Jonathan Spencer

Derek Marks

Corey Collins

Alexandru Trefi

OOD Final

10 May 2018

**Game Report**

Welcome aboard captain! The Helios 7 is yours to command...or so you thought. The premise of the game starts as a character of your choosing wakes up on the control bridge and has to make his way through several obstacles to make it to the escape pod. This game is designed to be played several times and cause some minor frustration...not from bad design but from the level of difficulty we coded the game with.

There were many difficulties faced when designing a game or any large project for that matter the first one that we as a group had to code in a similar way or learn how the others coded so we could adapt. Another large obstacle was delegating tasks. Not all members of every team have the same skill sets or the same amount of time outside of class or even the same level of dedication to the class and project.

We decided to use two different design patterns the first being the decorator pattern. The decorator pattern allows us to change how the objects work or what objects change them. In our game we took our characters and gave them a decorator pattern this allowed us to add items to an inventory as well as change things like health and status effects. The second design pattern that we implemented was the memento design pattern. This pattern is great it allows us to save objects in the state that they are currently in. A good example of this is that we save the objects at the starting point of each room so that when the player dies and we give an option to start the room over it goes back to the state at the beginning of the room allowing the player to continue rather than starting the game over after each death.

The program usage is very simple its a game. You play games to get away from the real world or to challenge your brain. With just a few changes to the code it could be a learning game or a game that is so challenging that it may be close to unbeatable.

**Pseudo Code**

**CharacterInterface**

Create the following variables

|  |  |  |
| --- | --- | --- |
| public int getOxygen(); | public void setHp2(int hp2); | public void loseStrength(); |
| public void setOxygen(); | public void getInventory(); | public int getStrength(); |
| public int getHp(); | publicArrayList<> getList(); | public int getCharisma(); |
| public int getHp2(); | public String getName(); | public int getIntellect(); |
| public void setHp(int hp); | public void addStrength(); | publicArrayListgetCombat() |

**Character**

Uses all variables from CharacterInterface and override them to be usable by the character class

**CharacterDecorator**

Implements CharacterInterface

Add and override all variables from Character

**The Capitan, Doctor, and Soldier Classes**

Uses characterInterface

Add variables

|  |  |  |
| --- | --- | --- |
| Int Strength | String Line | Arraylist Inventory |
| Int Charisma | String Name |  |
| Int Intellect | Arraylist combat |  |

Override variables all except HP to allow for use in individual class.

Add toString method and format in a way to describe the respective characters.

**ItemInterface**

Add Variables

|  |  |
| --- | --- |
| String getName | String getDetails |
| int getStrength | String getType |

**Letter, Knife, Suit, Pipe, Blue, Red and Green Syringes**

Implements itemInterface

Make constructor

Use and override variables in ItemInterface

Add Variables

|  |  |
| --- | --- |
| String getName | String getDetails |
| int getStrength | String getType |

**Main**

Add objects and variables

Add character ArrayList

Create method to list the characters that describes the characters

Add story code

Allow user to choose hero

Create hero of type character

Randomly choose villain from ArrayList - user’s choice

Call the rooms to start the game

**Caretaker**

Add ArrayList of type memento

Add addMemento method to add save state information to end of ArrayList

Add a getMemento method that gets a specified save state.

