

Doctor Who Inspired Project

Requirements:

1. Parent "Space" class (must be abstract)
 - a. You need 5 Children Space classes
 - i. 3 have to be of different types.
 - b. Each class has to have at least 4 pointers that point to other classes
 - c. Each class will inherit a special pure virtual function.
 - i. They each need to have a special action
2. You must have a goal for the player.
3. You must have a way to keep track of which space the player is in.
4. The player must have a container to carry "items".
 - a. Must have a limit
 - b. One or more of the items should be required as part of the solution.
5. No free-form input
6. Have something to encourage the game on so it doesn't go indefinitely
7. Player must interact with parts of the structure, not just collect things.
8. Game must have an interesting theme.






Idea:

In the year 2016, the Master has come up with another plan for world domination, only this time he is confident his plans will work. To ensure that the Doctor doesn't get in his way, he has wiped his memory and time lord essence, and has sent him far, far away from his paradox hiding all of the objects that could be used to help the Doctor remember who he is and overthrow the Master through all of time and space. These hiding places aren't random, either. With the help of some faithful companions, the Doctor must work and fight his way to collect all of the objects to finally defeat the Master once and for... all?

Specifics:

1. Compartment class (abstract parent) - 'challenge()' function
 - a. Space - points to the other 4 classes
 - i. Based on "Silence in the Library"
 - b. Future - points to the other 4 classes
 - i. Based on "Gridlock"
 - c. Past - points to the other 4 classes
 - i. Based on "Vincent and The Doctor"
 - d. PocketUniverse - points to the other 4 classes
 - i. Based on "Blink" & "Hide"
 - e. ParallelUniverse - points to the other 4 classes
 - i. Based on "The Sound of Drums"

Overview

"Compartment"	Antagonist	Mission/Challenge	Reward
1. Pocket Dimension (PocketUniverse.cpp) Year: 2016.5	Weeping Angel 	Find the screwdriver before the angels get to you. If you roll an odd number, the angels attack. You must roll a set number of points within a set number of rounds	sonic screwdriver + attack die
2. The Past (Past.cpp) Year: 1890	Krafayis 	Defeat the Krafayis by finding them and attacking. They attack back if they find you.	fob watch with Chameleon Arch + defense die
3. New New York (Future.cpp) Year: 5,000,000,053	Cybermen 	Answer the true and false questions correctly to make it out of the gridlock. If you answer a question wrong, the Cybermen attack	Advice from Boe Restore strength
4. The Library (Space.cpp) Year: 2016 ish	Vashta Nerada 	Find the diary before the Vashta Nerada get into your suit by Rolling an 8. You are limited to the number of chances you get.	1200 year old diary + life
5. Parallel Universe (ParallelUniverse.cpp) Year: 0 *Can Only get in if the player has collected all of the items*	The Master 	Defeat the Master & restore the world Dice Battle to the "death"	Win the game

The Details:

virtual

Compartment class (parent class)

Private:

- Compartment *north;
- Compartment *south;
- Compartment *east;
- Compartment *west;
- string roomType; // Each room will have a different name/location
- string year; // Each room takes place in a different year
- Creature *villain; // Each room will have a villain to fight
- string item; // Each room will have a special item for the player to win
- **Int visited;** // Keeps track of if the player can visit the room

Protected:

- Compartment()
 - Default constructor
- ~Compartment()
- void setCompass(Compartment* n, Compartment* s, Compartment* e, Compartment* w)
 - Sets up the game by pointing all of the room's pointers to other rooms.
 - this->north = n;
 - this->south = s;
 - this->east = e;
 - this->west = w;
- Compartment* menu()
 - Displays the different rooms the current room is connected to and allows the user to select which room they want to visit next. Includes input validation.
 - Returns a pointer to the room that they select.
 - usersChoice
 - Print: You're in the TARDIS
 - Print: 1. north->getRoomType();
 - Print: 2. south->getRoomType();
 - Print: 3. east->getRoomType();
 - Print: 4. west->getRoomType();
 - cin >> usersChoice
 - Validate input
 - If 1 = return north;
 - Else if 2 = return south;
 - Else if 3 = return east;
 - Else if 4 = return west;
- string getRoomType()
 - Returns the roomType variable, which stores the string with the room label.
- string getItem()
 - Returns item variable

- `string challenge(Creature* player) = 0`
 - Returns each room's year when the player wins the item to distinguish which item to add to the player's container.
- `string getYear()`
 - Returns year variable.

Future class: public Compartment class

private:

public:

- `Future()`
 - `this->north = NULL;`
 - `this->south = NULL;`
 - `this->east = NULL;`
 - `this->west = NULL;`
 - `roomType = Future "New New York"`
 - `Year = "5,000,000,023";`
 - `this->villain = new Cybermen();`
 - `Item = "Advice from the Face of Boe";`
 - `Visited = 0;`
- `~Future()`
 - Delete villain;
- `String challenge(Creature*) // Trivia Game`
 - If the player hasn't already won this item
 - Print out description & rules.
 - `Int goAgain = 1;`
 - Do (while 'goAgain == 0')
 - Have array with correct answers (`correctAnswers[8] = {0,0,1,1,0,0,1,1}`)
 - Used binary code for '3' since 'Gridlock' is episode 3 of that season.
 - 1 = True
 - 0 = False
 - `Round = 0;`
 - `Limit = 8`
 - `Right = 0; // Keeps track of the number of questions user answered right.`
 - `damage ;`
 - `Inflicted;`
 - `Creature *attacker;`
 - `Creature *defender;`
 - Instruct user to enter 1 for True or 0 for false
 - `while(i < limit && player->getStrength() > 0)`
 - Print: True or False
 - If `i == 0 && player->getStrength() > 0`
 - False Question
 - Else if `i == 1 && player->getStrength() > 0`

- False Question
- Else if i == 2 && player->getStrength() > 0
 - True Question
- Else if i == 3 && player->getStrength() > 0
 - True Question
- Else if i == 4 && player->getStrength() > 0
 - False Question
- Else if i == 5 && player->getStrength() > 0
 - False Question
- Else if i == 6 && player->getStrength() > 0
 - True Question
- Else if i == 7 && player->getStrength() > 0
 - True Question
- Do
 - Print: Your answer:
 - Cin >> usersAnswer;
- while(usersAnswer > 1 || usersAnswer < 0)
- If(usersAnswer == correctAnswers[i])
 - Print: Correct
 - right++
- If(usersAnswer == correctAnswers[i] && i < 7)
 - Print: Next Question
- Else if(usersAnswer != correctAnswers[i])
 - Print: Wrong Answer
 - sleep(2)
 - while(player's strength > 0 && villain's strength > 0)
 - Round++
 - Print: Round
 - Print: Player's strength
 - Print: Villain's strength
 - if(round % 2 != 0)
 - Attacker = player;
 - Defender = villain;
 - Else
 - Attacker = villain;
 - Defender = player;
 - Attacker attacks
 - Defender defends
 - Defender's strength is set
 - If(villain's strength == 0)
 - Print: He's dead
 - Delete villain;
 - Villain = new Cybermen();

- if(i != 7)
 - Print: Moving on
 - Else if(player->getStrength() == 0)
 - Print: That you lost.
 - i++
 - If(player's strength > 0 && they answered all questions right)
 - Print total points rolled
 - Print that they won
 - Set visit = 1
 - Return room's year;
 - Else if (player's strength > 0 && right != 8)
 - Print total points rolled
 - Tell them that they didn't get all of the questions right.
 - Do
 - Do you want to try again? 1 = no & 0 = yes
 - Cin >> goAgain;
 - while(goAgain > 1 || goAgain < 0)
 - while(goAgain == 0 && player->getStrength() > 0)
 - Else
 - Print that you've already been there.
 - Return "1"; // does not match the year.

