Lee, Melisa
leemeli@oregonstate.edu
CS 162-400
Final Project: Design + Reflection

Program Design:

Theme: The user is Mario and has to defeat Bowser to save Princess Peach.
How to Win: By getting a key to unlock the castle and defeating Bowser.
Goal: Reach the end of the board to save the princess. Requires a key to unlock the castles' door; the key is hidden in one of the spaces. In addition, must defeat Bowser. Bowser starting health is 50; Mario starting health is 15.

- If the user defeats Bowser before reaching the end of the board. As long as the player has the key and reach the end of the board Princess Peach will be saved.
- If the user does not kill Bowser by the time the player reaches the castle. The user must keep attack bowser until he is defeated.

Keep track of which space the player is in- printing text describing where the player is at.
Board- linear board with at least 6 space pointer, and is linked to each other. Board only moves in one direction.

- A menu will be prompt to the user to see what action he wants to make during each turn
    - 1. Open Door to the next space
    - 2. Attack bowser- based on a die roll
    - 3. Sing a Song to Princess Peach- doubles your attack
- First space (where user starts off at) = "lucky space"
    - Weapon is found
    - Buff: deal double damage
- Second and fourth Space are "unlucky spaces"
    - Bowser attacks
    - Finding no item and receives no buff
- Third and fifth space are lucky space.
    - Potion is found
    - Buff: deal double damage
- Sixth Space = castle final boss
    - Potion will be automatically used and hp is recovered to full
    - Buff: deal double damage

Container(allow user to carry items)- weapon(item given on 1st space), key (given on 5th space), potion heals Mario life to full

- Limit 3 items at once, only can have one of each item

Round Limit- 50 round limit

- Each round is based on the user turn
    - Menu is prompt to user
    - Bowser attacks every turn first

- Mario attacks second

Space: abstract class (base)
Member Variables: (private)
Space topPtr = NULL
Space bottomPtr = NULL
Space leftPtr
Space rightPtr
String name // hold name of the space
itemFound // flag if item is found on that space

Member Function: (public)
Virtual Functions
Action ; display action received from space
getItem ; returns unique item id num
Buff ; returns unique buff id num
spaceMenu ; display menu for each space
setLeft
Description: takes a space pointer as a parameter and set protected member var left
Input: nothing
Output: nothing
setRight
Description: takes a space pointer as a parameter and set protected member var right
Input: nothing
Output: nothing
setTop- unused set to NULL
Description: takes a space pointer as a parameter and set protected member var top
Input: nothing
Output: nothing
setBottom- unused set to NULL
Description: takes a space pointer as a parameter and set protected member var bottom
Input: nothing
Output: nothing
Name
Description: return name of space
Input: nothing
Output: return a string representing the name of the space
Left
Description: returns a space pointer to the left
Input: nothing
Output: returns a space pointer to the left
Right
Description: returns a space pointer to the right

Input: nothing
Output: returns a space pointer to the right

randomNum
Description: returns a random integer between two numbers
Input: min and max range of numbers
Output: returns random number between min- max

LuckySpace(base) - moves you to the next space

LuckySpace- constructor
Description: set space name
Input: nothing
Output: nothing

Action
Description: display action received by space
Input: nothing
Output: nothing

getItem
Description: returns a unique item id num for an item
Input: nothing
Output: returns a unique item id num for an item

getBuff
Description: returns a unique item id num for a buff
Input: nothing
Output: returns a unique item id num for a buff

UnluckySpace (base) - allow you to attack Bowser

UnluckySpace- constructor
Description: set space name
Input: nothing
Output: nothing

Action
Description: display action received by space
Input: nothing
Output: nothing

getItem
Description: returns a unique item id num for an item
Input: nothing
Output: returns a unique item id num for an item

getBuff
Description: returns a unique item id num for a buff
Input: nothing
Output: returns a unique item id num for a buff

SingASong (base) - sing a song to the princess that doubles your attack

SingASongconstructor
- Description: set space name
- Input: nothing
- Output: nothing

Action
- Description: display action received by space
- Input: nothing
- Output: nothing

getItem
- Description: returns a unique item id num for an item
- Input: nothing
- Output: returns a unique item id num for an item

getBuff
- Description: returns a unique item id num for a buff
- Input: nothing
- Output: returns a unique item id num for a buff

