Keisha Arnold
CS 162
Final Project: Design Document

Hint for Garrett:
To win the game, you need to play and win the games in Pairadice and Ixchel, pick up the key item in Tesseract, then go to Camazotz to pick up and open the box containing your body. Picking up the alarm clock in Pairadice sends you to a wormhole which is the "node" that is added and deleted. You only have 8 turns or else it's game over.

Requirements:
- Develop a theme for your game. The player will need to gather items to reach some goal.
- Create a series of rooms or compartments (at least 5 spaces of 3 different types) for the player to move through.
- Each space will be a class with at least 4 pointer variables that link to the other spaces.
- You will have an abstract class that will have a special pure virtual function. This will be redefined in each derived class so each type of space will have their own special action.
- You must have some way to keep track of which space the player is in.
- The player will have a container (backpack, knitting bag, notebook, etc.) to collect items and must have an item limit.
- One or more of these items will be required as part of the solution.
- You will add at least one space and remove at least one space. They do not need to be the same space.
- You should have a time limit to urge the player on. It doesn't need to be a literal clock, just some way to prevent the game from going on indefinitely.
- The player must interact with parts of the structure and not simply collect items.
- Prompt the user to start the game and execute all commands as intended.
- Properly implement some sequence of actions required for the player to exit (other than just moving through the spaces)

Design:
The first step I took was to draft a theme for my game. I loosely designed my game around one of my favorite books "A Wrinkle in Time" and the movie "Interstellar", so if you're familiar with those you may recognize some of the themes and naming conventions.
Theme: You wake up and realize you've lost your body. To win the game you must go to different planets to find your body before you run out of turns and completely disappear.

5 Different Planets/Classes
- Zee: Home planet, peaceful
- Pairadice: Vegas-like planet
- Tesseract: Parallel universe planet (like Zee but not quite)
- Ixchel: Dark planet, bad aliens
- Camazotz: Peaceful planet with large furry aliens
- Wormhole: To add/remove from structure

Classes I'll Need:
Space class (abstract parent class)
Zee (derived class)
Pairadice (derived class)

Tesseract (derived class)
Ixchel (derived class)
Camazotz (derived class)
Wormhole class (derived class)
Dice class (for Pairadice and Player classes?, reuse from Lab B)
Player class (reuse Character class from Assignment 3?)
PlayGame class (to minimize code in main function?)

Rough Design of Classes:

Space class (abstract):
// constructor
+ Space(string);
  - sets locationName

// pure virtual function
+ virtual void alien( ) = 0;
  - allows the player to talk to an alien

// virtual function? maybe pure virtual…
+ virtual playGame( );
  - allows the player to play a game to win an item

// getters and setters
+ virtual void setLocation(string);
+ virtual void getLocation( );

// 4 pointers to link spaces, do these need to be public?
+ Space *right;
+ Space *left;
+ Space *up;
+ Space *down;

// private member variables
- string location;

Zee class (derived from Space class):
*same as the Space class because it inherits all its properties, except…
// redefine pure virtual function
+ virtual void alien();
  - allows player to talk to an “alien” to possibly get clues
  - this alien is your mom (why not, it’s your home planet)

Pairadice class (derived from Space class):
*same as the Space class because it inherits all its properties, except…
// redefine pure virtual function
+ virtual void alien();
  - allows player to talk to an “alien” to possibly get clues

- this alien is the pit boss, he will want you to play a game to get information

// redefine virtual play game function

+ virtual int? playGm( );

- dice game, if you win you get an item needed to win the game

Tesseract class (derived from Space class):

*same as the Space class because it inherits all its properties, except…

// redefine pure virtual function

+ virtual void alien();

- allows player to talk to an “alien” to possibly get clues
- this alien is you, but in a parallel universe (this alien will give you a clue, why be mean to yourself!)

Ixchel class (derived from Space class):

*same as the Space class because it inherits all its properties, except…

// redefine pure virtual function

+ virtual void alien();

- allows player to talk to an “alien” to possibly get clues
- this alien is not nice, perhaps he will attack you? or just not give clues?

// redefine virtual play game function

+ virtual int? playGm( );

