

# MONK

Adam Birch u1761249

CIS2343-1819  
Assignment

**MONK – Specification ver. 2**

## Contents

Specification .....	1
Structure and Behaviour .....	4
Use of the Object-Oriented design pattern .....	7
Interface design and UI components .....	8
Implementation Notes:.....	10
To run the game.....	11
Testing .....	11
Failed Tests .....	18
Evaluation .....	19

## Specification

MONK is a procedurally generated dungeon crawler where the monk traverses the dungeons in search for treasures to support his monastery which is on the edge of financial collapse. But the monk must be wary, for they are not the only things walking within these walls. Many monks have come to the dungeons for the same purpose and met grisly ends – only the strongest will make it out alive.

The game has several requirements to be release worthy for the official release – week 22.

These basic requirements include:

**A dungeon generation system.**

- There are three types of room: Empty, Monster, and Treasure.
  - o Room is a class of its own, with bool values to deduce which room is which.
  - o This was intended to use inheritance, however storing rooms in a vector with different types was problematic.
- The rooms must be connected to form a dungeon.
  - o Rooms are stored in a 1D vector.
  - o  $\text{Index} = (y * \text{sizeX}) + x$  where sizeX is the number of elements in a row.
- There can only be one treasure room.
  - o The room is explicitly defined at the start of the generation. Generation is a recursive method that runs after this is defined.
- A door must pair with another in the next room.
  - o A room will check around it for doors leading to it before generating new doors.
  - o The next room is generated with a reference to its origin and always puts a door in the location.

- The rooms must fit within a 4X4 6X4 or 6X6 grid based on desired size.
  - o The rooms are stored in a 1D vector whose size changes to be (x \* y) in length based on the desired size.
  - o Rooms are initially created as NULL rooms (invalid rooms) to populate the vector.
  - o The Vector.at() method is then used to get and set values in the vector.
- The Monk must enter a room with no monster.
  - o Change the state of a monster room to empty if the monk spawns there.
  - o This can never be the treasure room as the search for a spawn starts there and searches each door until it can't go further.
- The treasure room should not have a monster within.
  - o Spawn monster method can only be called in a monster room. Explicit definition of the Treasure room doesn't allow for a monster.

### **An exploration system.**

- There can only be one monk.
  - o The monk class is a singleton.
  - o The monk is a subclass of BaseCharacter (as is Monster), which is abstract.
- The Monk must be able to get to each room.
  - o Monk has a current room index.
  - o This changes based on the index the monk is currently in
- The Monk can only leave a monster room once the monster is slain.
  - o Combat starts on entering a monster room and cannot be left until a character dies.
- The Monk must exit the dungeon from the treasure room.
  - o The win method can only be called in the treasure room.
  - o The only other official end is if the monk dies.
- The Monk must be able to use items within empty rooms.
  - o The monk can pray to regain health in any room that was defined as Empty.

### **A combat system.**

- The fight should begin upon entering a monster room.
  - o If current room = monster then fight().
- Whomever has the higher speed attacks first.
  - o If monk.speed > monster.speed...
- Actions have a 50% hit rate.
  - o Using a random number generator – 0 miss, 1 hit
- Actions include Attack and Defend.
  - o 2 possible actions which call a different method.
  - o Monster's action based on a random number, but never defends when at full health.
- A successful attack will subtract the attacker's attack level from the target's health.
  - o Implement hit method (Health – othercharacter.getAttack())
- A successful defend will add 1 to the defender's health.
  - o Implement defend method. (Health++)
- If a character's health is or is below 0, they die.
  - o If health <= 0 this.destroy

- If them Monk dies, the game is over.
  - If monk.getHealth <= 0 GAME OVER
  - This is done by breaking the loop that allows the game to continue playing.

The initial game is text-based. In the future it is possible for the game to be overhauled to be a top-down based game (either 8-bit style 2D, or a modern 3D with lighting and shadows.)

I chose to develop the game within the CLion IDE as I am more familiar with the services provided by JetBrains products and find that some of the features are more advanced than those within Visual Studio, however the use of CMake within the project does make it more difficult to export – however a Monk.exe has been produced to run within a command line interface.

## Structure and Behaviour

Included within this section is a set of diagrams outlining the functionality of several sections of the game.