**Editor**

Add the hero CharacterInterface

Add setHero method to reference this hero

Add save method which returns a new hero of type memento

Add restore method that that retrieves memento restore state

**Memento**

Add CharacterInterface

Set this character equal to character

Add getCharacter with return of character

**Room One**

Add Story code

Add choices to access terminal, search the room, and go to safe space

Add choices to check ship status and turn off alarm

Add chance to add suit and letter to inventory

Leave to next room

**Room Two**

Add room introduction

Add choices to search the room, put out fire, and deal with gas

Add pick up knife and eat food options

Add put out fire options

Add death menu

Add option to go to next room

**Room Three**

Add story code

Add choices to suicide, fix the hole, or suicide

Add choice to search the room, bench press, or leave the room

Add word scramble to get to next room or take damage

**Room Four**

Add story text

Add option to search the room, take a nap, or leave the room

Add syringes

Add option to move on to medical supplies

Add option to heal up or leave

Add word puzzle to determine how much you heal for

Force to next room

**Room Five**

Add story code

Add choice to fight or give up

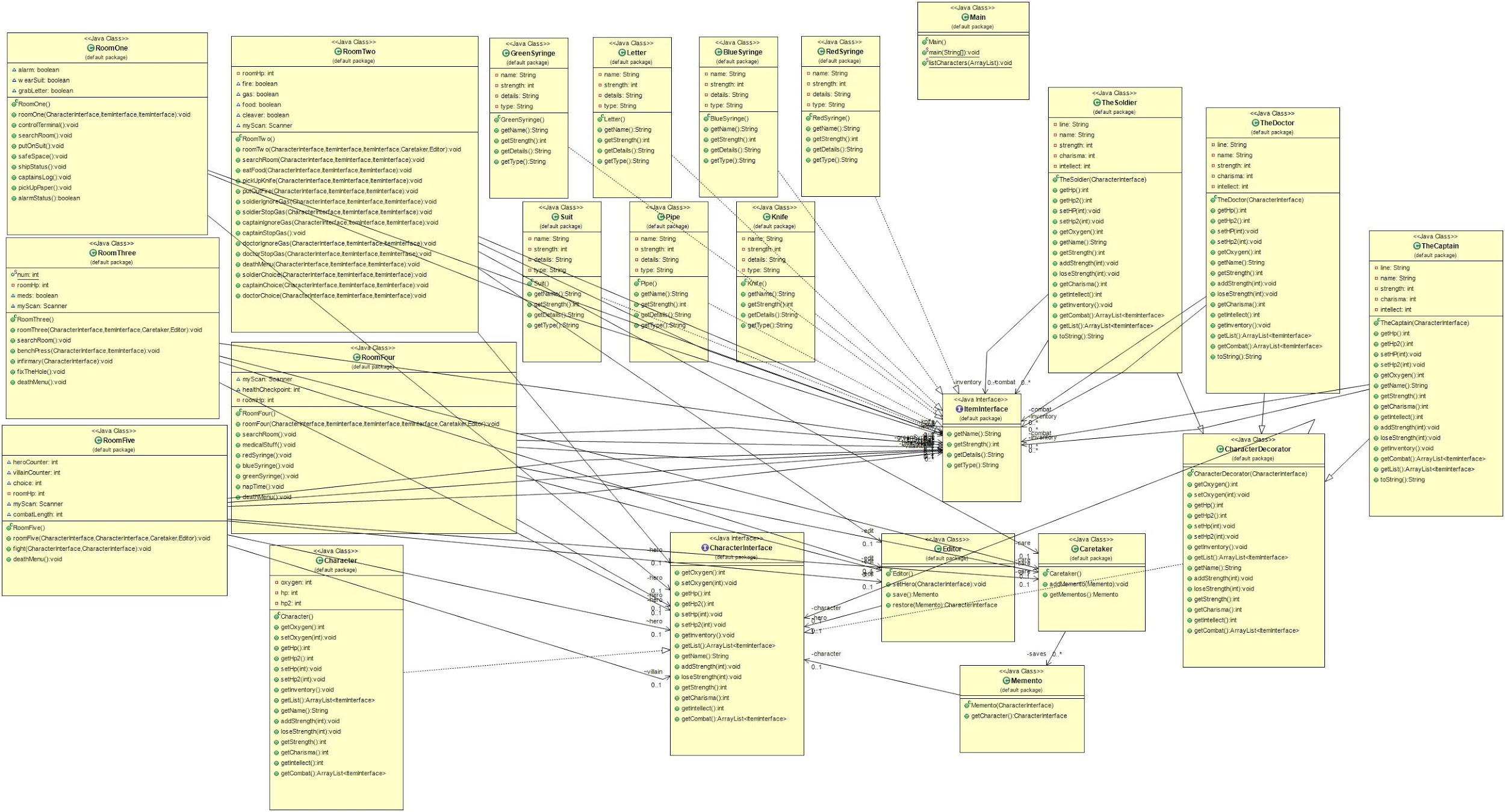
Add combat

Add end game scenario

Add choices to use access code, smash the terminal, or go down with the ship

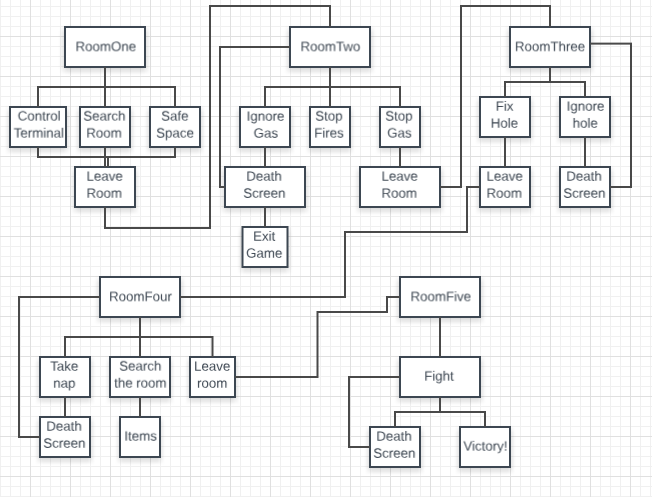
**UML Diagrams**

*Class Diagram*



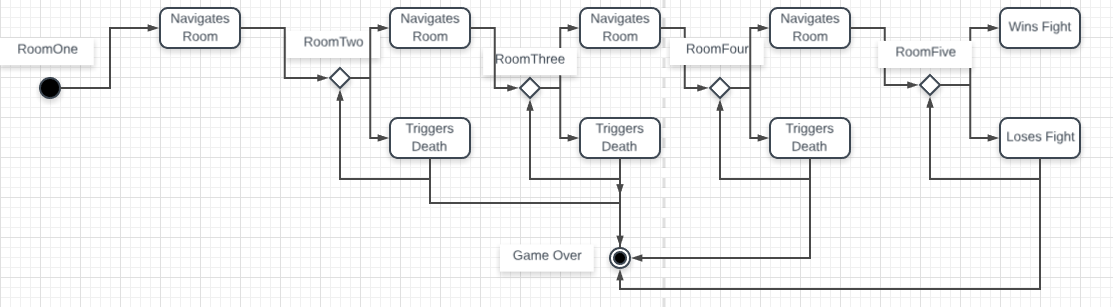
