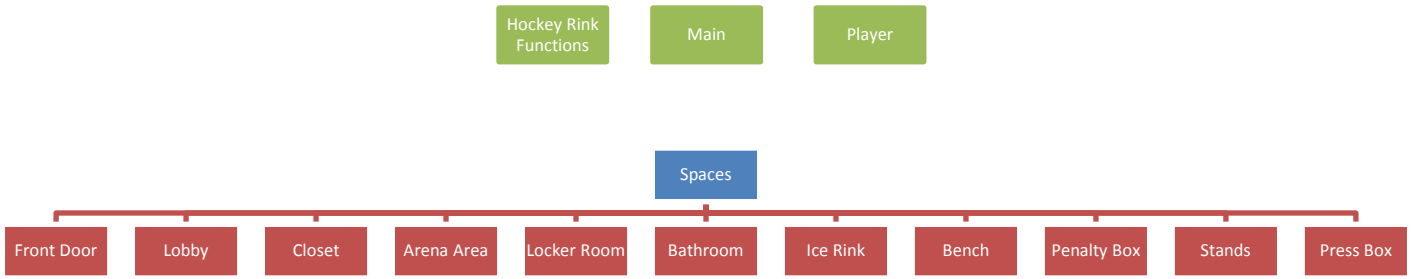

Spaces class(& subclasses) –contains 7 pointers to other "Rooms", the "room" name, the monster Name(if there is one) , the Item Name(if there is one)

Constructor : 3 inputs(room name identifier, item Name, monsterName)

Destructor: destructor of spaces

Int enterRoom: checks to see what is in the room and then displays message based off what is in the room and then returns int that indicates to main what actions to do

Void Set connected rooms: will take "room" inputs and link those rooms to this room

int GetNumConnectedRooms: this returns the number of connected rooms which is used for input validation

intiateSpecial:

Lobby: Turn On Lights

Closet: Opens a trap door…Chainsaw in there

Arena Area: yell…your name echoes

LockerRoom: Play music

Bathroom: Go to the bathroom

Ice Rink: Play Hockey…if you win you get more time…

Bench: you are sitting there collecting dust you bench warmer good work

Penalty Box: Lose 5 turns

Stands: Watch a hockey game

Press Box: Play Music throughout the arena and watch zombies skate in unison to some song

Void displayRoomOptions: display the connected rooms

Void displayItemOptions: display the item option(if there is one)

Spaces* getNewLocation: returns the pointer to the next room you requested

void addItem: puts an item in the room. Used for redistribution of item once dropped from satchel

String getName: this returns the SPECIFIC name of the room used to determine different rooms of the same class so user can see

String getRoomItem: returns the room item so the player can add it to their satchel and then erases it from that room

Int monsterAttack: gets item that the users uses and displays message based off of users chosen item. It will let you know whether the player defeated the monster or not. Then it returns an int which is used as an indicator to main as to what happened during the battle.

Bool hasItem: checks to see if the room has an item. Used for input validation and whether to display the pick up item option

Player Class –contains a pointer to the current location of the player, the number of moves, and a deque which is used as the "satchel"

Player(Spaces*): creates player and sets currentLocation to Spaces* input
~Player(); destructor of players
void addItem(string); adds item to the satchel
string dropItem(); removes item from the satchel
bool satchelEmpty(); checks to see if satchel is empty
bool satchelFull(); checks to see if satchel is full
Spaces* getCurrentLocation(); returns the current Location of the player
void changeLocation(Spaces*); changes the current location of the player
bool hasKey(); checks to see if the player has the key
string useSatchelItems(); allows the user to select an item from the satchel for battle

Other Functions (i.e. HockeyRinkFunctions)

void displayStartInstructions(); displays the starting instructions
void buildHockeyRink(Spaces* *hockeyRink); void DisplayRink(Spaces*); build and links all the rooms creating the "Rink"
void redistributeItem(Spaces* *hockeyRink, string droppedItem); redistributes dropped items randomly through the "rink"

int validateInput(Spaces* currentLocation, int ActionChoice, Player* player); validates user input for actions

---

BEGIN PROGRAM

Initialize variables

Creates “Rooms”

Link “Rooms”

Create Player and assign starting location

Display Game Instructions

LOOP----------

Display Game Map

If monster in room

- Display monster description and options
- Ask user what item they want to use(if they have one)
- Use item
  - If right item Monster dead
  - Else player died GAME OVER

Else no monster in room

- Display room options
- Display item options

Get User input

Validate input

If move rooms

Get new room location

Update players current location

Else if pick up item

Add item to satchel

Else if drop item

Remove item from satchel

Redistribute item to random location in rink

Else If initiate room special

Execute room special action

Increment game clock

EXIT LOOP (If GAME CLOCK is great than allotted time or your location is in the front door which means you win)

Display Game Over or WINNER!

END PROGRAM

TEST PLAN

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Move through all rooms/ensure every link takes you where it is supposed to go	Varied	Spaces::getLocation Player::changeCurrent Location	Never have it take you to a room you didn't tell it to and never have a seg fault	All links worked correctly. No holes or gaps
Input validation	Any option that is not displayed on screen	inputValidation	I cannot choose an option that isn't explicitly displayed to the user	I was not able to choose any options the user could not see
Pick up items	8	Spaces::getRoomItem Player::addItem	Ensure the item moves into the satchel and is removed from the room	Item was moved into the satchel and was removed from the room
Drop Items	9	Player::dropItem redistributeItem	Ensure the item is removed from the satchel and can be found in another room	Item was removed from satchel and distributed into another room
Satchel Empty	9	Player::satchelEmpty	Ensure that drop item option is not displayed when satchel is empty and you cannot drop items	Drop item did not display when satchel was empty and you could not drop an item
SatchelFull	8	Player::satchelFull	Ensure item is not removed from room if satchel is full and lets user know satchel is full if they try and add an item	The item was not removed when satchel was full and displayed error message to user
Use every weapon on every monster	Varied	Player::useItem Spaces::monsterAttack	Ensure certain items defeat certain monsters and ensure it falls in line with what I planned	All weapons worked correctly on each monster
Initialize specials for each room	0	Spaces::initiateSpecial	Ensure each special action works as planned	Each special action worked as planned

Problems & Resolved Problems:

With my initial design, I had the player class and Space class call each other which created a cyclical problem. I had to forward declare the classes so that they could see each other. However, I found that this was not the best method through some research so I decided to change my design so that I did not need to forward declare the classes. This fixed the issues as I was getting major errors before I made the design change.

I struggled with the design change on how to implement it without each class relying on each other and so I decided to have the Spaces functions output integers to help direct my program to do certain tasks such as engage in a fight, implement a special function, or just choose the next room location. I am not sure if I used the most efficient method, however I did get it to work.

I had a few issues with my input validation. I realized my logic was incorrect and I was using OR statements when I should have been using AND statements

Other than that I did not have any major issues. I had multiple spelling errors and missed semicolons which were all easily resolved.