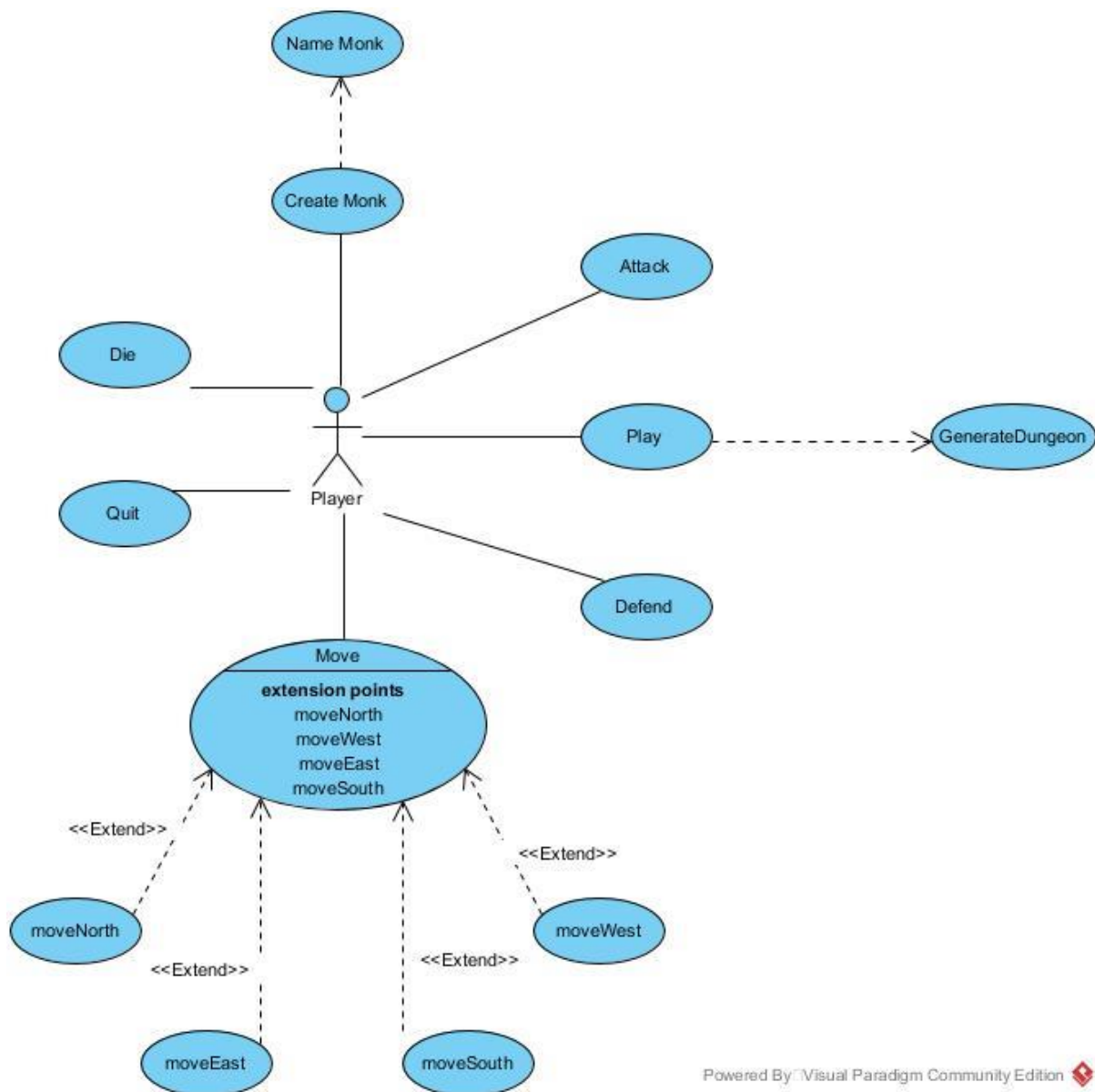


Figure 1

Figure 1 shows the user's basic interaction within the game. The player must be able to create a monk with a name and create a dungeon using the `GenerateDungeon` method. It must also be possible to win the game and lose.