Character class: (based class)

Member Variables: (private)

Int healthPoints	// store character's healthPoint
Bool alive	// indicate if the character is alive or not
String charName	// store character's name

Member Function: (public)

Character
- Description: this constructor set the private member variable healthPoints
- Input: positive integer
- Output: nothing

getHealthPoints
- Description: returns an integer based on the character's healthPoints
- Input: nothing
- Output: integer representing healthPoints

getCharName
- Description: returns a string based on the character's name
- Input: nothing
- Output: string representing character name

calcAttack - virtual similar concept to previous project
- Description: returns an integer based on the character damage
- Input: nothing
- Output: damage done to another character

calcHP - virtual similar concept to previous project
- Description: returns an integer based on the character hp

Input: integer damage done by another character
Output: integer based on character's current hp

Mario class

calcAttack - overrides base class base on Mario's own attack range
Description: returns an integer based on the character damage
Input: nothing
Output: damage done to another character

Luigi class

calcAttack - overrides base class base on Bowser's own attack range
Description: returns an integer based on the character damage
Input: nothing
Output: damage done to another character

Bowser class

calcAttack - overrides base class base on Bowser's own attack range
Description: returns an integer based on the character damage
Input: nothing
Output: damage done to another character

Game class

Member Variables: (private)

```
Player      ; point to player's character
Boss        ; point to bowser
Space1      ; point to space 1
Space2
Space3
Space4
Space5
Space6
```

Member Function: (public)

displayInstructions
Description: display instructions to users
Input: nothing
Output: nothing

createBoard
Description: create all differents spaces and link them
Input: nothing
Output: nothing

displayBoard
Description: display picture of board(maybe?) or display text describing space

