Group 045 Course Project:

The project can be accessed by running the main method of the Game class

Phase 2 Specification

The project is a grid based turn based strategy game based on the popular franchise Fire Emblem.

A main feature added in Phase 2 was menu screens, which appear at game startup and where you can select different characters to play with and maps to play on.This feature allows for easy extension in the case we want to add additional characters or maps.

The player selects a team of characters with varied stats and completes a map by eliminating all enemies. The game is controlled through a mouse, which is used to issue commands to player characters or to view stats.A health stat keeps track of character health,and when a character's health is depleted,they are removed from the map.

Commands include:
Movement, where characters move across tiles based on a speed stat. Phase 2 work on this feature included optimization, pathfinding around obstacles,
and elimination of errors.
Attacking, Characters can also attack enemies and deal damage based on their attack stat. Phase 2 work on this feature included the ability to attack
an enemy character beyond close range by moving in addition to attacking, which counts as only a single action. Also the addition of ranged characters and healers.
Item usage, a feature implemented entirely in phase 2, where a character selects and uses an item from a popup menu. Items currently implemented include stat modifiers.

Additional features from phase 2 include:
Transitions and graphics that were added to indicate when character actions are used and when turns are over.
Graphics that display stats and show when a character is selected.
Enemy AI characters actions, which are controlled by basic logic.
Various UI additions.

Likely areas for future extension and expansion:
Additional characters
Additional maps
Additional items

Additional graphics
Weapon system for increased strategy

Design Decisions:
In terms of design decisions we made as a group, we decided to implement a range system for the characters so that they can have a certain range that they can attack enemies. For items we decided to make each character have a certain number of items in their inventory right from the start rather than scatter them around the map due to time constraints and simplicity. We made decisions about implementing an AI system where it would simply move towards the closest enemy and attack them. A few other design decisions we made were to have obstacles as hidden characters on the map so that they simply can't walk through them.

Clean Architecture:
Clean Architecture: We have segregated our classes into packages based on their roles in the clean architecture hierarchy.
Characters, items and the map count as entities.
The Action class is our use case.
The Game class is a gateway, and the UI class is self-explanatory in its purpose.

SOLID Design Principles:
Single Responsibility principle - All of our classes have a single responsibility. Open/closed principle- All of our entities can be extended and we can modify details to subclasses without changing the functionality of the entity.
Liskov Substitution method - Our item interface, where you may make an item that is a consumable can be replaced by the parent type of Item, or Hppot the type can be substituted as a consumable.
Interface Segregation Principle - Our item interface is relatively small, with just a view key functionalities.
Dependency Inversion Principle - Changing individual pieces of our code, wouldn't require us to change other pieces of our code, such as characters and item, changes to either of them are independent.

Design Patterns:
When it comes to implementation of the interactions between items and characters, we used the Dependency Injection pattern, where we avoid hard dependency between the two classes and instead pass an object of character to a certain item method which prevents hard dependency. In a way we also implemented the Strategy design pattern, where we use an Item interface, where multiple different items with different statistical modifiers implement Item, so when implementing interactions like characters,

the action class, or the game class, it relies upon the Item interface, which means it is only dependent on the particular functions that all items have.

Phase 2 Progress Report:

Jack:
Phase 1:
-Implemented basic UI for a demonstration of the final product
-Added basic graphics and animations for a few characters
-Implemented sprites from the game *Fire Emblem Heroes* into the project
Phase 2:
-Overhauled Game and UI classes to allow for menu screens
-Added graphics and animations for more maps and characters
-Implemented sprites from the game *Fire Emblem Heroes* into the project
-Extended UI class with additional features
-Linked UI class to all other features including the map, characters, AI, items, etc
-Worked collaboratively to optimize character movement and add obstacle pathfinding

Antony:
Phase 1:
-implemented the item interface and created the subclasses of consumable items including attack and health pots
-created the character subclasses from the implementation of Jonathan's abstract character class
Phase 2:
-implemented item class and subclasses
-overhauled character class to eliminate code smells
-designed character ultimate abilities
-created 3 more solid character subclasses extending the abstract character class

Zihao:
Phase 1:
-implemented Map class
-assisted in modifications to the Map and Action classes
-assisted in writing of Game class
Phase 2:
-designed and implemented logic for AI related methods
-added Obstacle objects (this was later reverted as a more efficient method of marking obstacles was found)
-worked collaboratively to optimize character movement and add obstacle pathfinding
-added methods in action class for item usage and other additional functionality

Johnathan:
Phase 1:
-extended functionality of abstract character class
-assisted in the implementation of item class and related subclasses
-provided design oversight
-assisted with testing features implemented by others
Phase 2:
-wrote basic tests to ensure that core parts of the code work as intended