Future class Tests

Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor Future()	Future fut;	Instantiate an object of the future class with all of the attributes of the Default Constructor	Worked as desired	
setCompass()	future->setCompass(parUn,past,pockUn,space)	getNorth() return 'Parallel Univ.' getSouth()return 'past' getEast() return 'Pocket Dimen.' getWest() Return 'space'	Returned results as expected.	To test if this worked, I used temp function accessors that returned the room type of the objects that were being pointed to. They were deleted after they served their test purpose
menu()	future->menu()	Display: 'You're in the TARDIS' 1. Parallel 2. Past	Displayed as expected and returned the correct	(I abbreviated some of the names to save space. They should return 'Parallel Universe'

		<p>3. Pocket 4. Space</p> <p>Where would you like to go next: Take user's input, validate it, and then return a pointer to the place selected. I.e: 1 would return parUn 2 would return past 3 would return pockD 4 would return space</p>	<p>compartment. It also worked on validating input from the user when different ints were entered.</p>	<p>and 'Pocket Dimension'</p>
Menu::do while loop	<pre>Do{ }while(usersChoice > 4 usersChoice < 1)</pre>	<p>If users enters number greater than 4 or less than 1, it will prompt them to re-enter an option</p>	<p>Re-prompted user to enter a number when it was larger than 4 or less than 0.</p>	<p>I still have trouble with user validation if the wrong data type is entered (ie a char instead of an int) This is one area I need to work on.</p>
menu::if/else statements	<pre>1.usersChoice = 0 2.usersChoice = 1 3.usersChoice = 2 4.usersChoice = 3 5.usersChoice = 4 6.usersChoice = 5</pre>	<pre>1. Go into 'do while' loop and prompt for new input 2. Return parUn 3. Return past 4. Return pockD 5. Return space 6. Go into 'do while' loop and prompt for new input</pre>	<pre>1. Went into loop 2. Returned parUn 3. Returned past 4. Returned pockD 5. Returned space 6. Went into loop</pre>	<p>This function will print different results for each 'Compartment' based on which compartments they are set to point to.</p>
getRoomType()	future->getRoomType()	Return 'future'	Returned 'future'	
getItem()	future->getItem()	Return 'advice from Face of Boe'	Returned 'advice from Face of Boe'	
getYear()	future->getYear()	Return '5,000,000,023'	Returned '5,000,000,023'	
challenge(Creature *player)		<p>If the player survives and answers all 8 questions correctly, it will return the year so they can collect the</p>	<p>Player survived and answered all of the questions correctly, the</p>	<p>I realized that there was no 'end' to the challenge if the player dies. I had to change my code and add features so that if</p>

		item. Else if the player survives but does not answer all of the questions correctly, they can retry the challenge. Else it returns "1". If player dies during challenge, the game ends.	function returned "year". Player survived but didn't answer all of the questions correctly, the player was allowed to either retry the round or go back to the TARDIS (menu). Player did not survive, printed that they lost and ended the game.	they player dies, the game ends.
challenge::do{while(goAgain == 0)	1. goAgain = 1 2. goAgain = 0 3. goAgain = 2	1. 1 == return to menu 2. 0 == play again 3. 2 == check input	1. Returned to menu 2. Played again 3. Prompts user to re-enter input	
Challenge::for(int i = 0; i < limit; i++) while(i < limit && player->getStrength() > 0)	1. I < limit & player lives 2. I = limit and player lives 3. I < limit & player dies 4. I = limit and player dies.	Loops for the limited amount of loops as long as the player is still alive	1. Looped until limit was reached 2. Stopped loop and moved on through function 3. Ended loop 4. Ended loop	
Challenge::if/else if (to print questions)	1: print question 1 2: print question 2 3: print question 3 ... N: print question n 8: print question 8	Each loop will print out a different question.	Worked as expected for all 8 questions.	
Challenge::do{while(usersAnswer > 1 usersAnswer < 0)	1. -1 2. 0 3. 1 4. 2	Validates user's input. If > 1 or < 0, it will loop again	1. Looped again 2. Moved on through function 3. Moved on through function 4. Looped again	

challenge::if(usersAnswer == correctAnswers[i])	1. usersAnswer == 1 & correctAnswers[i] == 1 2. usersAnswer == 0 & correctAnswers[i] == 0	If the user answers the question right, it prints: correct and increments 'right'	Incremented 'right' and printed that they got it correct for both cases.	This is reinforced now that I print how many the user got right out of the 8 at the end of the match.
challenge::if(usersAnswer == correctAnswers[i] && i < 7) Else if(usersAnswer != correctAnswers[i])	1. usersAnswer == correctAnswers[i] & i < 7 2. usersAnswer == correctAnswers[i] & i = 7	Will print 'next question' as long as the user got the question right and i is <7. If user answers wrong, they will fight a cyberman to the death.	1. Printed 'next question' 2. Did not print 'next question'	This was added to let the user know that there is another question to come
challenge::if(villain->getStrength() == 0)	1. Villain's strength = 0 2. Villain's strength != 0	If that Cyberman has no strength, it will delete villain and point it to a new cyberman to fight. Print: Cyberman is dead	1. Printed that cyberman was dead & created new cyberman for next attack 2. Skipped this section	
challenge::while(player->getStrength() > 0 && villain->getStrength() > 0)	1. Player's strength > 0 & villain's strength > 0 2. Player's strength = 0 & villain's strength > 0 3. Player's strength > 0 & villain's strength = 0	Will take turns fighting until one of them is dead.	1. Continues to loop 2. Ended the loop and moved on through the function 3. Ended the loop and moved on through the function	
challenge::if(round%2 != 0)/else	1. Round is even 2. Round is odd	If the round is not even, assign player to attacker and villain to defender. If it is even, assign villain to attacker and player to defender.	1. Villain was attacker, player was defender 2. Villain was defender, player was attacker	
challenge::if(player->getStrength() > 0 && i != 7)	1. Player's strength > 0 & i != 7 2. Player's strength	If player still has strength and they haven't answered the	1. Printed 'next question' 2. Moved on	

	= 0 & i != 7 3. Player's strength > & i = 7	last question, it will print "moving on to next question"	through the function 3. Moved on through the function	
challenge::if(player->getStrength() > 0 && right == 8)/else if(player->getStrength() > 0 && right != 8)	1. Player's strength > 0 & answered all questions correctly 2. Player's strength > 0 & did not answer all questions correctly 3. Player's strength = 0 & did not answer all questions correctly.	If the player survives and answers all questions correctly: print that they won, set visit = 1, and return room's year.	1. Printed that they won & allowed them to collect the item. 2. Let them retry the challenge 3. Moved through the function and ended the game.	

ParallelUniverse class: public Compartment class

Private:

Public:

- ParallelUniverse();
 - this->north = NULL;
 - this->south = NULL;
 - this->east = NULL;
 - this->west = NULL
 - roomType = "Parallel Universe"
 - Year = "0";
 - this->villain = new TheMaster();
 - item = " ";
 - Visited = 0
- ~ParallelUniverse();
 - Delete villain;
- String challenge(Creature*)
 - if(visited == 0)
 - Print description
 - String line;
 - Ifstream myfile(Master.txt)
 - if(myfile is open)
 - Print: enter to continue
 - while(getline(myfile, line))
 - Cin.get
 - Print: line
 - Close file

- Else
 - Print: error unable to open.
- Int round = 0;
- Int damage;
- Int inflicted;
- Creature *attacker;
- Creature *defender;
- Print: press enter to start attack
- cin.get()
- while(player's strength > 0 && villain's strength > 0)
 - Round++
 - Print attack
 - Print player's strength
 - Print villain's strength
 - if(round % 2 != 0)
 - Attacker = player
 - Defender = villain
 - Else
 - Attacker = villain
 - Defender = player
 - Print who is attack who
 - Attacker attacks
 - Defender defends
 - Defender's strength is set
 - sleep(2)
- if(player's strength > 0)
 - Print: You won!
 - Visited = 1;
 - Print win.txt
 - Ifstream winnerfile("Win.txt")
 - if(file is open)
 - Print: press enter to continue through text
 - while(getline(winfile, line))
 - Cin.get
 - Print: line;
 - Close file
 - Else
 - Print: error, unable to open.
- Return year;
- Else
 - Print lost.txt
 - Ifstream loserfile("Lost.txt")
 - if(file is open)

