the measures you took to enhance the quality of your code
    - javax.swing.JPanel - JPanel can provide a common container for light controls added to the form
    - java.awt.Color - All classes used to create user interfaces and draw graphic images

    - java.awt.Dimension - The height and width values of the Dimension class are integers that indicate how many pixels there are
    - java.awt.Font - Set the window to display the desired font
    - Our group had a branch assigned to each member. This made it easier to avoid any merge conflicts, and allow members to work on their code with ease. Once the member was finished their implementation, they pushed the code to the master branch.
- discuss the biggest challenges you faced during this phase
    - At the beginning of Phase 2, it was difficult to split our work among each member. We did not have a clear understanding of where to start coding.
    - How to merge our work into the master branch.
    - Giving a AI path finding to enemy
    - How to match both class when we are designing our own class, at first we work seperately so the whole structure has lot of conflict and chaos
    - Leaning how to use maven in our job
    - Using design pattern for improving program

| | |
|---|---|
| Feb.14 | The team discussed the problem of how to implement abc, sorted out all the work, and assigned it to everyone. |
| Feb.21 | Made simple program file structures, including maven and pastern design. Integrate the project into a uniform format, and declear the method we need |
| Feb.28 | Each team member exchanged views, redesigned some unreasonable parts of the program, and unified the methodology, so that the program can be better understood and applied in the later stage |
| Mar.07 | Implementing method, Debug, delete redundant method and code. |
| Mar.14 | Merge individual branch to master, edit report, add Javadoc and comment within the code. |