James:
Previous
-linked Game, Action and Map classes to newly introduced UI class
-wrote logic for mouse inputs/commands in Game class
-wrote helper methods in Game class
-implemented initial attack and movement actions in Action class
Phase 2
-assisted Jack and Zihao with the implementation of the AI method in the Action class
-modified Game to ensure continued compatibility with UI and Map classes
-assisted with merging code/resolving conflicts and general bug fixes and troubleshooting
-assisted with AI testing and logic

Relevant Pull Request/Github Features:
Our use of github features was limited mostly to pulls and commits to individual branches. Merge conflicts were often sorted out individually when pulling from another group members branch. Major pull requests include:

Commit 4b2028a - Antony:
Jhin and Irelia classes · CSC207-UofT/Course-Project-group-045@4b2028a (github.com)
Due to technical difficulties with my laptop and not exactly a lot of time to deal with it, Antony never really made a pull request so instead it will be from a commit or push that would've been a pull request if it weren't for said difficulties. It was already reviewed by group members Jonathan and Jack, before this was pushed.
This commit essentially created the first two solid character subclasses with unique stats and abilities and would be the template for later characters that were implemented after, basically combining the functionality of the character abstract class and giving them solid names and attack power, speed, etc.

Pull #15 - Jack:
Jack by Jackliang441 · Pull Request #15 · CSC207-UofT/Course-Project-group-045 (github.com)

This pull combined much of the basic AI of the enemy characters and the advanced UI features. It changed our game from a simulation to a proper turn-based game, as the enemy characters now had an AI to attack and the UI to reflect their actions.

Pull #17 - Jonathan:
https://github.com/CSC207-UofT/Course-Project-group-045/pull/17
This pull request cleaned up and added some basic tests which tested core functionality of the code for important classes such as Character and Map.

Pull #11 James:
https://github.com/CSC207-UofT/Course-Project-group-045/pull/11
This pull request from phase 1 contained major changes to the Game, Action and Map classes in order to ensure compatibility with the new UI class. It also contained code for the basic logic of many of the methods in the Action class, which were then built upon and expanded in the following phase.

Pull #13 Zihao:
https://github.com/CSC207-UofT/Course-Project-group-045/pull/13
This pull request added an additional obstacle object to be placed on maps. Obstacle objects can significantly alter the pathfinding of AI enemies while also limiting the player movement. Character class was also updated to work with the new added class.

Testing Coverage:
Character test coverage ensures that:
-Characters cannot move through other characters
-Characters cannot move over obstacles
-Characters cannot attack characters on the same team

AI test coverage ensures that enemies will:
-attack a player character if it is in range
-attack the lowest health player character if multiple player characters are in range
-move towards the closest player character if no player characters are in range

Additional coverage ensures:
-players cannot enter a map with no characters in party

Packaging Strategy:
We intended to package our code according to clean architecture, separating our classes into Entities, Use cases, and controllers, however we were unable to since it required major

refactoring of the code, which we were unable to do due to time constraints. This would have improved code organization and helped define the roles of each class. Character and all of its subclasses are entities, as well as the Map and item classes. The Game class and the UI class are controllers. Action is the main use case and encompasses the ways that all the entity classes interact with each other.

Phase 2 Accessibility:
Equitable use: the project provides an similar experience for all users without segregating or stigmatizing
Flexibility in use: The game is controlled using a mouse, which allows for both right and left handed use. Precise clicking is unnecessary due to the size of the grid and character models. Game adapts completely to the user's pace as it is turn-based with no timer.
Simple and Intuitive use: Game controls are easy to learn and straightforward. Minimal literacy and language is needed. Relevant actions have visual feedback (moving models, attack animations).
Perceptible information: Game uses multiple systems to highlight important information including attack and movement range indicators, character stat screens, icons.
Tolerance for error: Character models are large and easy to click. Action preview shown for movement and attacks, allowing for cancellation if the player changes their mind. Eliminated errors/exceptions from unexpected actions in the graphical interface.
Low physical effort: Game only requires mouse controls, allows for healthy body position and ease of use.
Size and space for approach and use: Game screen is compact with large graphics, allowing for clear visuals even when standing or from a distance. Mouse controls neatly in a small area, no need for a giant mousepad or desk

Target demographic is casual gamers/people with little gaming experience, and those who simply like clicking buttons and receiving visual feedback. Since the game is turn-based and single player, it appeals to people who don't have the time to spend long sessions gaming or are looking for a more relaxing experience.
The strategy element also appeals to those who play board games or have similar hobbies.

There are demographics that the game is unlikely to appeal to, such as individuals who like to play games with friends, as the game is single player.
Individuals who prefer activities such as sports or other entertainment other than video games will also find little use in our program.
individuals who prefer real time games or more action intensive games may find the turn based nature of the game unappealing.
The cartoony visual design may also be unappealing to older demographics who believe games and cartoons are for kids.