6/5/2016

CS12320 MainAssignment (bonks and zaps)

GLYN ROUX – gjr4@aber.ac.uk

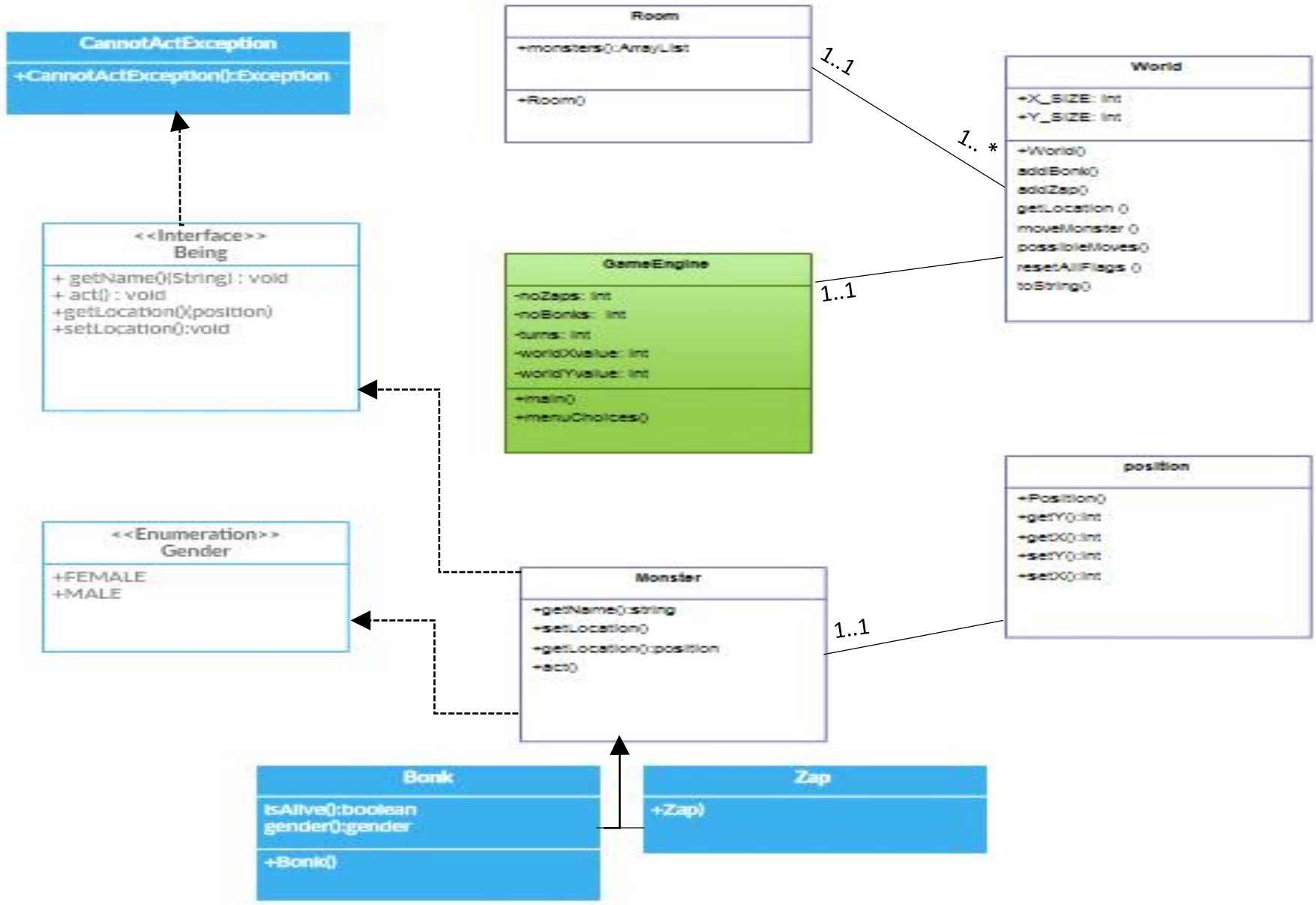
Contents

Introduction

These following pages detail my experiences of designing and writing the code to the specification of this project. I start by looking at the design of the project, my first steps when thinking about the project and how initially, I thought to solve the problem that was given to me. I also include a class diagram that I used to give me a basic idea of how I would structure my program I include a textual description of each class. some pseudocode that I drafted whilst planning the project will be included as well as a sequence diagram for the more complex interactions. a testing section will be given that shows screenshots of the code that I have written and outputs from the code in the terminal. I will evaluate my experience of the project and sum up how I found it and what I'd do differently in the future If I got the chance to do it again.