- Print: press enter to continue through text
- while(getline(loserfile, line))
 - Cin.get
 - Print: line;
- Close file
- Else
 - Print: error, unable to open.
- Else
 - Print: You've already been here.
- Return "1"

ParallelUniverse class Tests				
Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor ParallelUniverse()	ParallelUniverse parUn;;			
setCompass()	parUn->setCompass(past,pockUn,space,fut)	getNorth() return 'past' getSouth() Return 'Pocket Universe' getEast() return 'Space' getWest() return 'Future'	Returned results as expected.	To test if this worked, I used temp function accessors that returned the room type of the objects that were being pointed to. They were deleted after they served their test purpose
menu()	parUn->menu()	Display: 'You're in the TARDIS' 1. Past 2. Pocket Universe 3. Space 4. Future Where would you like to go next: Take user's input, validate it, and then return a pointer to the place selected. I.e: 1 would return past 2 would return	Displayed as expected and returned the correct compartment. It also worked on validating input from the user when different ints were entered.	(I abbreviated some of the names to save space. They should return 'Parallel Universe' and 'Pocket Dimension')

		Pocket Univ. 3 would return Space 4 would return Futre		
Menu::do while loop	Do{ }while(usersChoice > 4 usersChoice < 1)	If users enters number greater than 4 or less than 1, it will prompt them to re-enter an option	Re-prompted user to enter a number when it was larger than 4 or less than 0.	I still have trouble with user validation if the wrong data type is entered (ie a char instead of an int) This is one area I need to work on.
menu::if/else statements	1.usersChoice = 0 2.usersChoice = 1 3.usersChoice = 2 4.usersChoice = 3 5.usersChoice = 4 6.usersChoice = 5	1. Go into 'do while' loop and prompt for new input 2. Return past 3. Return space 4. Return parUn 5. Return pockD 6. Go into 'do while' loop and prompt for new input	1. Went into loop Returned past 2. Returned space 3. Returned parUn 4. Returned pockD 5.1. Went into loop	
getRoomType()	future->getRoomT ype()	Return 'future'	Returned 'future'	
getItem()	future->getItem()	Return 'advice from Face of Boe'	Returned 'advice from Face of Boe'	
getYear()	future->getYear()	Return '5,000,000,023'	Returned '5,000,000,023'	
challenge(Creature *player)	If player beats challenge, returns the year of the room. Else if player survives but doesn't beat challenge, returns "1". Else if player dies, ends the game.	1. Player beats challenge, will return year. 2. Player survives but doesn't win, gets to loop back through and try again. 3. The game ends.	1. Player won, year was returned, item was added to their TARDIS. 2. Player survived, didn't win, it offered choice to try again. 3. The player died and the game ended.	
challenge::if(visited == 0)	1. Visited == 0	1. It will go through	1. Allowed player	

else	2. Visited == 1	the challenging part of the function. 2. It will print that you've already been there.	to play out the challenge 2. It printed that they had already been there.	
challenge::if(myfile.is_open()); else	If ifstream myfile("Master.txt"), it will open. Else, it will print an error.	1. Will print the content of 'Master.txt'. 2. It will print an error.	1. Printed the content of 'Master.txt' 2. Printed an error.	
challenge::while(getline(myfile,line))	ifstream myfile("Master.txt")	Pressing "enter" will play the text file	1. You press 'enter' it continues through loop until end of file is reached	It did as expected, and was very magical. I forgot that I didn't have to compile the text file for changes to appear, so you could edit as it went before it was printed to the console.
challenge::while(player->getStrength() > 0 && villain->getStrength() > 0)	1. Player & villain both have strength > 0 2. player's strength = 0, villain's strength > 0 3. Player's strength > 0, villain's strength = 0	Will loop until either the player or the villain does not have any strength left.	1. Continued to loop and allowed creatures to attack 2. Quit looping and finished through the remainder of the function. 3. Quit looping and finished through the remainder of the function.	
challenge::if(round % 2 != 0) else	1. Round is even 2. Round is odd	If round is odd, player will be attacker and villain will be defender. Else, player will be defender and villain will be attacker.	1. Player was defender and villain was attacker 2. Player was attacker and villain was defender	
challenge::if(player->getStrength() != 0)/Else	1. Player's strength > 0 2. Player's	If Player's strength > 0...Visited = 1; Print 'Win.txt'.	1. Visited was set to 1 and it printed the 'Win.txt'	

	strength = 0	Else Print 'Lost.txt'	context 2. Printed 'Lost.txt' and gave option to restart the game.	
challenge::if(winnerfile.is_open()) else	1. Winner file is there to open 2. Winner file can't be opened	If the file is there to open, it will print it. Else, it will print an error.	1. File printed to screen 2. Error message printed.	
challenge::if(loserfile.is_open())/else	1. Loser file is there to open 2. Loser file can't be opened	If the file is there to open, it will print it. Else, it will print an error.	1. Loser file was printed to the screen. 2. Error message was printed.	
while(getline(winnerfile, line))	Win.txt	It will print the entire Win.txt file.	Printed the entire file.	
while(getline(loserfile, line))		It will print the entire Lost.txt file.	Printed the entire file.	

Past class: public Compartment class

Private:

Public:

- Past()
 - this->north = NULL;
 - this->south = NULL;
 - this->east = NULL;
 - this->west = NULL;
 - roomType = "The Past"
 - Year = "1890"
 - this->villain = new Krafayis();
 - Item = "fob watch with Chameleon Arch"
 - Visited = 0;
- ~Past()
 - Delete villain;
- String challenge(Creature *player)
 - If you haven't won this round yet
 - Print: description
 - Int goAgain = 1;
 - Do
 - Int round = 0;
 - Int damage;
 - Int inflicted;

- Creature *attacker;
- Creature *defender;
- Int playerRoll, villainRoll;
- while(player's strength && villain's strength are not 0)
 - Print: Press enter to fight
 - Cin.get
 - Round++
 - Print: Attempt
 - Print player's strength
 - Print villain's strength
 - Do
 - playerRoll = player->attack();
 - villainRoll = villain->attack();
 - while(playerRoll == villainRoll)
 - if(player's roll > villain's roll)
 - Print: player found the villain
 - Defender = villain
 - Attacker = player
 - Else if(player's roll < villain's roll)
 - Print: villain found the player
 - Defender = player
 - Attacker = villain
 - Else if(player's roll == villain's roll)
 - Print: the villain overpowers the player
 - Defender = player
 - Attacker = villain
 - Print out who is attack who
 - Attacker attacks
 - Defender defends
 - Defender's strength is set
- if(player's strength > 0)
 - Visited = 1;
 - Return year;
- Else
 - You die and game ends.
- Do
 - Prompt user to enter 1 to return to the TARDIS or 0 to try again.
 - Cin >> goAgain;
 - while(goAgain > 1 || goAgain < 0)
 - while(goAgain == 0 && player->getStrength() > 0)
- Else
 - Print that you've already been there

- Return "1"

