Final Project Reflection

Kelvin Lu

Initial Design:

Because I really enjoyed the work that we did on designing a combat simulator for project 3 and project 4, my idea for the final project was to incorporate that system into a board game. The user would play as a character who needed to collect items, fight enemies, and ultimately defeat the boss to win the game. Spaces the user lands on have a chance to trigger an event of some kind, specific to the subclass of the general Space class that each space inherited from. I decided my implementation would have the following classes:

Character Class:

- I used the character class from project 3 and 4, but with one immense addition. I created a new subclass called "Player" which would have many special properties, yet also still maintain the strength, armor, attack, and defend functions that the other characters have.
- **Player subclass:**
    - Member Variables
        - Item Array: Defined below, the player will be allowed to hold up to 6 items.
        - Number of items: integer denoting the number of items the player is currently carrying
        - Bonus Attack: Total bonus attack from items
        - Second Wind: A Boolean noting if the player has a second wind or not. Similar to Harry Potter, the first time the player reaches 0 health or less they will be able to come back with 10 hp.
        - Player Space: Space pointer for the space that the player is currently occupying.
    - Member Functions
        - Getters and setters for the new variables
        - Item Menu: Menu displaying the items the player currently has
        - Add Item: Add or replace an item in the player's inventory
        - hasKey: Indicates if the player has the key to the boss's room

Item Class:

- Member Variables:
    - None
- Member Functions
    - Print Description: Print a description of the item
    - Use Item: To be called by the player to use an item
    - Equip Item: To be called when player picks up a piece of equipment
    - Remove Item: To be called when player decides to replace a piece of equipment
- **Potion subclass:**

    - Overrides the use function, takes in a Player pointer and increases the player’s HP (strength) by 50%
- **Sword subclass:**
    - Overrides the equip function: increases the player’s bonus attack by 3
    - Overrides the remove function: decreases player’s bonus attack by 3
- **Armor subclass:**
    - Overrides the equip function: increases player’s armor by 3
    - Overrides the remove function: decreases player’s armor by 3
- **Bow subclass:**
    - Overrides the use function: Let’s a player attack an enemy without retaliation.

Spaces Class:

- Member Variables:
    - Top: Pointer to space above it
    - Bottom: Pointer to space below it
    - Left: Pointer to space to the left of it
    - Right: Pointer to space to the left of it
- Member Functions:
    - Setters and getters for each member variable
    - Copy Position: Takes a space pointer and sets the member variables to be the same as that space
    - Action: Special event occurs that occurs when the user moves onto the space.
    - Combat: Fight between a player and an enemy. Player can decide to fight, use an item, or flee.
- **Border subclass**
    - Acts as a border that the player cannot move on to.
- **Blank subclass**
    - A blank space that does not have any special events.
- **itemSpace subclass**
    - Space that holds an item
    - Action: Add’s the item to the user’s inventory
- **keySpace subclass**
    - Space that holds the key needed to unlock the boss’s space.
    - Action: Add’s the key to the user’s inventory
- **Enemy subclass**
    - Action: Forces a fight between the player and a randomly generated enemy (calls combat)
- **Boss subclass**
    - Boss character pointer
    - Action: Forces a fight between the player and the boss (calls combat)

Map Class:

- Member Variables:
    - Data: Pointer to a 2-dimensional array of Space pointers
    - Boss Space: Space pointer for keeping track of the boss space.
    - Player Space: Space pointer for keeping track of the player's space
    - numRow: number of rows in the map
    - numCol: number of columns in the map
- Member functions
    - Constructor: Allocates memory for the map
    - Setters/Getters for all variables
    - Display Map: Display the map
    - Move player space: move's the pointer to the player's space
    - Generate Space: Replace a blank space with a different space.

Testing Plan:

Description	**Functions**	**Precondition**	**Input Value**	**Expected Outcome**	**Actual Outcome**
Combat (Attack, Defend, Use Item, Flee)	Spaces::combat() Character::attack() Character::defend() Player::useItem() Item::use()	Full Items, No Items, Second wind, no second wind	All enemies, randomly generated enemies	Attacks/Defense works correctly Player can use item, and it will be deleted from the inventory Player can flee from normal enemies but not bosses. Player will get a second wind for the first time their health reaches 0 or lower. Fight ends when either player/enemy health reaches 0 or player flees.	Attack/Defense working correctly. Errors in the item menu: Did not display items correctly, included a maxItems constant integer to keep track of the maximum items that the player can keep. Player can flee from enemies, but not the boss. Player gains a second wind when their health

					reaches 0 for the first time, but will not after that.
Movement	setPlayerSpace() Space setters	Spaces at corners of board, middle of board	Move top, bottom, left, right	Player will move in the right direction. Cannot move into a border space, or the boss space without the key	There was a few errors in movement where the player was moving in the wrong direction. Player did not move into border spaces or boss space without the key.
Using/Adding/Replacing items	Item::itemMenu Item::addItem	Full inventory, empty inventory	Key, non-usable items, usable items	Usable items can be used, non-usable items cannot be used. Key cannot be replaced, has to replace another item. When there is no items in the inventory, print empty	Usable items can be used, non-usable items cannot be used. Key cannot be replaced, has to replace another item. When there is no items in the inventory, print empty
Win/Loss Conditions (Boss dies, Player dies, Turns	Player::getStrength Boss:getStrength Game::turns Game::MaxTurns	Player strength reaches 0 or less, Boss strength reaches 0 or less, turn exceeds	No input	Game registers if the user “won”(boss died) or “loses” (player dies, max turns)	Created a function in a new class Game to monitor win/loss conditions. Game correctly registers if

		max turn limit			the user "won"(boss died) or "loses" (player dies, max turns)
Map manipulation/actions	Map::setplayerSpace Map::blankOut Map::generateSpace Space::actions	All spaces	No input	When a player lands on a space, it triggers the space's action. When player leaves the space, the space changes to blank.	Errors happened when coming back to spaces. Changed setters for setting position to also have each space in all directions point back to new space. Changed player space to coordinates rather than Space pointer because there were errors with memory allocation
Map creation	Map::map()	No preconditions	No Input	Map is allocated correctly	Discovered I was allocating spaces for the corners of the map twice.

Reflection:

There were a few errors in my initial design that caused me to heavily revise my program from my initial design. The first issue that gave me a few problems were multiple dependencies between each class. This gave me a few errors where I could not create pointers to a class because it had not been defined yet. I ended up removing some dependencies such as the Character/Item classes having to include the Space class and combining the Character/Item class into one file, an “Objects” class.

Another problem I ran into was having too many pointers to Spaces. For example, when one pointer was updated with a new Space allocation, other pointers were not updated, and instead pointed to an address containing a freed space allocation. To circumvent this issue, I removed a few pointers, such as the playerSpace pointer in the Map class. I instead replaced it with two coordinates with the row/column number, and when referring to the space instead use the coordinates with the Space two-dimensional array. Also, whenever I replaced a space with another space in the map, I also changed the pointer of the space’s around it to point back to the new space.

I also had a problem with the display/use/replacement of items. There were many times where the numbered options did not correspond with their actual selection. In the end, I decided to display the entire menu each time the inventory is accessed and display slots as empty if they do not have anything in them. This way I could have a constant menu size regardless of what the inventory consists of.

I also decided to create a new “Game” class to monitor the status of the game. This class contains Player and Boss pointers to view their health after each “round”. It also contains the current turn and turn limit.