Design

UML Class Diagram



Description of Classes

Being:

Using this interface meant having to implement all the methods included in the interface to that which implements it. I used this interface in the Monster class so had to include the methods ‘getName’, ‘getLocation’ and ‘setLocation’. I did this so that I could use these methods to get a specific location of a bonk or a zap. I also used the ‘setLocation’ method in order to randomly set the location of each ‘monster’ in the world.

GameEngine:

This class is where I put all the code concerned with creating instances of each object needed. I used it to add new bonks and new zaps to the world as well as initialising an instance of the world itself. Also as part of an extension I also allow the user to choose if they want to use the default settings for the game, 20 bonks 5 zaps and a world of 20 by 20, or if they wanted to create their own version with their own numbers.

Gender:

An enum used so that I could set the gender of each bonk and I used it in the monster class. I used this instead of a Boolean value because I felt this it integrated better with my solution.

Monster:

A superclass with the two subclasses ‘Bonk’ and ‘Zap’. Used for setting the location of and getting the location of ‘monsters’. I used two flags that are true when a bonk or zap has acted each turn so they cannot act more than once. The Bonk class extends this superclass and is used for Bonk type monsters. I used a Boolean that dictates whether the bonk is dead or alive, it is set to ‘B’ for every bonk as well as setting a gender for each bonk that is randomly generated. The zap class sets the identifier to Z for every zap.

Position:

Defines a new position, each position is to be structure with an x and a y coordinate. Each room would have a position.

World:

Responsible for the creation of the world array as well as creating bonks and zaps that populate it. The code that moves each monster every turn Is written here.

Room

This class creates the rooms of the world each room as an ArrayList populated with bonks and zaps.

Pseudocode examples

addMonster

```
Create bonk or zap
Add to Random position
```

Bonk

```
Bonk isAlive
Create unique id for bonk
Every time bonk made count+=1
 Create identifier from number and 'B'
      if (randon number greater than 0.5)
         gender = male
      else
         gender = female
```

getLocation

```
For (whole array)
            for each (store all monsters turn by turn)
                        if (monsters name equals identifier we are looking for)
                        return  position
                        else return null
```

moveMonster

```
Current location = monsters current location
      For each (store every monster in the current location in turn)
         if (monsters name = what we are looking for)
            delete monster in current location then use setLocation function to re-set it
            break out of loop
```

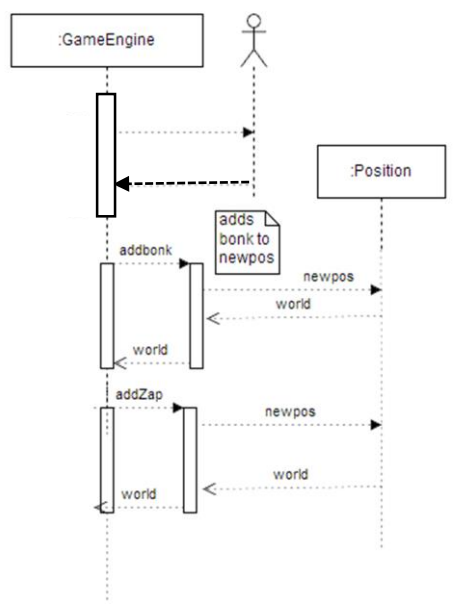
act

```
if (being zap and not acted)
            kill every bonk in same room as current zap
            set every bonk to dead using Boolean isAlive
if (being zap and not acted)
            if(bonk  alive)
                        mate with all opposite genders room
                        offspring +=1
if (being bonk or zap)
move using possible moves method
```

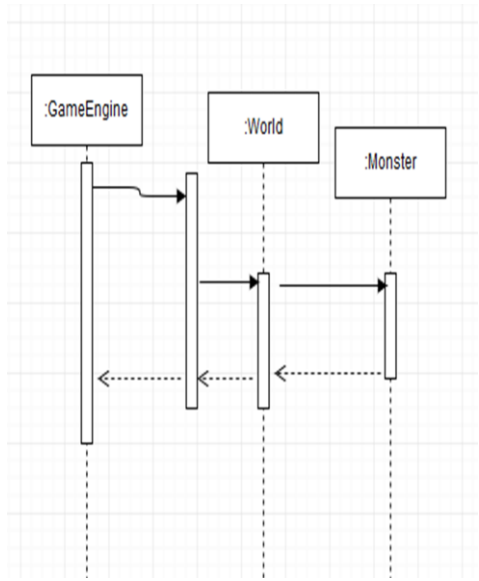
possible moves

```
for (the num. of the row behind the current position to the number of the row
in front)
            for (the num. of the column behind the current position to the
number of the column in front)
            if(the row is not = to row 0) {
               int newX = position of new x pos
               int newY = position of new y pos
               if (newX is in the array and newY is in the array
the new move is allowed
```