Past class Tests

Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor Past()	Past past;			
challenge(Creature *player)				
challenge():if(visite d == 0)/Else	1. Visited = 0; 2. Visited = 1;	1. Will play through the challenge 2. Else it will print that you've already been there	1. Let the player play through the challenge. 2. Printed they had already been there.	
challenge():do{whil e(goAgain == 0 && player->getStrength () > 0)	1. goAgain = 0 && player's strength > 0 2. goAgain = 0 & player's strength = 0; 3. goAgain = 1 & player's strength = 0; 4. goAgain = 1 & player's strength > 0	1. Will let player play through challenge again. 2. Will go to the end of the function/game. 3. Will go to the end of the function/game. 4. Will take player to the menu	1. Let player try again 2. Ended the game 3. Ended the game 4. Printed the menu for the user to select from.	
challenge():while(pl ayer->getStrength() > && villain->getStrength() > 0)	1. Player & villain's strength > 0 2. Player's strength = 0, villain's > 0 3. Player's strength > 0, villain's strength = 0.	1. Continue to loop 2. Loop will stop and game will end 3. Loop will stop and it will move on through the function.		
challenge():do{whil e(playerRoll == villainRoll)	1. playerRoll > villainRoll 2. playerRoll < villainRoll 3. playerRoll ==	1. Will break the loop 2. Will break the loop 3. Will loop back through until game is over		

	villainRoll			
challenge()::if(playerRoll > villainRoll)/else if(playerRoll < villainRoll)/else if(playerRoll == villainRoll)	1. playerRoll > villainRoll 2. playerRoll < villainRoll 3. playerRoll == villainRoll	1. Print player finds villain. Player == attacker, villain == defender 2. Print villain finds player. Player == defender. Villain == attacker. 3. Print villain overpowers player. Villain == attacker and player == defender	1. Player was set as attacker, villain set as defender. 2. Player was set as defender, villain set as attacker. 3. Player was set as defender, villain was set as attacker.	I left the case in which they both were equal in there because I was having seg faults for some strange reason. I also have the creatures rerolling their attack die to get a new int instead of using the winning int for the same reason. Couldn't figure that issue out.
challenge()::if(player->getStrength() != 0) else	1. player->getStrength() != 0 2. player->getStrength() == 0	1. Visited = 1, print winning script, return year. 2. Print that you died and end game.	1. Printed the winning script and set visited to 1. 2. Printed that you died and ended the game.	

PocketUniverse class:public Compartment class

Private:

Public:

- PocketUniverse()
 - this->north = NULL
 - this->south = NULL
 - this->east = NULL
 - this->west = NULL
 - roomType = "Pocket Dimension"
 - Year = 2016.5
 - this->villain = new WeepingAngels()
 - Item = "Sonic Screwdriver";
 - Visited = 0
- ~PocketUniverse()
 - Delete villain
- String challenge(Creature *player)
 - If the player hasn't visited more than 3 times
 - Print: description
 - Int goAgain = 1;
 - Do
 - Int round = 0;

- Int Damage = 0;
- Int inflicted = 0;
- Int totalDieCount = 0;
- Limit = 10;
- Int roll = 0;
- Int maxCount = 70;
- Do
 - Print: Press enter to roll
 - Cin.get
 - Round++
 - Print: Chance
 - Print: Total Points earned thus far
 - Print Player's strength
 - Roll = player->attack()
 - totalDieCount += roll;
 - if(player rolls an odd number)
 - Print that they blinked
 - Print that the villain gets closer to them
 - Villain attacks player
 - Player defends
 - Player sets strength
- while(player's strength > 0 && round < limit && totalDie < maxCount)
- Visited++;
- if(totalDieCount >= maxCount)
 - Visited = 4;
 - Return year;
- Else if(visited < 3)
 - Do
 - Prompt user to press 1 to leave or 0 to play again
 - Cin >> goAgain;
 - while(goAgain > 1 || goAgain < 0);
- Else if(visited == 3)
 - Print that the pocket dimension is closing
 - while(goAgain == 0 && visited < 3)
- Else
 - Print that you've already been there
- Return "1"

PocketUniverse class Tests				
Test Subject	Input/Expressio	Expected Results	Observed Results	Comments

	n			
Constructor PocketUniverse()	PocketUniverse pockU;			
challenge(Creature *player)				
challenge():if(visited < 3) else	1. Visit < 3 2. Visit == 3 3. Visit > 3	1. It will let you play through the game 2. It will print else statement 3. It will print else statement	1. Let the player return to play for a total of 3 times. 2. Printed the warning statement 3. Printed that you couldn't enter	
challenge():do{}while(go Again == 0 && visited < 3 && player->getStrength() > 0)	1. goAgain == 0 & visited < 3 & player has strength 2. goAgain = 1, visited < 3, player has strength 3. goAgain = 0, visited = 3, & player has strength 4. goAgain = 0, visited < 3, & player's strength = 0.	1. Will let player play through challenge a total of 3 times. 2. Will print the menu for the player to select a new location. 3. Will say that the pocket dimension is closing. 4. Will end the game.	1. Let the player play through again for a total of 3 times. 2. Printed the menu for the player to select from. 3. Printed closing warning. 4. Ended the game.	
challenge():do{}while(player->getStrength() > 0 && round < limit && totalDieCount < maxCount)	1.player->getStrength > 0, round < limit, totalDieCount < maxCount 2.player->getStrength = 0, round < limit, totalDieCount < maxCount 3.player->getStrength > 0, round = limit, totalDieCount < maxCount	1. Will let the loop continue 2. Will end the game 3. Will move on through the function. 4. Will move on through the function.	1. Continued through the loop 2. Ended the game. 3. Moved on through the function based on the other criteria 4. Moved on through the function based on the other criteria.	

	4.player->getStrength > 0, round < limit, totalDieCount = maxCount			
challenge()::if(roll % 2 != 0)	1. Roll = odd 2. Roll = even	1. Villain will attack 2. Villain won't attack	1. Villain attacked. 2. Nothing happened.	
challenge()::if(player->getStrength() > 0)/else	1.player->getStrength() > 0 2. player->getStrength() == 0	1. Will test if totalDieCount >= maxCount, else if visited < 3, else if visited == 3. 2. Will print that they lost and end the game	1. Did as expected. 2. Ended the game.	
challenge()::if(totalDieCount >= maxCount)/else if(visited < 3)/else if(visited == 3)	1.totalDieCount >= maxCount 2.totalDieCount < maxCount && visited < 3 3.totalDieCount < maxCount && visited == 3	1. Visited = 4; return year; 2. Ask them if they want to try again or exit to main game 3. Warn that the pocket dimension is closing	1. Did as expected and let the player add the item. 2. Let user try again or see the menu 3. Warned that it was closing.	
challenge()::do{}while(goAgain > 1 goAgain < 0)	1. goAgain = 1 2. goAgain = 0 3. goAgain = 2	1. Breaks loop 2. Breaks loop 3. Loops again	1. Printed the menu for the user to select from 2. Let the user replay the challenge if visit < 3	This is so repetitive, it could be moved to a function in the Compartment class. (This is such an afterthought, it isn't even funny.)

Space class:public Compartment class

Private:

Public:

- Space()
- ~Space()
 - Delete villain
- String challenge(Creature *player)
 - If visited == 0
 - Print: description
 - Int GoAgain = 1;

- Do
 - Int round = 0;
 - Int damage;
 - Int inflicted;
 - Int attemptForDiary;
 - Int limit = 5;
 - while(player's strength > 0 && round hasn't met limit)
 - Print: press enter to continue
 - cin.get()
 - attemptForDiary = player's attack roll
 - if(attemptForDiary == 10)
 - Mark as visited
 - Return year
 - Else
 - Print: villain attacks player
 - Villain attacks
 - Player defends
 - Player sets strength
 - Do
 - Print: 1 to return to TARDIS or 0 to play again.
 - Cin >> goAgain
 - while(goAgain > 1 || goAgain < 0)
 - while(goAgain == 0)
 - Else
 - Print: You've already been here
 - Return "1";

Space class Tests

Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor Space()	Space space;			
challenge()::if(visited == 0)/else	1. Visited = 0 2. Visited = 1	1. Will let player play the challenge 2. Will print that the player has already been there.	1. Let the player play the challenge through 2. Printed that they couldn't go back	
challenge()::do{}while(goA gain == 0 && player->getStrength() > 0)	1.goAgain = 0, player's strength > 0 2. goAgain = 1,	1. Let the player replay the challenge. 2. Will print the	1. Did as expected. 2. Printed the menu as expected 3. Ended the game.	

