Combined Systems Requirements

By James Hicks and Jonathan Kimbro

Summary :

This is a combined version of all of our system requirements.

1. Menu Requirements
    a. The player will need to be able to select what they would like to do (create a new game or load a previous game).
    b. The player will need to be able to enter the name of the file that they are creating or loading.
2. New Game Requirements
    a. The player will need to be able to enter the name of the file they wish to create.
    b. After the player has chosen a name for that file the machine needs to enter the set amount of data that the player will have at the beginning of the game (what troops they have, how much money they will start with, and what level they will be).
    c. After the game has created the player file it will need to go to the main function so the player can start the game.
3. Load Game Requirements
    a. The player will need to enter the name of the file they previously played on, the machine will then search for the file.
    b. If the file is found the game should take the data from that file and transfer the info into the playerData class which will contain the player's data.
    c. If the file is not found it should tell the player that it could not find the file and send them back to the main menu.
    d. After the file has been found and loaded it should go to the main function.
4. Save Game Requirements
    a. The player should be able to save the file from the main method.
    b. Whenever the player saves from the main method it should find the name of the current game and find the corresponding file.
    c. Once it has found the corresponding file it should overwrite the player data in the file with the player data that is currently in the game.
    d. Once it is done with that it should return to the main method.
5. Player Data Class
    a. The player data class will need to be able to hold all of the values that will need to be carried throughout the game.
    b. It should be able to hold the file name for the game, the amount of money that the player has, the highest number level that they have completed, and a list of troops.

c. Whenever the game starts the playerData class should have all of the information from the file that was loaded or created.
d. Whenever the player saves the game the contents of the playerData class should be saved to the file that the player currently has.

6. Main Method

a. The player should start in the main method.
b. The first thing that the main method should bring up would be the start menu options.
c. After the player has loaded or created their game it should give the player the option to save the game, go to the shop, or quit the game (the battle option will come in a later implementation).
d. The main menu should loop the options until the player has decided to quit.

7. The Troop Class

a. It should contain the following values :
    i. The troops minimum and maximum health
    ii. The amount of spaces that the player can move
    iii. The range that the troop can attack
    iv. The amount of damage that the troop can deal
    v. Their status(dead or alive)
    vi. Their buy and sell value
    vii. Their move status and attack status
    viii. Their x and their y
    ix. The troops troop tag
b. It should have get and set methods for all of these values
c. It should contain two toString() methods(one for the shop and one for the battle field)

8. The Shop Menu

a. The shop menu should be able to access the player's data and tell what troops they have and how much money they have.
b. The shop menu should be able to change the player's data based on whether the player has bought or sold troops.
c. There should be a minimum number of troops that the player can have so that they don't sell too many.
d. There should be a maximum number of troops that the player can have so that they don't have too many.
e. The shop should be able to display the troop's stats and their buy and sell value.
f. The shop should be able to show what troops are available to buy.
g. The shop should be able to show the troops that the player currently has.
h. It should check that the player has enough money to buy the troop that the player has selected and add that troop to the player's data if they do and subtract the money paid from the player data.

i. The shop should be able to check if the player does not go below the minimum number of troops that they can have and if they don't it should remove the sold troop from the player's data and add their sell value to the player's data.

9. The Battle Class

a. The battle class should be able to spawn a random number of enemies based on how many troops the player has.
b. After it has spawned the enemy troops it should spawn the player's troops randomly.
c. It should display the map with the x and y on the top and side of the map.
d. Enemy and Player troops should not be able to be in the same space.
e. The player should be able to choose the action that they would like to use move or attack.
f. After the player has chosen an action they should be able to select a troop to perform that action.
g. The player should be able to cancel the selection of their unit.
h. After they have selected a troop they should be able to enter the location that they would like to do that action.
i. The player's troop should only be able to do one of each actions per turn.
j. The battle should end if all of the enemy troops are dead or all of the ally's troops are dead.
k. The player should be able to end their turn, when the player ends their turn the troops available actions should refresh.
l. After the player ends their turn the enemy should be perform their move and attack action.
m. After the enemies turn the player's turn should start a new turn and repeat until the battle has ended.
n. After the battle has ended the function should return to the main menu and display whether the player has won or lost.