Input: nothing
Output: nothing

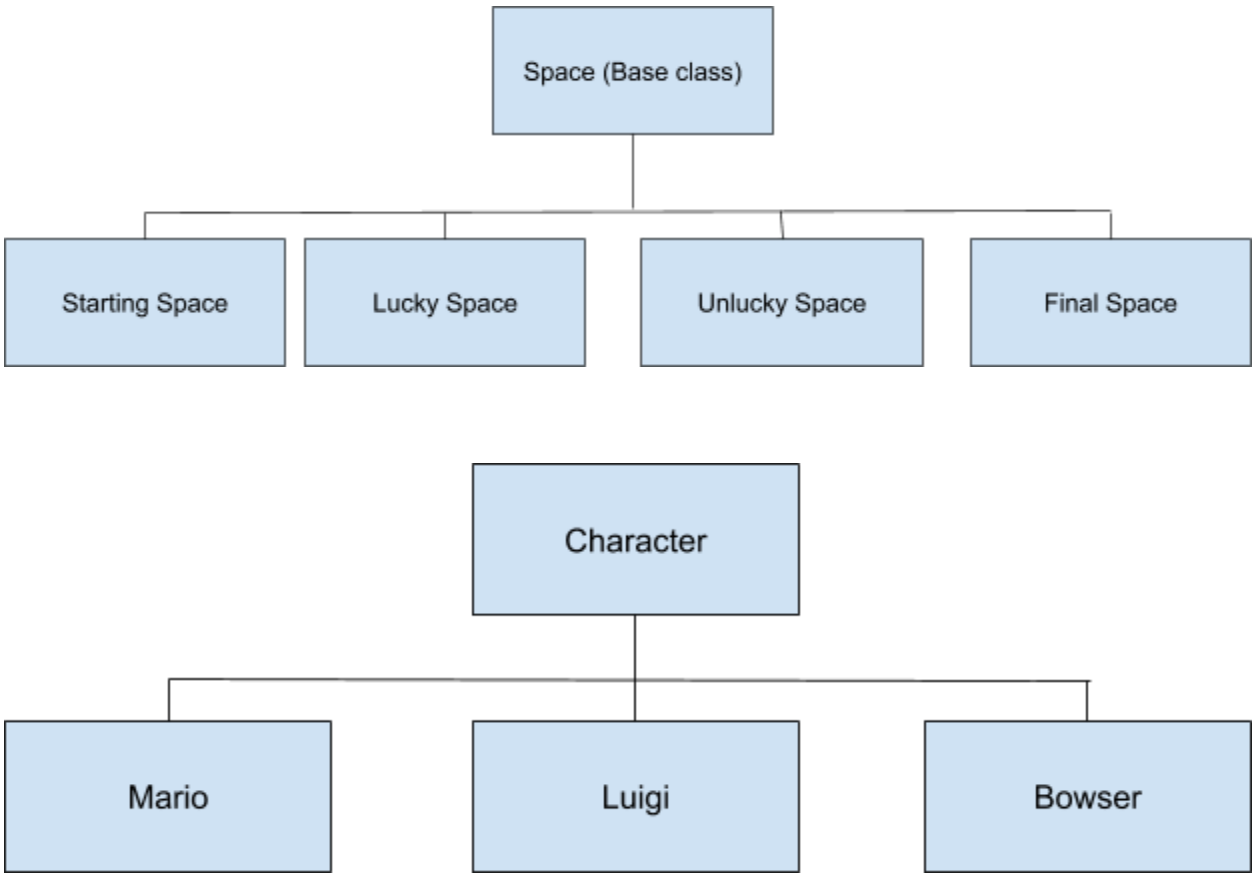
startGame

Description: begins the game my creating the board and running through a menu for the user until both Bowser is defeat and key is found

Input: nothing
Output: nothing

Validate function and Menu function is reused from previous assignments.

Class Hierarchy Diagram



Test Case:

Win Scenarios

Test Case	Input Values	Driver Func	Expected	Observed
Bower is defeat before finding the key	none	startGame(), runSpace(), player->getAlive(), boss->getAlive()	Game continues	Game ended
Key is found Bower is not defeated	none	startGame(), runSpace(), player->getAlive(), boss->getAlive()	Game continues	Game continues
Both key is found and Bowser is defeat	none	startGame(), runSpace(), player->getAlive(), boss->getAlive()	Game ends	Game ends
Player died on unlucky space	none	startGame(), runSpace(), player->getAlive(), boss->getAlive()	Game ends	Game continues

Menu Scenarios

Test Case	Input Values	Driver Func	Expected	Observed
Open Door to the next space	1	startGame(), runSpace(), action(), item(), buff()	User character moves on to the next space	User character successfully move to the space on the right
Attack bowser-based on a die roll	2	startGame(), runSpace(), action(), item(), buff()	User character inflicts damage done to Bowser and stay on the same space	User character successfully deals damage to Bowser
Sing a song to	3	startGame(),	User character	User character

Princess Peach- doubles your attack		runSpace(), action(), item(), buff()	inflicts double damage to Bowser and stay on the same space	successfully deals double damage to Bowser

Reflection

I wanted to build this project off of project's 3 character class hierarchy. Some issues I had was linking all the spaces together and managing the spaces. I ended up creating a current space pointer to help me keep track of player's current space. In addition, I had a hard time linking all the spaces together because I did not create all the spaces in the beginning. I was trying to access spaces that has not been created yet.

Another issue I had was when my character chose to attack Bowser and remained on the current space. Items were added to the bag continuously creating duplicated items. Therefore, I had to add a flag to check if key, potion, and weapons were found. In addition, because there was two lucky space on the board, each time the player landed on a lucky space additional potion was added. If the character landed on the space they received the item and the flag was set to true. If the flag is true, it will not add a repeated item again. Due to not having a flag for finding a key the game ended once Bowser was defeated.

An issue I had was that when the player died before reaching the end of the board, the game would continue. I ended up creating three conditions for the game to run, if endGame flag is false, the key was false, and if the player was alive.

I ended up creating a runSpace function that would direct the user to the correct action when selecting a space menu option. Overall, I felt that my derived space class could be more complex. I felt that majority the character's action were in my Game class instead of my space class. In addition, I could of use a better way to label items and character. The unique identification number could've been used if my project had many more items and buffs. However, it only had a few and seeing numbers made it difficult to keep track and remember each identification number. Another thing I could of change was to not make the character's calcHP function a virtual function because I was not redefined in any way in the derived class. I should've added an armor statistic that could justify creating the virtual function.

Some minor issues/annoyance was that I was using comparison equals (==) instead of equal and was not properly setting or changing my flags.

.