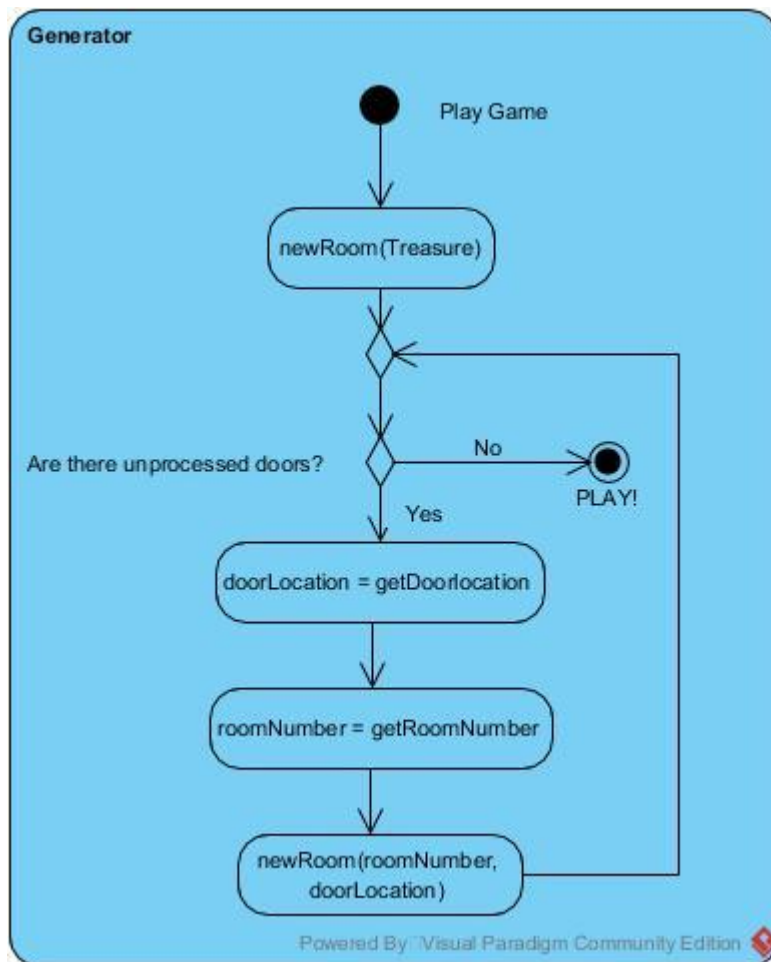


Figure 2

Figure 2 shows the first iteration of the GenerateDungeon method. The first room to be generated is the Treasure room and a cascading recursive function runs until either the dungeon is full or there are no more doors that lead to nowhere. This initial concept is one of the few things that stayed very true to the initial concept as it was detailed enough to implement but flexible enough to be able to implement in a variety of ways when one attempt failed or wasn't appropriate for the rest of the program.

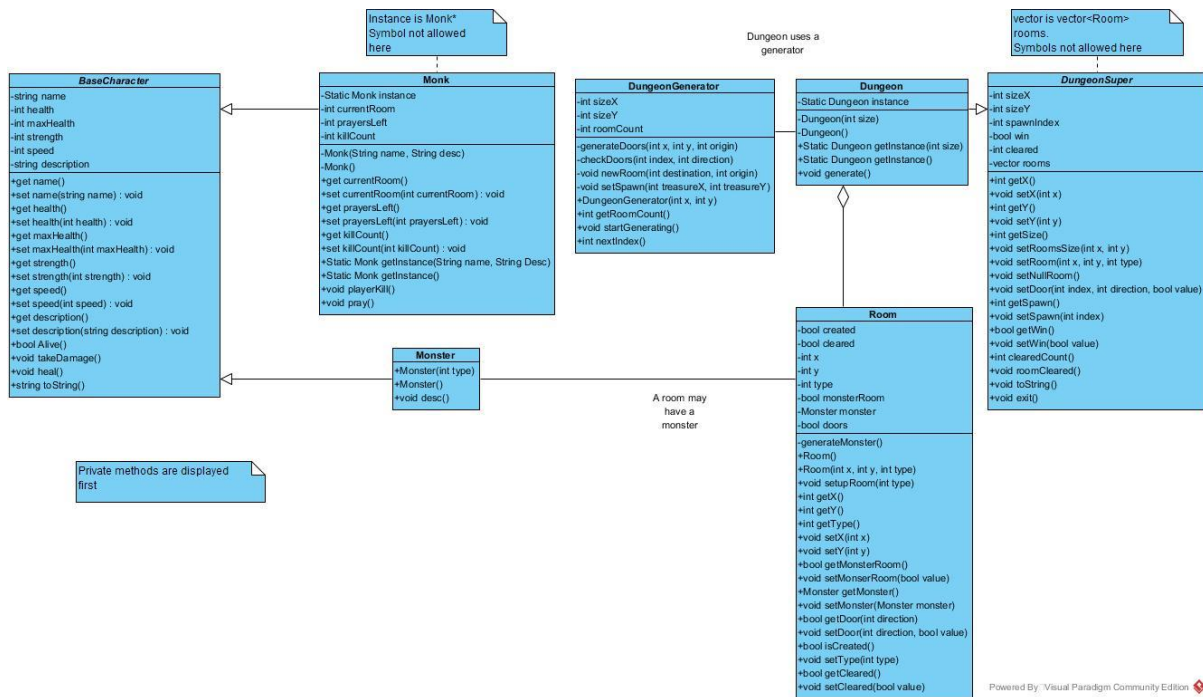


Figure 3

Figure 3 is an overall class diagram showing how the Character classes and Dungeon classes interact with each other – discounting the usage of classes with play functionality. BaseCharacter and DungeonSuper are abstract classes used to define the functionality of subclasses. This was done for BaseCharacter as it is shared between Monk and Monster but was done for DungeonSuper as it was the only way I was able to get the Dungeon singleton to work correctly.

Monster is the only enemy type currently but can be extended to create different sub types of monsters which implement other types of methods which may include using items or special abilities.