- dice game, if you win you get an item needed to win the game

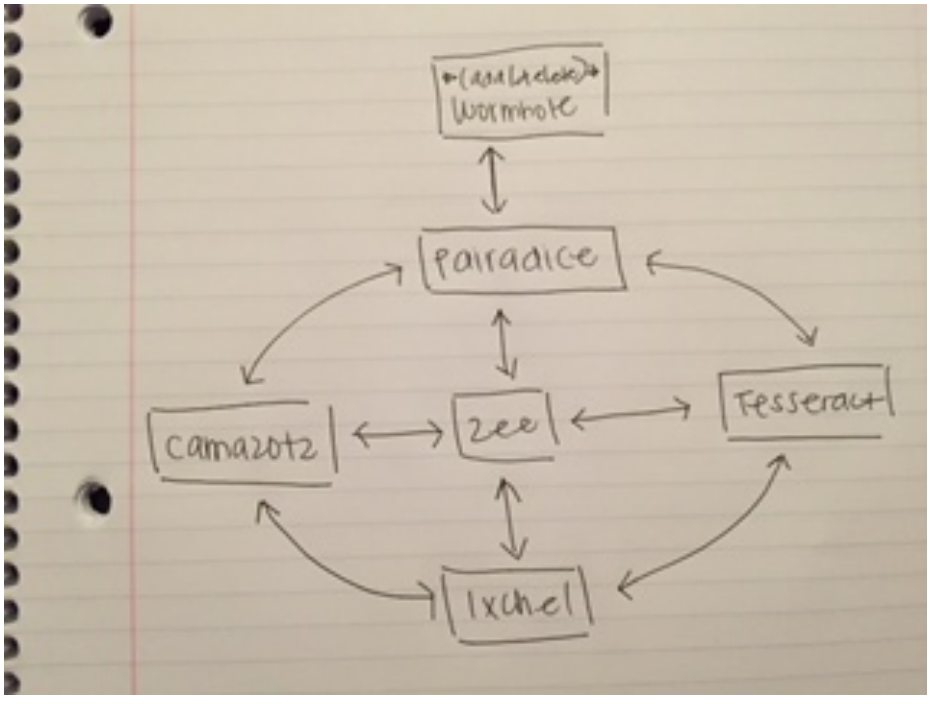
Camazotz class (derived from Space class):

*same as the Space class because it inherits all its properties, except…

// redefine pure virtual function

+ virtual void alien();

- allows player to talk to an “alien” to possibly get clues
- this alien is nice, but probably not helpful



After forming an idea about a theme for my game and the goal I wanted to player to accomplish, I drafted the classes as seen above so I could at least create a Player object (I reused the Character class for this) and move the player through the linked "worlds". I'm not yet taking into account the backpack the player carries or the items he collects because for now I simply want to test the inheritance of my classes, the linked structure, and moving my player through the structure. Incremental development!

How do I link the worlds?

At first I was trying to imitate a linear linked list too closely and was racking my brain trying to figure out how nodes and next/head/back pointers were going to be implemented. I finally realized that the classes themselves were nodes and I just had to use the up, down, left and right pointers to connect them. At first, I linked the world in my main function, but then I realized it was going to get complicated and lengthy quickly so I moved it to my PlayGame class.

How do I move the player between worlds?

Now I have the worlds linked, so how the heck do I move the player between the different worlds? At first I created a Space *current pointer in the Player class as a way to keep track of the player's current location. But then I realized it made more sense in the PlayGame class so that I could access it during the game. Then in the PlayGame constructor I initialized the current pointer to point to Zee, meaning the player would start the game in the Zee world. To test the player moving I simply set current = current->left/right/up/down and displayed the players location.

Once I knew the player was moving through the structure, I figured the easiest way to test and to prompt the user was to have menus at each world so the player could choose which world they wanted to go to. Again, I don't have any special functions or items yet, I just want to test moving through the structure. So I created menus for each world giving the user options as to which world they could jump to. Then I created a function to get the user's choice and another function to move the player's current location based on what they chose by setting current = current->left/right/up/down. Then each time I displayed the player's current location.

Once I was confident the player was moving through the structure correctly, I added the special alien function for each class. I began by just displaying a simple line of code when the function was called, so I knew the polymorphism was working correctly. At this point I had to stop and start thinking about how the player was going to interact with the game beyond moving the player around. I ended up writing a PlayGame function that basically runs the game. It prints the game introduction to set the scene and layout the goals for the player. Then since we start the game in the Zee world it prints the Zee menu, gets the users choice and if needed, moves the players current location. Then based on the player's current location it prints the correct menu, gets the users choice and if needed, moves the players current location. Right now, this repeats until I quit the game just so I can test moving the player and the special alien function. Since there are so many classes and moving parts for the project, it was just easier to create a play function so I didn't have to hard code in the function calls each time.

When I was sure the polymorphism for the alien function was working, I went back and revised each function so that some were mean and didn't give you clues and some were nice and maybe gave you clues.

After that I wanted to get the user to pick up, remove and display items in their inventory. At first, I reused the Item and List classes from Assignment 1, but I ended up just combining these functions into my Player class because the Item class was so short. The final element I implemented was the timer. I simply incremented the turn variable every time the player either picked up an item or played a game. This was just so the game doesn't run indefinitely.

Testing:

As I mentioned above, because I was incrementally designing my program, I was constantly checking if everything was working properly by displaying the players' location, turns, and inventory. That way if something did go wrong I could trace it back to whatever I just added rather than trying to find an error within hundreds of line of code. Therefore, the following table shows the testing I did after I had all the elements working.

Game:

The game always starts in the Zee world. I decided to have an introduction and menu print out each time so the player knows where they were and what options were available to them (pick up item, play game, move to "x" world, etc). The player only has 8 turns until the game ends. (The turn count displays each time).