	player's strength > 3. goAgain = 0, player's strength = 0	menu for the user to change locations. 3. Will end the game.		
challenge():while(player->getStrength() > 0 && round < limit)	1. Player's strength > 0 & round < limit 2. Player's strength = 0 & round < limit 3. Player's strength > 0, round = limit 4. Player's strength = 0 & round = limit	1. Will continue to go through loop 2. Game will end 3. Will continue with the function 4. Game will end.	1. Looped the loop 2. The game ended 3. Moved on from the loop through the rest of the function. 4. Game ended	
challenge():if(attemptForDiary == 10)/else	1. attemptForDiary == 10 2. attemptForDiary != 10	1. Will return the year and marked the place as visited. 2. Will let the player retry to roll a 10 (depending on what round they are on).	1. Did as expected. 2. Let player go again (when it wasn't their last chance).	
challenge():if(player->getStrength() > 0)				
challenge():do{}while(goAgain > 1 goAgain < 0)	1. goAgain = 1 2. goAgain = 0 3. goAgain = 2	1. Breaks loop 2. Breaks loop 3. Loops again	1. Printed the menu for the user to select from 2. Let the user replay the challenge if visit < 3	

Creature class

Protected:

- String creatureType
- Int attackDieCount
- Int attackDieSides
- Int defenseDieCount;
- Int defenseDieSides
- Int armor
- Int strength

Public:

- Creature()
- Virtual int attack()
 - Int sumOfDice = 0;
 - Int loop = 0;
 - while(loop < attackDieCount)
 - sumOfDice += (rand() % attackDieSides) + 1;
 - Loop++;
 - Print: creature rolled sumOfDice
 - Return sumOfDice
- Virtual int defense(int damage)
 - Print: number of defense die count.
 - Int sumOfDice = 0;
 - Int loop = 0;
 - while(loop < defenseDieCount)
 - sumOfDice += (rand() % defenseDieSides) + 1;
 - Loop++;
 - Print: what creature rolled for defense
 - if(damage <= sumOfDice)
 - Print: attack failed
 - Return 0
 - Return (damage - sumOfDice);
- Virtual void setStrength(int inflicted)
 - if(inflicted > 0)
 - Print: the armor of the creature
 - Int appliedDamage;
 - appliedDamage = inflicted - armor;
 - Print applied damage
 - if(appliedDamage < 1)
 - Print that the creature was protected
 - Else if(appliedDamage >= strength)
 - Print that creature takes a fatal hit
 - Set strength = 0;
 - Else
 - Strength -= appliedDamage;
 - Print strength
- Virtual string getType()
 - Returns creatureType;
- Virtual int getStrength();
 - Returns strength;

Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor()	Creature creat;			
Virtual int attack()	creat->attack()	Will return an int representing the number "rolled" (damage)	Returned an int within the appropriate range for the creature	This was tested for all of the derived classes, as well.
attack():while(loop < attackDieCount)	1. Loop < attackDieCount 2. Loop = attackDieCount	1. Will continue to loop 2. Will break the loop and continue with function.	1. Continued to do it's loop thang. 2. Broke loop and moved on through function	
Virtual int defense(int damage)	creat->defense(damage)	Will return an int representing the total inflicted damage on the creature.	Returned the int within the appropriate range for the creature.	This was also tested for all of the derived classes.
defense::while(loop < defenseDieCount)	1. Loop < defenseDieCount 2. Loop = defenseDieCount	1. Will continue to loop 2. Will break the loop and process the remaining portion of the function.	1. Continued to do it's loopy thing. 2. Broke free and did its thing to return an int (inflicted)	
defense::if(damage <= sumOfDice)	1. Int <= sumOfDice 2. Int > sumOfDice	1. Will print "attack failed" and return 0. 2. Will return int - sumOfDice (i.e. will skip the if statement)	1. Worked as was anticipated. 2. Worked as was anticipated with correct math.	
Virtual void setStrength(int inflicted)	creat->setStrength(int)	Will accept the int and assign a value to strength depending on certain requirements that are met.	Accepted the int and assigned the strength accordingly.	
setStrength::if(appliedDamage < 1)/else if(appliedDamage >= strength)/else	1. appliedDamage < 1 2. appliedDamage >= strength 3. appliedDamage >	1. Print that the creat was protected 2. Print that it was a fatal attack and set strength = 0; 3. Subtract appliedDamage from strength	1. The creat was protected 2. The creat died. 3. The creat's strength was reduced.	

	0 && appliedDamage < strength			
Virtual string getType()	creat->geType	Will return the creatureType for the creature	Returned the creatureType	
Virtual int getStrength();	creat->getStrengt h()	Will return the current strength of the creature	Returned the current strength of the creature (as expected).	

Cybermen class:public Creature

Private:

Public:

- Cybermen();
 - this->creatureType = "Cybermen"
 - this->attackDieCount = 2;
 - this->attackDieSides = 10;
 - this->defenseDieCount = 1;
 - this->defenseDieSides = 6;
 - this->armor = 2;
 - this->strength = 10;

Krafayis class:public Creature

Private:

Public:

- Krafayis()
 - this->creatureType = "Krafayis"
 - this->attackDieCount = 1;
 - this->attackDieSides = 10;
 - this->defenseDieCount = 1;
 - this->defenseDieSides = 6;
 - this->armor = 3;
 - this->strength = 10;

TheMaster class:public Creature

Private:

Public:

- TheMaster()
 - this->creatureType = "The Master"
 - this->attackDieCount = 2;
 - this->attackDieSides = 10;

- this->defenseDieCount = 2;
- this->defenseDieSides = 6;
- this->armor = 5;
- this->strength = 20;

VashtaNerada class:public Creature

Private:

Public:

- VashtaNerada()
 - this->creatureType = "Vashta Nerada"
 - this->attackDieCount = 2;
 - this->attackDieSides = 10;
 - this->defenseDieCount = 0;
 - this->defenseDieSides = 0;
 - this->armor = 0;
 - this->strength = 0;

WeepingAngels class:public Creature

Private:

Public:

- WeepingAngels()
 - this->creatureType = "Weeping Angels"
 - this->attackDieCount = 2;
 - this->attackDieSides = 6;
 - this->defenseDieCount = 1;
 - this->defenseDieSides = 6;
 - this->armor = 6;
 - this->strength = 3;

Player class:public Creature

Private:

- deque<string> TARDIS
- Int lifeCounter
- Compartment* you_are_here;
- Int maxStrength;
- String outfit
- String userName;
- Int itemCount
- Bool advice
- Bool watch
- Bool screwdriver
- Bool diary

Public:

- Player()
 - srand(time(NULL))
 - this->attackDieCount = 2;

- `this->attackDieSides = 6;`
- `this->defenseDieCount = 2;`
- `this->defenseDieSides = 6;`
- `this->armor = 5;`
- `this->strength = 10;`
- `this->maxStrength = 10;`
- `this->lifeCounter = 3;`
- `this->outfit = "leather jacket with a t-shirt";`
- `this->you_are_here = NULL;`
- `this->advice = false;`
- `this->screwdriver = false;`
- `this->watch = false;`
- `this->diary = false;`
- **Do**
 - `Print: enter your name:`
 - `getline(cin, creatureType);`
- **while(creatureType == "");**
- **Void setStrength(int inflicted)**
 - `if(inflicted > 0)`
 - `Int appliedDamage`
 - `appliedDamage = inflicted - armor;`
 - `Print: the potential damage`
 - `if(appliedDamage < 1)`
 - `Print that the player dodged the damage`
 - **Else if(appliedDamage >= strength)**
 - `Print that it was a fatal hit.`
 - **Set strength = 0**
 - `lifeCounter --;`
 - `setOutfit()`
 - `if(lifeCounter == 2)`
 - `Print that you're regenerating`
 - `Set outfit`
 - `Set strength to max strength`
 - `sleep(2)`
 - `Print new outfit`
 - **Else if(lifeCounter == 1)**
 - `Print that you're regenerating`
 - `Set outfit`
 - `Set strength to max strength`
 - `sleep(2)`
 - `Print new outfit`
 - **Else**
 - **Set strength = 0;**