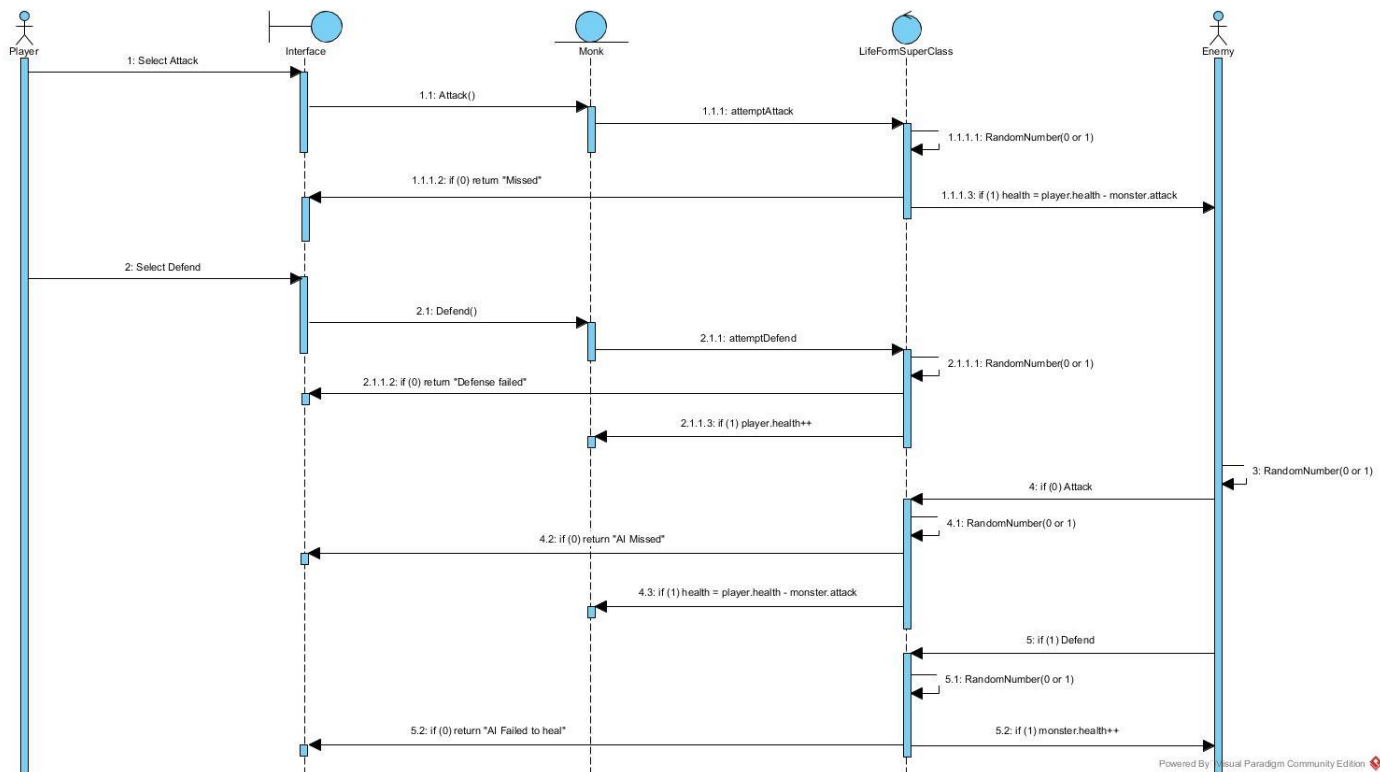


Figure 4

Figure 4 shows the initial concept for the combat mechanics. This doesn't check speed, so the player always moves first. The player can attack and defend as can the AI. Both characters have a 50% chance of success. The AI randomly selects a move, but this could be changed in the future to use logic based on each character's health. In the implementation there is a check before this that the character with the higher speed goes first.

## Use of the Object-Oriented design pattern

There are many examples of good Object-Oriented designs within my solution.

Each element within the game is an object, giving me flexibility within the game to perform many functions within the runtime.

I am using inheritance to try to reduce the quantity of redundant code and allow easier modification in the future to add more functions within the future, including more monster and room types, larger dungeons or dungeons with multiple floors, etc.

Implementing monsters to be an Object allows for quick additions of different types to be made, due to all monsters needing similar attributes. This allows each type of monster (and the Monk) to be subclasses of a super class relating to creatures.

I am using the Singleton design pattern for the dungeon, monk and the print class. This is because both can only exist as a single instance and should retain the changes made to them throughout the different run times within the lifecycle of the game. All are called repeatedly within the functions of other methods as those methods access or modify them. Being able to call the instance without



being tied to a name or risk changing a local version and not saving the changes is very beneficial within this context.

The dungeon generation uses a recursive function to create a cascading effect of rooms being generated from the treasure room outwards. It checks that the next room is possible, then randomly selects whether to add that room or not. As I have used an OO approach for this, the dungeon can search rooms easily by using room methods on the room objects within the vector.

The combat is contained within a while loop that runs while both characters are alive. This is how a fight would happen in this situation so makes sense here. The monk is unable to pray in a room with a monster, either living or dead, meaning that if the player wanted to heal they may need to backtrack one or many rooms to be able to get to a location they can pray from.

## Interface design and UI components

Before I found I didn't have the time to do so - I wanted to include a list of descriptions that could be allocated to each room that, when the room is explored is shown to the player. This would be to dynamically create a story around the player.