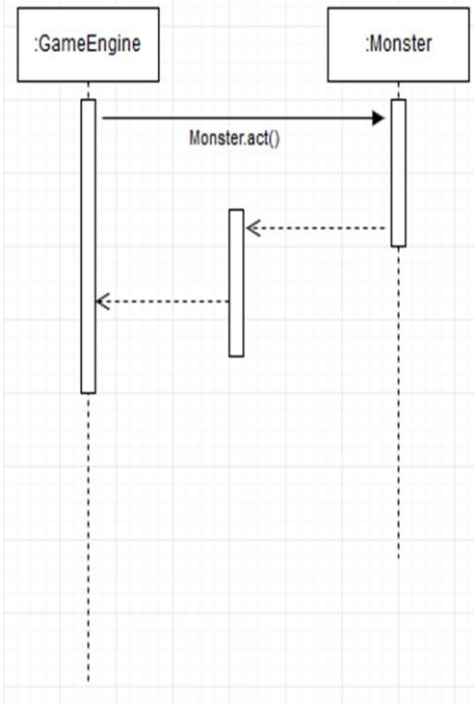
UML Sequence Diagrams



Getting user input, Adding a bonk and zap to a random position in the world



Getting the location of a monster in the world



Monster.act(), this is what happened when the monsters are told to act

Testing

Screenshots of running program and description of each screenshot

Description	Screenshot
Option is given to run the default setting, 20 bonks and 5 zaps in a world of size (20,20) with 5 turns, or define their own settings I do this with a case statement.	GameEngine "C:\Program Files (x86)\Java\jdk1.8.0_77\bin\java" ... Do you want to use default game settings(enter yes or no): npilation completed successfully in 28s 315ms (moments ago)
Series of printlines show bonks and zaps created and what rooms they are put into, after doing this the initial state of the world is shown.	GridWorld has begun Adding bonk to room[19][17] Adding bonk to room[8][0] Adding bonk to room[12][19] Adding bonk to room[8][19] Adding bonk to room[7][18] Adding bonk to room[0][1] Adding bonk to room[3][15] Adding bonk to room[13][7] Adding bonk to room[14][8] Adding bonk to room[3][11] Adding bonk to room[5][1] Adding bonk to room[14][11] Adding bonk to room[17][2] Adding bonk to room[18][18] Adding bonk to room[7][10] Adding bonk to room[11][3] Adding bonk to room[13][0] Adding bonk to room[6][5] Adding bonk to room[5][14] Adding bonk to room[6][1] Adding zap to room[16][14] Adding zap to room[1][13] Adding zap to room[9][16] Adding zap to room[6][19] Adding zap to room[17][8] GameEngine [0,1] B5MA [0,2] [0,3] [0,4] [0,5] [0,6] [0,7] [0,8] [0,9] [0,10] [0,11] [0,12] [0,13] [0,14] [0,15] [0,16] [0,17] [0,18] [0,19] [1,0] [1,1] [1,2] [1,3] [1,4] [1,5] [1,6] [1,7] [1,8] [1,9] [1,10] [1,11] [1,12] [1,13] Z1 [1,14] [1,15] [1,16] [1,17] [1,18] iles are up-to-date (3 minutes ago)

When all beings have acted printlines that show where each bonk has moved to, each bonk that has died and details if any baby bonks have been born.	Turn 2 B4 moving from (0,17) to (0,18) B11 moving from (1,0) to (1,1) B5 has been killed Z0 moving from (1,11) to (2,10) B7 moving from (1,13) to (1,14) B13 moving from (2,3) to (1,3) B14 moving from (3,17) to (2,16) B16 moving from (4,5) to (3,5) B0 has been killed Z1 moving from (4,8) to (5,9) B19 moving from (4,16) to (5,16) Z4 moving from (5,19) to (5,18) B3 moving from (8,7) to (7,8) B8 moving from (8,17) to (9,16) B9 moving from (9,8) to (8,7) B2 moving from (10,2) to (11,2) Z3 moving from (10,12) to (11,12) B18 moving from (10,16) to (9,15) B15 moving from (12,1) to (11,0) B1 moving from (12,7) to (11,6) B12 moving from (12,19) to (12,18) B10 moving from (14,15) to (15,15) Z2 moving from (15,7) to (16,7) B6 moving from (18,8) to (17,7) es are up-to-date (3 minutes ago)