- Else
 - Strength -= appliedDamage;
 - Print Creature's strength.
- Void addItem(Compartment*c)
 - String i = (compartment->getItem())
 - Print that you have the item
 - Add it to the back of the TARDIS
- String removeItem()
 - String removed
 - If the TARDIS isn't empty
 - Copy the front to removed
 - Pop the front off
 - Add the front back to the end (to recycle it)
 - Return removed;
- Void setLocation(Compartment*)
 - You_are_here = c;
- Void setOutfit()
 - if(lifeCounter == 4)
 - Outfit = "big, floppy, grey cravat and silky waistcoat"
 - Else if(lifeCounter == 3)
 - Outfit = "leather jacket with t-shirt"
 - Else if(lifeCounter == 2)
 - Outfit = "brown trench coat and suit"
 - Else if(lifeCounter == 1)
 - Outfit = "red fedora and bowtie"
- Compartment* getLocation()
 - Return you_are_here;
- String getOutfit()
 - Return outfit;
- Void setCreatureType(string n)
 - creatureType = n;
- String getType()
 - Return creatureType;
- Void setAdvice()
 - Print script about seeing the Face of Boe and collecting the advice
 - Advice = true
- Bool getAdvice()
 - Return advice;
- Void setScrewdriver()
 - Print script about getting your screwdriver back.
 - Screwdriver = true
- Bool getScrewdriver()
 - Return screwdriver;

- Void setWatch()
 - Print script about getting the watch back.
 - Watch = true;
- Bool getWatch()
 - Return watch;
- Void setDiary()
 - Print script about finding the diary
 - Diary = true;
- Bool getDiary()
 - Return diary
- Int getLifeCounter()
 - Return lifeCounter;
- Void setItems(string s, Compartment* c)
 - if(s == "5,000,000,023")
 - setAdvice()
 - addItem(c)
 - Else if(s == "1890")
 - setWatch();
 - addItem(c);
 - Else if(s == "2016.5")
 - setScrewdriver();
 - addItem(c);
 - Else if(s == "2016(ish)")
 - setDiary();
 - addItem(c);
- Int attack();
 - Int sumOfDice = 0;
 - Int loop = 0
 - String front;
 - Front = removeItem()
 - while(loop < attackDieCount)
 - sumOfDice += (rand() % attackDieSides) + 1;
 - Loop++
 - Print what player rolled
 - If(current room == "Parallel Universe")
 - if(advice == true && sumOfDice == 3 && front == "Advice from the Face of of Boe")
 - Advice = false; // keeps it from being able to repeat
 - Restore max strength
 - Print that you got the advice and built strength
 - Else if (screwdriver == true && sumOfDice == 6 && front == "Sonic Screwdriver")
 - Screwdriver = false;

- attackDieCount += 1;
- Print that you got extra attack die due to screwdriver
- Else if(watch == true && sumOfDice == 8 && front == "fob watch...")
 - Watch = false;
 - defenseDieCount += 1;
 - Print that you got an extra defense die due to the watch
- Else if(diary == true && sumOfDice == 11 && front == "1200 year-old diary")
 - Diary = false;
 - LifeCounter += 1;
 - Print that you got an extra life due to diary.
- Return sumOfDice;

Player class Tests

Test Subject	Input/Expression	Expected Results	Observed Results	Comments
Constructor()	Player player;			
Constructor::do{while(creatureType == "")}	1. creatureType == "" 2. creatureType != ""	1. Will continue to loop 2. Will move on with the text	1. Continued to loop da loop (thankfully) 2. Moved on with text from file.	This was added in case the user missed where to type their name, it would reprompt them for it until they entered it.
Void setStrength(int inflicted)	player->setStrength(int)	Will take the int as an argument and will potentially alter the player's strength if certain criteria are met.	Did as expected (based on the following tests)	
setStrength():if(appliedDamage < 1)/else if(appliedDamage >= strength)/else	1. appliedDamage < 1 2. appliedDamage >= strength 3. appliedDamage > 0 & appliedDamage < strength	1. Print that the player dodged the damage. 2. Print it was a fatal hit. Decrement lifeCounter, set strength = 0, run through if/else if/else statements 3. It will subtract appliedDamage from strength.	1. Printed the thing. 2. Printed fatality and set strength = 0. 3. Re-assigned strength to the new value.	
setStrength():else	1. lifeCounter == 3	1. Print regeneration	1. He's wearing a	

<pre>if():if(lifeCounter == 3), else if(lifeCounter == 2), else if(lifeCounter == 2),else</pre>	<pre>2. lifeCounter == 2 3. lifeCounter == 1 4. lifeCounter == 0</pre>	<pre>speel, set outfit, reset strength to max, pause for a second and then print new outfit 2. Print regeneration speel, set outfit, reset strength to max, pause for a second and then print new outfit 3. Print regeneration speel, set outfit, reset strength to max, pause for a second and then print new outfit 4. Set strength = 0</pre>	<pre>leather jacket and t-shirt 2. He's wearing a brown trench coat and suit 3. He's wearing a red fez and a bowtie 4. I guess he's naked because he be deed.</pre>	
<pre>addItem(Compartment*c)</pre>	<pre>player->addItem(Compartment* c)</pre>	<pre>Will copy the item from a class and store it as a string. It will then print that it took that item (to test to make sure it actually copied the item and that it is the correct one) and will add it to the back of the TARDIS.</pre>	<pre>So much easier than creating our own. Worked like a charm.</pre>	
<pre>String removeItem()</pre>	<pre>player->removeItem()</pre>	<pre>If the TARDIS isn't empty, it will copy the front of the tardis to a string variable, pop it off and then add it to the back of the TARDIS</pre>	<pre>2 for 2. Worked as expected.</pre>	<pre>The reason I have it add the item back into the TARDIS is because I wanted the player to "get lucky" by having certain criteria met. If they were all removed and not added back in, the user would only get a max of 4 chances to use the items, which isn't what I wanted. And if I had them be removed on a certain number, then it would take away the</pre>

				randomness. This was just a design choice on my end.
Void setLocation(Compartment*)	player->setLocation(compartment)	Assigns location to 'you_are_here'	Assigned the location passed to it to "you_are_here"	
Void setOutfit()	player->setOutfit()	Changes the player's outfit based on the lifeCounter.	Did as expected (based on the next test's results)	
setOutfit():if(lifeCounter == 4), else if(lifeCounter == 3), else if(lifeCounter == 2), else if(lifeCounter == 1)	1. lifeCounter == 4 2. lifeCounter == 3 3. lifeCounter == 2 4. lifeCounter == 1 5. lifeCounter == 0	1. Sets outfit to silk scarf with wasticoat 2. Sets outfit to leather jacket with trenchcoat 3. Sets outfit to brown trench coat and suit 4. Sets outfit to red fez and a bowtie. 5. Does nothing	1. Yep. 2. Fantastic! He's my favorite. 3. Good, good 4. Stud for days. 5. Nada. Yep, they all worked like expected.	
Compartment* getLocation()	player->getLocation()	Returns the current location of the player	Did as anticipated.	
Void setCreatureType(string n)	player->setCreatureType(string)	Will name the player anything passed to it.	Named the player the players name. Renamed the player 'The Doctor'	
Void setAdvice()	player->setAdvice()	Print statement and set advice to true	Printed the statement and changed the bool to true	
Bool getAdvice()	player->getAdvice()	Return 'true' or 'false' based on advice.	Returned true when advice was true and false when it was false.	
Void setScrewdriver()	player->setScrewdriver()	Print statement and set screwdriver to true	Got the statement and it was set to true	
Bool getScrewdriver()	player->getScrewdriver()	Will return 'true' or 'false' based on the screwdriver.	Returned 'true' when screwdriver was true and 'false' when screwdriver was false.	