For the game in its current state, the user uses the number pad to interact with the game. These are the keybinds:

[7] Attack	[8] North	[9] Defend
[4] West	[5] Print Data	[6] East
[1] Print Map	[2] South	[3] Heal/Pray

As it stands, the player is unable to change the keys to suit the machine, which is an oversight that could be altered in the future.

[5] is used to print data to the screen. This includes the monk's name, current health, number of rooms cleared, number of kills, current room and number of prayers left.

[3] is used to pray to set the monk's health to full. This can only be done in a room where the `monsterRoom` bool is false.

Unfortunately, the user needs to hit enter to register their input. As it is a single key press that needs to be registered, it would be possible to automatically enter the data on key press – but I didn't find a way to do this.

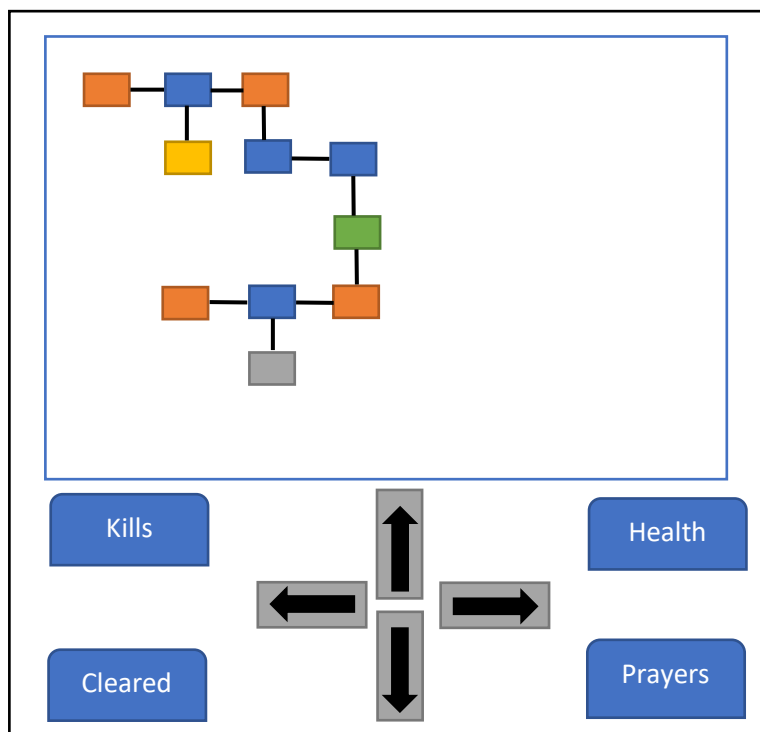
Currently the interface looks like this:

```
6 X 6 Dungeon
[Emp] [ E ]-[ M ]-[ M ]-[ M ]-[ E ]
      |   |
[ E ] [ E ]-[ S ] [ T ]-[ E ]-[ M ]
      |   |
[ E ]-[ E ] [ M ]-[ M ] [ M ] [ M ]
      |   |
[ E ]-[ M ]-[ M ]-[ M ]-[ E ]-[ M ]
      |   |
[Emp] [ E ] [ M ]-[ E ] [Emp] [Emp]
      |
[Emp] [Emp] [ E ] [Emp] [Emp] [Emp]

You are able to move:
[8] North [4] West
Or press [5] to see statistics [3] to pray or [1] for the map.
```

If I were to create a GUI version in the future, I would add dedicated on-screen buttons with associated key binds to give the user more options and have the map always on, also showing where the player is.

Here is a rough example:



In the above example, each room would have a meaning, e.g. Yellow for Treasure, Grey for spawn, green for current Location, Orange for Monster and Blue for Empty. There would be on-screen buttons and persistent displays of data including health and prayers left.

## Implementation Notes:

These are the main problems I faced during the development. These are not necessarily in order of occurrence.

Problem	Solution
<b>I was unable to return an array from a method.</b>	I was able to change the method to return a pointer to an array rather than the array itself.
<b>I was unable to define the length of a 2D array at run-time.</b>	I changed from using a 2D array to a 2D Vector.
<b>The 2D vector was complicated to implement and threw lots of issues.</b>	I changed to a 1D Vector.
<b>How do you move North in a horizontal line?</b>	I was able to use the number of rooms in a row, along with a room's X and Y to get an index. $\text{Index} = (Y * \text{sizeX}) + X$
<b>I was unable to store different types of rooms in the vector.</b>	I removed the inheritance from the rooms and used an Integer to define what type it is. The Win condition moved to the Dungeon rather than the room. All Room objects are now the same object but have different values.
<b>Random integers were not random.</b>	Variables are stored in memory which has numerical addresses. I created two integer values whose contents were their own addresses as integers and used some maths to create a seed for the random.
<b>I could only generate one room after the treasure room.</b>	I changed the location of the recursion and changed the way doors are checked.
<b>Dead monsters can still attack.</b>	I added a check that the monster is alive still after you attack. If you have the same speed they still attack as you both attack at the same time.
<b>I kept getting Null Pointer exceptions from accessing the rooms.</b>	I entered the formulae wrong. I put $(Y + \text{sizeX}) + X$ rather than $(Y * \text{sizeX}) + X$ (I spent three hours on this.)
<b>Re-entering a monster room after killing it would start combat again.</b>	I added a win condition to a fight that sets cleared to true, and monsterRoom to false.
<b>Monsters try to heal at full health.</b>	I added a basic condition where at full health a monster can only attack.
<b>If the user added a space " " inside the description the game crashed and wouldn't allow an input.</b>	CIN reads a command, of which a space is not part of and usually ends a command. I changed how the input is processed for the name and description and this was fixed.