Specific program testing

Requirement	Description	Inputs	Expected outputs	Pass/fail
Baby bonks cannot reproduce	When a baby bonk is made it cannot reproduce that same turn	The creation of a baby bonk	The baby bonk is created see screenshot 6 cannot reproduce	Pass see screenshot 10
Bonks and zaps cannot move if room does not exist	The code doesn't allow monsters to move to a position that it not adjacent to them or off the grid	The monster to be moved	The monster is moved to another point in the grid	Pass screenshot 9 shows the code used
Dead bonks are not allowed to move	Once a bonk has died it should not be given the	A dead bonk	That dead bonk stays in the same place from turn to turn	Pass see screen shot 8

	option to move			
If zaps are in the same room as bonks all the bonks die	Zaps must kill all bonks in the same room as them	A zap and one or more bonks	The program will print the bonks that died	Pass see screenshot 7
Bonks must breed with bonks of opposite gender	when two bonks of different gender are in the same room they produce a baby is then born	Two bonks of opposite gender	A print line that shows baby bonk	Pass see screen Shot 6
At the end of every turn bonks and zaps must move to a room adjacent to the room they are	The bonks and zaps are told to move at the end of every turn	The bonks themselves are put into a position to start	At the end of the turn they will move if they are alive	Pass see screen shot 4 and 5
My program must create zaps correctly and populate them	Zaps created with the identifier that includes Z and a number	written code that will handle the creation of zaps	Zaps created identifier and placed on the grid	Pass see screen shot 3

My program must create bonks and populate them	Zaps created with the identifier that includes B, a number, their condition and their gender	written code that will handle the creation of bonks	Bonks created with identifier and placed on the grid	Pass see Screen shot 3

program must handle input of int values as well as handling non int values	will check that the program takes an int value that is entered for any of the game settings that ask for user input as well as not accepting non int input	Normal: X size:4 bonks: 5 Zaps: 1 Turns: 5	inputs taken and the program runs with specified inputs	Pass
		Erroneous: Input letters instead of numbers	program will not allow the input will continue to ask for a number	Pass see screenshot 1

My program must handle input ‘yes’ or ‘no’ and erroneous inputs	will check the output when a user enters their choice for game	Normal: yes Normal: no	Normal: if yes then game default is run if no custom game is run	Pass
		Erroneous: 1 Erroneous: e	The inputs will not be accepted	Pass see screenshot 2

Screenshot1

```
"C:\Program Files (x86)\Java\jdk1.8.0_77\bin\java" ...
Do you want to use default game settings(enter yes or
no
Enter x value for world:
a
Please enter a number:
sdf
Please enter a number:
&
Please enter a number:
```

Screenshot2

```
"C:\Program Files (x86)\Java\jdk1.8.0_77\bin\java" ...
Do you want to use default game settings(enter yes or no):
o
Do you want to use default game settings(enter yes or no):
4
Do you want to use default game settings(enter yes or no):
ghj
Do you want to use default game settings(enter yes or no):
```

Screenshot3

```
 GridWorld has begun
Adding bonk to room[1][0]
Adding bonk to room[1][1]
Adding bonk to room[0][1]
Adding bonk to room[1][1]
Adding bonk to room[1][0]
Adding zap to room[0][1]
[0,0]
[0,1] B2MA Z0
[1,0] B0MA B4MA
[1,1] B1MA B3FA
```

Screenshot4

```
3
 GridWorld has begun
Adding bonk to room[1][1]
Adding bonk to room[0][2]
Adding bonk to room[0][2]
Adding bonk to room[2][2]
Adding bonk to room[2][1]
Adding zap to room[1][0]
[0,0]
[0,1]
[0,2] B1FA B2FA
[1,0] Z0
[1,1] B0FA
[1,2]
[2,0]
[2,1] B4MA
[2,2] B3FA
```

Screenshot5

```
Turn 1
B1 moving from (0,2) to (0,1)
B2 moving from (0,2) to (1,1)
Z0 moving from (1,0) to (0,1)
B0 moving from (1,1) to (1,0)
B4 moving from (2,1) to (1,2)
B3 moving from (2,2) to (1,1)
[0,0]
[0,1] B1FA Z0
[0,2]
[1,0] B0FA
[1,1] B2FA B3FA
[1,2] B4MA
[2,0]
[2,1]
[2,2]
```