Void setWatch()	player->setWatch();	Print statement and set watch to 'true'	Printed the dialog and set watch to true	
Bool getWatch()	player->getWatch()	Return 'true' if watch is 'true' and 'false' if watch is 'false'	Returned true when true and false when false.	
Void setDiary()	player->setDiary()	Print statement and set diary to true	Printed the statement and set diary to 'true'	
Bool getDiary()	player->getDiary()	Return 'true' if diary is 'true' and 'false' if diary is 'false'	Returned as expected	
Int getLifeCounter()	player->getLifeCounter()	Return the current lifeCounter	Did the thing.	
Void setItems(string s, compartment*c)	player->setItems(s, c)	Will compare 's' to the preset strings to see if they match. If s matches one, it will execute that statement and add the item from the compartment that is passed to the TARDIS.	HUZZAH! Getting something to work that simplifies your code is an amazing feeling. This worked like a charm	
setItems():if(), else if(), else if(), else if()	1. S == "5,000,000,023" 2. S == "1890" 3. S == "2016.5" 4. S == "2016(ish)" 5. S != any of the above	1. Will call setAdvice() & will add item to tardis. 2. Will call setWatch() and add item to TARDIS 3. Will call setScrewdriver() and will add item to the TARDIS. 4. Will call addDiary() and will add item to TARDIS. 5. Will do none of the sort.	1. Printed the statement and added the item. 2. Printed the statement and added the item 3. Printed the statement and added the item 4. Printed the statement and added the item. 5. Basically skipped this function.	
Int attack()	player->attack()	Will "roll" a random number within a certain range and will return that number.	Did as expected.	

attack():if(current Room == "Parallel Universe")	1. currentRoom == "Parallel Universe" 2. currentRoom != "Parallel Universe"	1. Will allow the following if/else if statements to execute when appropriate. 2. Skip to end of function	1. Suprisingly worked for all of them (based on many runs of this function) 2. Ignored this portion of the function	I learned that you cannot cheat the random feature, so I just printed out everything and kept running it until I saw that all 4 statements worked.
attack():if():if(), else if(), else if(), else if()	1. Advice == true, player rolls a 3, & front == advice item 2. Screwdriver == true, player rolls an 6, and front == screwdriver item 3. Watch == true, player rolls an 8, front == watch item 4. Diary == true, player rolls an 11 & front == diary item 5. None of these previous conditions are met.	1. Will restore max strength and print that it happened. Sets advice to false to keep it from being reused. 2. Will add an attack die and will print that it happened. Sets screwdriver to false to keep it from being reused. 3. Will add a defense die and print that it got one 4. Will add a life and print that it got one 5. Acts as a normal roll	After lots, and lots of rolls. It works. For every case. And it really does add some excitement to the game.	I didn't want the player to get to use the items every time they rolled the certain die (because they were only carrying one of each item, anyways), so I set it to false to "switch it off". I wanted it to almost be like a last rally against the master. He may start by beating you but you can come up as the underdog in the end.

Main.cpp

- Int gameRun = 0;
- Do
 - Clear screen
 - Print title & description
 - Int seeKey;
 - Prompt user to enter '1' to see the key or '0' to move on.
 - Do
 - Cin >> seeKey;
 - while(seeKey < 0 || seeKey > 1)
 - if(seeKey == 1)
 - Print the key
 - Instantiate one of each compartment types.
 - Compartment *fut = new Future();
 - Compartment *parUn = new ParallelUniverse();
 - Compartment *past = new Past();
 - Compartment *pockUn = new PocketUniverse();
 - Compartment *space = new Space();
 - String line;

- Prompt user to enter 1 to skip to game else enter 0 to read text.
- Do
 - Cin >> gameRun;
- while(gameRun > 1 || gameRun < 0)
- cin.ignore()
- if(gameRun == 0)
 - Ifstream myfile("Library.txt")
 - Print "Press Enter to continue through the text"
 - Cin.ignore
 - if(myfile.is_open())
 - while(getline(myfile, line))
 - cin.get()
 - Print line
 - myfile.close()
 - Else
 - Print error message
- Instantiate player
 - Player *player = new Player()
- if(gameRun == 0)
 - Ifstream nextfile("Library_part2.txt")
 - if(nextfile.is_open())
 - while(getline(nextfile, line))
 - cin.get()
 - Print line
 - Else
 - Print error message
- String option = "1";
- fut->setCompass(parUn, past, pockUn, space)
- parUn->setCompass(past, pockUn, space, fut)
- past->setCompass(pockUn, space, fut, parUn)
- pockUn->setCompass(space, fut, parUn, past)
- space->setCompass(fut, parUn, past, pockUn)
- player->setLocation(space)
- Do
 - if((player->getLocation() != parUn) || (player->getDiary() == true) && (player->getWatch() == true) && (player->getAdvice() == true)))
 - Print current location
 - Print current year
 - Print current outfit
 - Option = ((player->getLocation())->challenge(player))
 - player->setItems(option, (player->getLocation()));
 - Else
 - Print that you can't enter parallel universe

- if (option != "0" && player->getLifeCounter() > 0)
 - player->setLocation(player->getLocation())->menu())
 - while(option != "0" && player->getLifeCounter() > 0)
 - if(player->getLifeCounter() == 0 && (player->getLocation() == parUn))
 - Print to select a number
 - Validate input
 - Else if(player->getLifeCounter() == 0)
 - Prompt user to either try again or end game
 - Validate input
 - Delete all the objects & pointers and set them all to NULL.
 - Delete player
 - Player = NULL
 - Delete fut
 - Fut = NULL
 - Delete parUn
 - parUn = NULL
 - Delete past
 - Past = NULL
 - Delete pockUn
 - pockUn = NULL
 - Delete space
 - Space = NULL
- while(gameRun == 3)
- Return 0;

mainTests				
Test Subject	Input/Expression	Expected Results	Observed Results	Comments
do{}while(gameRun == 3)	1. gameRun == 3 2. gameRun != 3	1. Will continue to loop 2. Will stop the loop after 1st iteration	1. Continued to loop it 2. Stopped looping and terminated program	
do{}while(seeKey < 0 seeKey > 1)	1. seeKey < 0 2. seeKey > 1 3. seeKey == 0 4. seeKey == 1	1. Loop and ask user to re-enter 2. Loop and ask user to re-enter 3. Will move on 4. Will move on	1. Prompted user to re-enter int 2. Prompted user to re-enter int 3. Moved on 4. Moved on	
if(seeKey == 1)	1. seeKey == 1 2. seeKey != 1	1. Will print the key 2. Will move on	1. Printed the key 2. Moved on	

do{}while(gameRun > 1 gameRun < 0)	1. gameRun < 0 2. gameRun > 1 3. gameRun == 0 4. gameRun == 1	1. Loop and ask user to re-enter 2. Loop and ask user to re-enter 3. Will move on 4. Will move on	1. Prompted user to re-enter int 2. Prompted user to re-enter int 3. Moved on 4. Moved on	
if(gameRun == 0)	1. gameRun == 0 2. gameRun != 0	1. Will print file 2. Will skip file	1. Printed file 2. Skipped file	
if(myfile.is_open())else	1. File is there to open 2. File can't be opened	1. File will be printed 2. Error message will be printed	1. File was printed 2. Error was printed	
while(getline(myfile, line))		Will read the entire text file, line by line.	Read it like it was supposed to. Woop.	
if(gameRun == 0)else	1. gameRun == 0 2. gameRun != 0	1. Will read the file 2. Will skip the file	1. Read the file 2. Skipped it	
if(nextfile.is_open())else	1. File is there to open 2. File can't be opened	1. File will be printed 2. Error message will be printed	1. File was printed 2. Error was printed	
while(getline(nextfile, line))		Will read the entire text file, line by line.	Read it like it was supposed to. Woop. Woop!	
do{}while(option != "0" && player->getLifeCounter() > 0)	1. Option != "0" & player's life count > 0 2. Option == "0" & player's life count > 0 3. Option != "0" & life count = 0.	1. Continues through loop 2. Ends the loop 3. Ends the loop	1. Continued through loop 2. Ended the loop 3. Ended the loop	
if(player->getLocation() != parUn (player->getDiameter() == true) && (player->getWatching() == true) && (player->getAdvice() == true)))else	1. Player is not in parUn 2. Player has all the items necessary 3. Player is not in parUn and doesn't have all necessary items	1. Goes through if statement 2. Goes through if statement 3. Skips if statement and goes into else (prints that you can't enter.)	1. Did as expected 2. Did as expected 3. Thankfully, did as expected.	