<b>I was able to make many dungeon instances even as a Singleton.</b>	I created DungeonSuper and inherited from it, making it abstract in the process. For some unknown reason this fixed the problem and makes it possible to add a new dungeon type in the future.
<b>The user was unable to enter an integer.</b>	The input is assumed to be a string. I added a method where required that checks that the input is an integer and converts it to an integer. This integer is then used over the input.
<b>My CMake list got corrupted with no backup!</b>	I made a new project, which generates a CMake, and remade each class. I copied all code from one to another. (In future I need to make better use of Version Control)
<b>I couldn't create text files.</b>	I created another singleton class that is dedicated to creating and updating files. Rather than using COUT within the program, I call a print function within the FileOut class.

### To run the game.

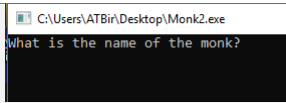
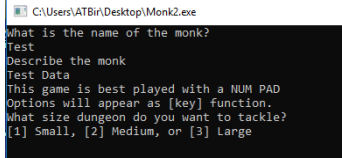
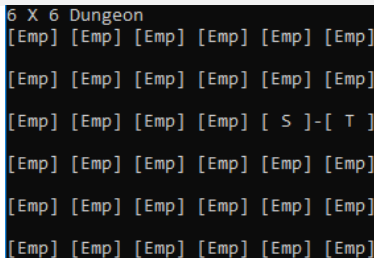
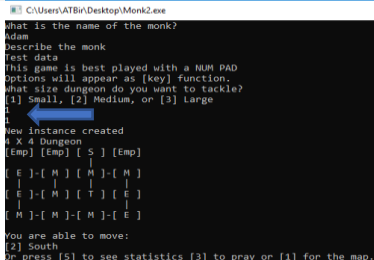
To run the game, it can either be opened as a project in CLion or Monk.exe (found in cmake-build-debug, has a shortcut in the main file) can be run.

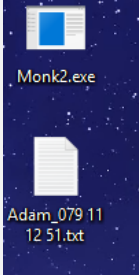

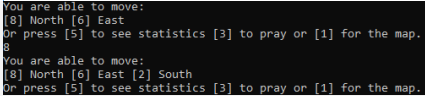
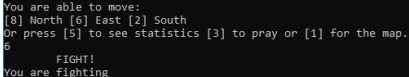
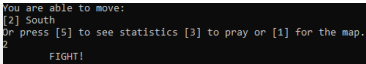
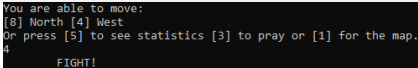
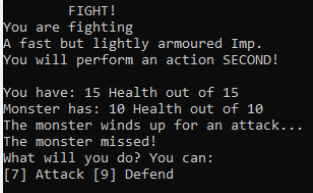
Txt files are created within the run time of the game. These can be found in the cmake-build-debug folder (or wherever the Monk.exe is installed).

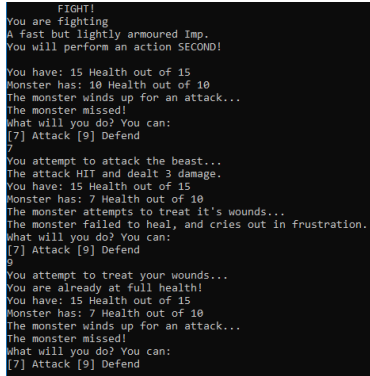
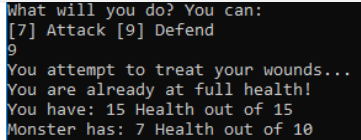
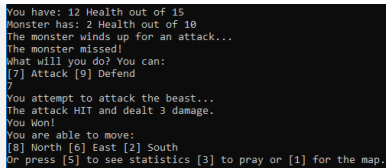
## Testing

All tests were conducted by human usage of the standalone exe file, installed to the desktop. As the exe is the version I will be using in the demo, it is the version I will be testing. Although it shouldn't be the case, any issue may be just the exe version or work for the exe but not the project version.

Failed tests are referenced by their number below the table saying why they failed and how they have / could be fixed.