Screenshot6

```
Baby bonk in room (18,8)

[18,8] B0FA B5MA
```

Screenshot7

```
[0,0]
[0,1] B4MA
[0,2] B1MA
[1,0]
[1,1]
[1,2]
[2,0] Z1
[2,1] B0FA
[2,2] B2MA B3FA Z0

Turn 1
B4 moving from (0,1) to (0,0)
B1 moving from (0,2) to (0,1)
Z1 moving from (2,0) to (1,1)
B0 moving from (2,1) to (1,0)
B2 has been killed
B3 has been killed
Z0 moving from (2,2) to (1,2)
[0,0] B4MA
[0,1] B1MA
[0,2]
[1,0] B0FA
[1,1] Z1
[1,2] Z0
[2,0]
[2,1]
[2,2] B2MD B3FD
```

Screenshot8

```
[0,0] B3FA
[0,1] Z1
[0,2]
[0,3]
[1,0]
[1,1]
[1,2]
[1,3] Z0 B1MA
[2,0]
[2,1]
[2,2]
[2,3]
[3,0]
[3,1] B2FA
[3,2] B0MA B5FA
[3,3] B4FA

Turn 3
B3 moving from (0,0) to (1,1)
Z1 moving from (0,1) to (1,2)
B1 has been killed
Z0 moving from (1,3) to (0,3)
B2 moving from (3,1) to (3,2)
B0 moving from (3,2) to (2,1)
B5 moving from (3,2) to (3,1)
B4 moving from (3,3) to (2,2)
[0,0]
[0,1]
[0,2]
[0,3] Z0
[1,0]
[1,1] B3FA
[1,2] Z1
[1,3] B1MD
[2,0]
[2,1] B0MA
[2,2] B4FA
[2,3]
```

Screenshot9	Screenshot10

```java
// this method determines all possible rooms that can be moved
// as an arraylist
ArrayList<Position> possibleMoves(Position position) {
    ArrayList<Position> moves = new ArrayList<>();
    for (int i=-1; i<2; i++)
        for (int j=-1; j<2; j++)
            if (!(i==0&&j==0)) {
                int newX = position.getX() + i;
                int newY = position.getY() + j;
                // if the room is still in the world i.e. the c
                if (newX >= 0 && newX < X_SIZE && newY >= 0 &&
                    moves.add(new Position(newX, newY));
            }
    return moves;

}
```

```java
if (getName().charAt(0)=='B'&&((Bonk)this).isAlive&&!acted) {
    Gender gender = ((Bonk)this).gender;
    int offspring=0;
    // go through all the monsters
    for (Monster monster : GameEngine.world.rooms[getLocation
        // if a bonk and alive (dont reproduce with a dead bo
        if ((monster.getName().charAt(0)=='B')&&((Bonk)monste
            Gender otherGender = ((Bonk)monster).gender;
            // if a female we reproduce with all males. this
            if ((gender== Gender.FEMALE&&otherGender== Gender
                // we just keep a count of number of offsprin
                offspring++;
        }
```

Evaluation

I drafted and redraft solutions as well as debugging my program in order to try and implement a solution. I found the project enjoyable and a great opportunity to exercise my knowledge using java.

have created an array as the basis for grid world as well as allowing user interaction, the users can pick the size of the world and the amount of bonks and zaps that populate the world and how many turns to run the program for. I also included a default game mode. I found the step of implementing this enjoyable.

The second problem was every array position in the world needs to be indexed with a row and column integer, this would form each room. Each of the positions in the array for the world was a room I did this by making an arrayList so each position in the world was an arrayList. Each room could have an unlimited amount of bonks and zaps. Each room is given an x and a y coordinate.

The next task concerned the visualisation of each array list as a room, each room would have a door to rooms adjacent to it. When moving any monster can travel to any room that is connected to it. When designing a method that dealt with this movement I had to allow the bonks and zaps to move to a position in the array, this part of the code proved difficult.

I then had to create the game engine, this contain the main method my code and the one that was tasked with running the program each time. Along with this I had to write methods in the superclass monster for both the subclasses zap and bonk. Due to the interface being I had to have specific methods within it. I had deal with bonks mating with only other bonks of a different gender to them, this was done by focusing on the male bonk mating with all females in the same room as it. This was so that I didn't have the same bonks mating twice, this part of the program was the most in depth because I had to allow for many different cases where the rules specified would have been broken. I had to include code that allowed the zaps to kill all bonks in the same room as them and for this I had to create a copy of the room I was working on so I could iterate through it and edit it at the same time

Overall the project was very time consuming, there are still elements of the projects that I feel I could add or improve, such as a GUI for the actual world. In this assignment I learnt that you can define an exception of your own which is something I didn't know already, I learnt how to use interfaces as well. I feel that on this assignment due to the fact I have a working project and have implemented a menu with user interaction I should receive a grade of at least 70%.