To win, the player must play and win the games in the Ixchel and Pairadice worlds to collect antidote and a key, respectively. They must also pick up the item (key) in the Tesseract world and go to Camazotz to open the box and revive their body. Picking up an item in Pairadice sends the player to a wormhole and they have to use the item to exit the wormhole (adding/deleting node).

Input Values	**Description of Each Test**	**Expected Output of Each Test**
Player chooses options from the menu that simply move them between worlds ("Open Portal to the Pairadice World"), chooses each world at least once. Player chooses to quit from the menu	Simply tests movement through the structure Tests inheritance of classes Tests the PlayGame class	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location.
Player chooses options from the menu that simply move them between worlds ("Open Portal to the Pairadice World"), chooses each world at least once Player also chooses the option to "Ask someone if they've seen your body" in each world Player chooses to quit from the menu	Simply tests movement through the structure Tests inheritance of classes Tests polymorphism & pure abstract function (alien function) Tests the PlayGame class	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location. Player also chooses the option to ask someone if they've seen your body. This should output a different response depending on what world the player is in demonstrating the polymorphism works.

Input Values	Description of Each Test	Expected Output of Each Test
Player chooses options from the menu that simply move them between worlds (“Open Portal to the Pairadice World”), chooses each world at least once Player also picks up an item in each world (except Zee) Player chooses to quit from the menu	Simply tests movement through the structure Tests inheritance of classes Tests the player class to make sure the addItem and printInv functions are working properly. *Tests the adding and deleting a node (Wormhole) when the user picks up an item in the Pairadice world (50% chance) Tests the PlayGame class	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location. Player also chooses the option to “Pick up item”. Inventory should display that the item was added and only allow the Player to add up to 5 items. It also doesn’t allow the user to pick up the same item twice. *When player picks up the alarm clock in the Pairadice world, it sends them to a Wormhole 50% of the time. Wormhole menu should display and when the user uses their alarm clock they are sent back to Pairadice, deleting the Wormhole.
Player chooses options from the menu that simply move them between worlds (“Open Portal to the Pairadice World”), chooses each world at least once. Player chooses “Play a Game” in the Pairadice and Ixchel worlds. Player chooses to quit from the menu	Simply tests movement through the structure Tests inheritance of classes Tests polymorphism & abstract function (playGame function) Test the PlayGame class Tests the Player class (adding items/displaying inventory).	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location. Player also chooses the option to play a game. The Pairadice game plays an even/odd game and if the player wins they get a gold coin added to their inventory. The Ixchel game is a riddle and gives the user 3 tries to answer correctly. If the user wins, they get antidote added to their inventory.

Input Values	Description of Each Test	Expected Output of Each Test
Player chooses a combination of #2 - #5 and picking up an Item in Pairadice (to add/delete node) Player reaches the 8 turn maximum	Simply tests movement through the structure Tests inheritance of classes Tests the player class to make sure the addItem and printInv functions are working properly. *Tests the adding and deleting a node (Wormhole) when the user picks up an item in the Pairadice world (50% chance) Tests the PlayGame class Tests the timer	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location. *When player picks up the alarm clock in the Pairadice world, it sends them to a Wormhole 50% of the time. Wormhole menu should display and when the user uses their alarm clock they are sent back to Pairadice, deleting the Wormhole. Each time they pick up an item or play a game the timer increments. When the player reaches 8 turns, the game ends.
Win the game Player chooses a combination of #2 - #5 and picking up an Item in Pairadice (to add/delete node) Specifically, the player should play and win both games in the Pairadice class to receive those items, pick up the key item in Tesseract and go to Camazotz to successfully open the box with all these items.	Simply tests movement through the structure Tests inheritance of classes Tests the player class to make sure the addItem and printInv functions are working properly. Tests polymorphism & abstract function (playGame function) *Tests the adding and deleting a node (Wormhole) when the user picks up an item in the Pairadice world (50% chance) Tests the PlayGame class Tests the timer	Player starts in the Zee world, depending on what options they choose from the menu they will move locations and the program will display the correct current location. *When player picks up the alarm clock in the Pairadice world, it sends them to a Wormhole 50% of the time. Wormhole menu should display and when the user uses their alarm clock they are sent back to Pairadice, deleting the Wormhole. Each time they pick up an item or play a game the timer increments. When the player picks up the box they should use the key to open it. The player should then use the antidote and coin to revive the body and win the game.

Input Values	Description of Each Test	Expected Output of Each Test
Same as above, but user doesn't gather the necessary items (e.g. they don't play/win the games in the Pairadice/Ixchel worlds and don't pick up the key in Camazotz)	Simply tests movement through the structure Tests inheritance of classes Tests the player class to make sure the addItem and printInv functions are working properly. Tests polymorphism & abstract function (playGame function) Tests the PlayGame class Tests the timer	Same as above, but when the player picks up the box in Camazotz they can't open it without a key. If they do have a key, they can't revive the body without BOTH the antidote and gold coin. If the player exceeds 8 turns the game ends.