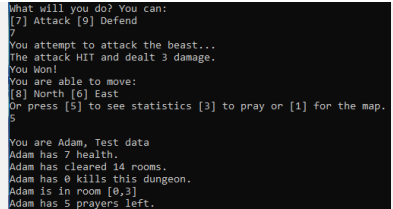
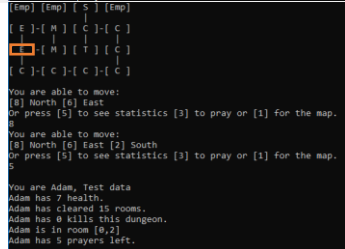
TEST NUMBER	WHAT'S BEING TESTED	HOW IT'S BEING TESTED	TEST RESULTS	PASS / FAIL	EVIDENCE (WHERE APPLICABLE)
1	Does the exe run?	Open the exe from the desktop	The window opens	Pass	
2	Can the user enter data?	Enter the name Test as the player name, and Test data as the Description.	The window accepts the user input and continues to run the game.	Pass	
3	The user can create a dungeon of three sizes.	Create three dungeons of small, medium and large	Three dungeons were created.	Pass	Figures 1, 2, and 3 in the table below
4	Does the dungeon add doors to invalid rooms.	I will generate several dungeons and no [EMP] rooms should have doors.	No dungeon (that I tested) displayed invalid doors.	Pass	
5	No dungeon should spawn less than 3 rooms.	I will generate several dungeons and check they all have more than 3 rooms.	I generated a dungeon with only 2 rooms.	FAIL	See the image above.
6	The user can select the size of the dungeon.	Try to enter 1, 2, and 3 for the input.	The input must be given twice.	FAIL	

7	The program can generate a file.	Run the program until the point above	A new file, Adam_079 11 12 51.txt, was created on the desktop	Pass		079 – 79 <sup>th</sup> day of the year at 11:12:51
8	The program can write to the file.	Open the generated file.	The file contains the same output as test 4.	Pass		
9	The user can enter [8] to go north where appropriate.	I will press [8] to move north.	I successfully move rooms	Pass		
10	The user can enter [6] to go east where appropriate.	I will press [6] to move east.	I successfully move rooms (into a monster room)	Pass		
11	The user can enter [2] to go south where appropriate.	I will press [2] to move south.	I successfully move rooms (into a monster room)	Pass		
12	The user can enter [4] to go west where appropriate.	I will press [4] to move west.	I successfully move rooms (into a monster room)	Pass		
13	The user enters combat when entering a monster room.	I will enter a monster room and combat will be triggered.	I initiated combat with a faster monster than me.	Pass		

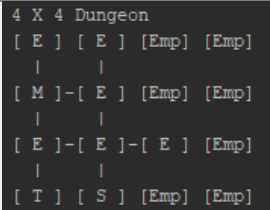
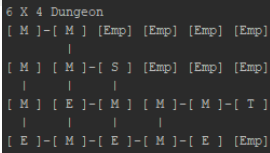
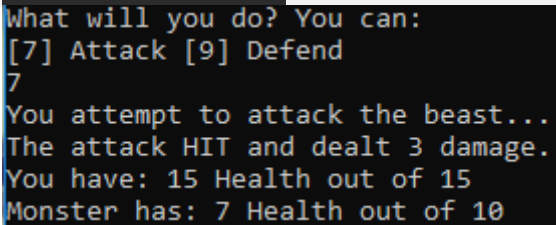
14	Both characters can attack and defend.	I will press [7] and [9] to attack and defend and check that the monster does as well.	I can attack and defend, as can the enemy monster.	Pass	 <pre> FIGHT! You are fighting A fast but lightly armoured Imp. You will perform an action SECOND!  You have: 15 Health out of 15 Monster has: 10 Health out of 10 The monster winds up for an attack... The monster missed! What will you do? You can: [7] Attack [9] Defend 7 You attempt to attack the beast... The attack HIT and dealt 3 damage. You have: 15 Health out of 15 Monster has: 7 Health out of 10 The monster attempts to treat it's wounds... The monster failed to heal, and cries out in frustration.. What will you do? You can: [7] Attack [9] Defend 9 You attempt to treat your wounds... You are already at full health! You have: 15 Health out of 15 Monster has: 7 Health out of 10 The monster winds up for an attack... The monster missed! What will you do? You can: [7] Attack [9] Defend </pre>
15	Attack and Defend have a 50% hit chance.	I will run the code and some actions should hit while others miss.	Some hit and some miss, however it is hard to tell if it is 50% or some other chance.	Pass	See test 14 for a successful demonstration of hits and misses.
16	Neither character can defend when at full health.	I will press [9] at full health and should not be able to gain more than my max health.	I pressed [9] and was told "You are already at full health."	Pass	 <pre> What will you do? You can: [7] Attack [9] Defend 9 You attempt to treat your wounds... You are already at full health! You have: 15 Health out of 15 Monster has: 7 Health out of 10 </pre>
17	Successful attacks deal the characters strength in damage to the opponent.	I will press [7] and deal 3 damage to the monster, who will deal 2 damage to me.	Both characters can deal damage to the other.	Pass	Figures 4 and 5 in the table below.
18	Successful Heal adds 1 to the current health.	I will press [9] once I have taken damage, and the monster can also heal.	Both characters can gain 1 health at a time.	Pass	Figures 6 and 7 in the table below.
19	Combat ends when one character dies.	I will fight the monster to the death.	I killed the monster and combat ended.	Pass	 <pre> You have: 12 Health out of 15 Monster has: 2 Health out of 10 The monster winds up for an attack... The monster missed! What will you do? You can: [7] Attack [9] Defend 7 You attempt to attack the beast... The attack HIT and dealt 3 damage. You Won! You are able to move: [8] North [6] East [2] South Or press [5] to see statistics [3] to pray or [1] for the map. </pre>