<pre>if(player->getLifeCounter() == 0 && (player->getLocation() == parUn))/else if(player->getLifeCounter() == 0)</pre>	1. Player's lifeCounter = 0 & master killed them 2. Player won	1. Will ask user for input and will validate it to see if they want to continue 2. This will be ignored.	After a couple minor changes, it works seamlessly	
------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------	---------------------------------------------------	--

Game Play

- The player wakes up on the floor of what appears to be a library, not knowing where they really are.
- A woman (River Song) approaches them and calls them 'The Doctor' but they are confused because that's not their name.
- She asks them what their name is
 - Have user enter their name;
- She explains that you have false memories making you think you are that person, but you are actually 'The Doctor' and that your archenemy, The Master, has erased your memory and sent you through space in time far, far away from him so that he can first take over the world (and then possibly the universe). Not only that, but he has stripped you of your Time Lord essence and scattered your belongings throughout time and space to keep you from foiling his plan. She also informs you that you are not in a library but on a planet called 'The Library' and that there are creatures lurking in the dark, waiting to devour you. According to her, you knew this day would come (and that you would be sent here on this time), so you hid your 1200 year old diary somewhere in the library here to help remind you of who you are and asked her to be there to help you once your memory was wiped.
- Your first challenge is to try and find the diary. You must find it before the Vashta Nerada get to you.
 - To find the diary, you must roll an 8 with your attack die.
 - You will have a maximum of 5 chances to roll the 8.
 - After each attempt, the Vashta Nerada will take their chance at attacking you.
 - If you do not find the diary, you will be given another chance (meaning more attacks.).
 - You will either be able to try the challenge again, die trying, or return to the TARDIS to try your hand at another challenge.
 - If you find the diary, your memory will be restored, identity changed, and you will be able to bring the diary back with you to the TARDIS to use later on (perhaps against The Master..)
- You will then have the choice of 4 other challenges:
 1. Visit the Future
 2. Visit the Past
 3. Visit a Pocket Dimension

4. Visit a Parallel Universe (this will not be selectable until you have acquired the majority of the items you search... but more on that later.)

If you go into the Future...

- You will end up in a gridlock underneath New New York in the year 5,000,000,023. To pass the time until you reach the surface, you get to play a “Doctor Who Trivia Game” (you’re welcome. This is to motivate you to get interested in Doctor Who [if you aren’t already].). But be warned, Cybermen are reportedly going from car-to-car and converting people to join their army.
 - There will be a total of 8 true or false questions you must answer to successfully complete this challenge.
 - Yes, you must get all 8 questions right to win.
 - As a tip, this round is based on the episode “Gridlock”, which is the 3rd episode of it’s season. The key to the questions is the binary code for ‘3’, 00110011, where 0 = “false” and 1 = “true”.
 - If you don’t answer a question correctly, you are attacked by a Cybermen. You must fight the Cyberman to the death (either yours or his).
 - If you survive the challenge but didn’t get all of the questions correctly, you have the option of trying it again or moving on to another challenge.
 - If you get all of the questions correct, you make it to the surface and you get to visit you friend, the Face of Boe, who is dying in the hospital. Before his time is up, he says his final goodbyes and gives you some advice. You get to take that advice back with you to the TARDIS to use later on.

If you go into the Past...

- You will end up in 1890 Provence with Vincent van Gogh who is the only person that can see this invisible (and blind) creature that has been attacking and killing people. He asks for your help to take of the beast. You agree, of course, and help him fight the Krafayis.
 - This is a dice rolling game. You and the Krafayis roll attack dice to see who can roll the highest number.
 - If you roll the highest, it means you think you know where the Krafayis is so you take the opportunity to attack it.
 - If the Krafayis rolls the highest number, it means he figures out where you are and attacks you.
 - If you guys roll the same number, you both get to roll again. (but as a safeguard to keep from seg faulting, the Krafayis will overpower you during a struggle and attack you. :))
 - This game is also a fight to the death (either yours or his.)
 - If you survive, Vincent van Gogh helps you defeat the Krafayis once and for all. He then bestows upon you a pocketwatch that he found while painting. It turns out to be your fob watch with a Chameleon Arch (aka the thing that can restore your Time Lord essence). Huzzah!
 - If you die, the game is over.

If you go into the Pocket Dimension/Universe...

- You end up in a Pocket Dimension full of Weeping Angels. Yikes! You find out that your Sonic Screwdriver is located in this Dimension, but it's being held for ransom.
 - To earn it back, you must roll a total of 70 points with your dice. But be warned, if you blink (roll an odd number), the Angels will attack.
 - If you survive but don't earn enough points, you can either try again or go somewhere else. Since Pocket Dimensions are known for being short lived, you will only get 3 chances to win it back, though.
 - If you don't win it after 3 tries, oh well! You must leave the Dimension or you will get stuck there... forever.

You can only cross over into the Parallel Universe when....

- You have collected: the advice, the diary, and the watch (the screwdriver is optional due to the limitations on attempts).
- In this Universe, The Master has taken over the world as Prime Minister. You must overthrow him to save the world/universe/time/space!
 - This round is also a fight to the death, but you both just take turns.
 - The player just gets to watch this fight take place.
 - The Master seems to be more powerful than you, and you watch as your strength dwindles away.
 - But the game isn't over!
 - Because you collected those items, you have the chance of making a gain on The Master!
 - As the fight ensues, you might have the chance to earn an extra life (thanks to the diary), restore your strength (thanks to the advice), gain an attack die (thanks to the screwdriver), or even receive an extra defense die (thanks to the watch).... But it all depends on your roll/what you have in the TARDIS.
- Only one of you will make it through the fight. I hope for everyone's mercy, it's you.

Reflections

I really enjoyed this challenge, although it was very daunting to begin. The test document came in really handy when I was trying to plan out my game. One of the biggest mistakes I would have missed without it was that I did not actually have a way of terminating the game if the player died within 4 of the 5 Compartments. Yikes!

I am very pleased with how it turned out, although I know there are some areas that still need work. I spent a lot of time and put a lot of effort into this game trying to make it something interesting for anyone (even people that did not know anything about Doctor Who).

Just as a recap if a non-whovian plays the game:

- The doctor regenerates into a new form when he "dies"
- River Song is his wife, whom is also a time travelers... and is also an assassin.
- Cybermen have no emotions, so they want to eradicate them.
- The Master is also a Time Lord (like The Doctor), but he was raised for evil.

- The Face of Boe is a giant face in a glass container (think Futurama)



-
- The TARDIS is The Doctor's time machine that lets him go anywhere in time and space.
- The sonic screwdriver is his tool that he uses to help him out of sticky situations (and yes, that is why Tommy Pickles carried a screwdriver if you watched the Rugrats).
- Each of the different outfits represents a previous doctor.
 - You start out as 9 (because he is my favorite), but depending on your luck you can be anyone from 8, 9, 10, or 11.
- Weeping Angels are these statue-like aliens that freeze whenever you look at them. Once you turn your back or blink, they come after you. If they touch you, they send you into the past to live out your life before you were even born.
- Krafayis is blind (although their species usually aren't) and is abandoned on earth because of it. He looks like a dinosaur/bird like creature.
- Vashta Nerada sneak into space suits and kill you while letting your energy live in the suit.. Slowly draining itself. Saddest part, no one knows you are dead until after it's too late.

I have learned a lot throughout this course, and continued to even with this assignment. I feel like I can accomplish so much now because of this class. It has been a wonderful (yet stressful) experience.