Our class UML diagram is hefty to say the least it goes through each class and all of its parameters allowing for easy transition to code. An interesting thing that we found while designing this is that you can take a well written UML and there are java translators that will change your UML to java code. (See attached for easier to read version)

*Object Diagram*



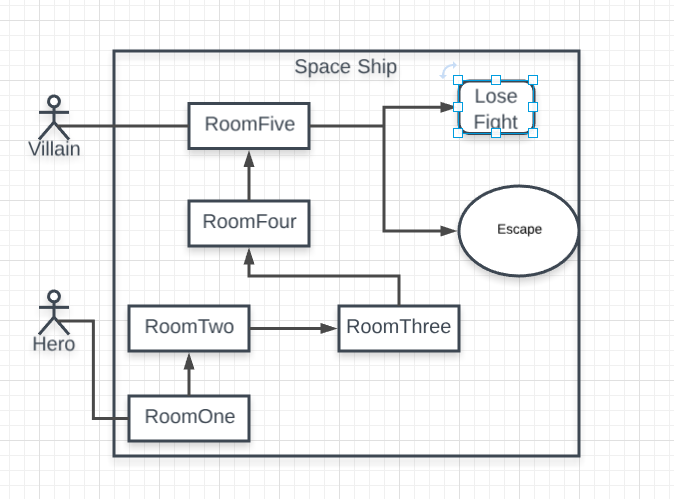
Our object UML is a simple layout of all five rooms and the choices which gives you an idea of all the different times that the player can die and that there are very specific choices that need to be made in order to succeed and get to the end of the game. This also allowed us to see what order in which things needed to happen so that we could delegate some of the work around so that it didn’t take so much time.

*State Diagram*



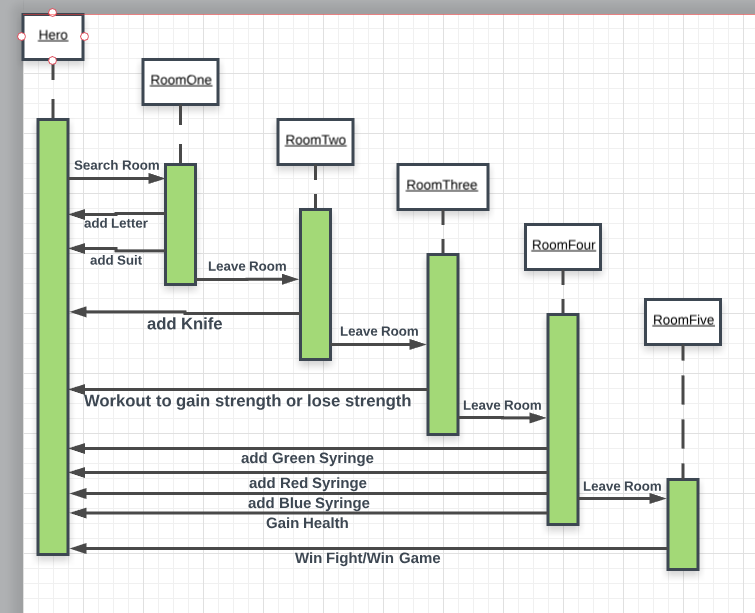
Our state chart goes through each room with the intent to show the state of the player as they progress through the story the two states are alive and advancing or dead.

*Use-Case Diagram*



The use-case diagram shows the two characters in the game and goes through the game with them showing that their interaction takes place in the last room. The hero needs to travel through four rooms until he reaches the villain.

*Sequence Diagram*



The sequence UML goes through the order in which the events happen and to which objects. In the full implementation of the game there are a couple of other events that happen like taking damage in some of the rooms as well as the ability to die in most of the rooms.