20	The character with the higher speed stat moves first.	Initiate combat with different monsters.	The character with higher speed makes a move first. Error with the display for the troll.	FAIL	Figures 8, 9, and 10 in the table below.
21	The user can enter [5] to see information.	Press [5] and expect Adam and Test data to be the monk's name and description.	Detailed information is shown about the current progress.	Pass	<pre> You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You are Adam, Test data Adam has 15 health. Adam has cleared 1 rooms. Adam has 0 kills this dungeon. Adam is in room [2,0] Adam has 5 prayers left.  You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>
22	The user can enter [1] to display the map.	Press [1] and expect the same map as before.	The map showed the same as before.	Pass	<pre> You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 1 4 X 4 Dungeon [Emp] [Emp] [ S ] [Emp]  [ E ]-[ M ] [ M ]-[ M ]       [ E ]-[ M ] [ T ] [ E ]       [ M ]-[ M ]-[ M ]-[ E ]  You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>
23	The user can't pray when at full health.	Press [3] to pray, I expect a message and no prayer to happen.	I am told "You are already at full health."	Pass	<pre> You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You are already at full health... You are able to move: [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>
24	The player can press [3] while damaged to pray.	I will enter combat, take damage and pray.	I can pray in rooms marked as empty.	Pass	<pre> You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You successfully pray and the gods heal your wounds. You have 5 prayers left. You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>
25	The player can heal up to 5 times.	I will pray and watch the counter go down.	The counter never changes from 5	FAIL	<pre> You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You successfully pray and the gods heal your wounds. You have 5 prayers left. You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 You are Adam, Test data Adam has 15 health. Adam has cleared 10 rooms. Adam has 0 kills this dungeon. Adam is in room [3,2] Adam has 5 prayers left.  You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>
26	The number of rooms cleared is correct.	Count the number of rooms I have been in and check it is the same as the display.	It counts every time I walk through a door.	FAIL	<pre> You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 4 X 4 Dungeon [Emp] [Emp] [ S ] [Emp]  [ E ]-[ M ] [ C ]-[ C ]       [ E ]-[ M ] [ T ] [ C ]       [ M ]-[ C ]-[ C ]-[ C ]  You are Adam, Test data Adam has 15 health. Adam has cleared 10 rooms. Adam has 0 kills this dungeon. Adam is in room [3,2] Adam has 5 prayers left.  You are able to move: [8] North [2] South Or press [5] to see statistics [3] to pray or [1] for the map. 5 </pre>



27	The number of Kills in the dungeon is correct.	I will kill a monster and watch the counter increase.	The counter doesn't increase.	FAIL	
28	The displayed index is correct.	I will count the room and calculate the index.	The room is correct. Highlighted room is [0,2] with my index scheme.	Pass	
29	The game ends if you reach the treasure room.	I will enter the treasure room and win.	The game terminated and wrote to the file.	Pass	Figures 11 and 12 in the table below.
30	The game ends if you die.	I will try to be killed in combat.	I died and the game ended, writing to the file.	Pass	Figures 13 and 14 in the table below.

## FIGURE IMAGE

1		
2		
3		
4		

5	<p>What will you do? You can:          [7] Attack [9] Defend          7          You attempt to attack the beast...          You missed!          You have: 15 Health out of 15          Monster has: 8 Health out of 10          The monster winds up for an attack...          The attack HIT and dealt 2 damage.          You have: 15 Health out of 15</p>	
6	<p>Monster has: 8 Health out of 10          The monster winds up for an attack...          The attack HIT and dealt 2 damage.          What will you do? You can:          [7] Attack [9] Defend          9          You attempt to treat your wounds...          You successfully heal 1 health point.          You have: 14 Health out of 15          Monster has: 8 Health out of 10</p>	
7	<p>What will you do? You can:          [7] Attack [9] Defend          7          You attempt to attack the beast...          You missed!          You have: 15 Health out of 15          Monster has: 7 Health out of 10          The monster attempts to treat it's wounds...          The monster successfully heals 1 health point.          What will you do? You can:          [7] Attack [9] Defend          7          You attempt to attack the beast...          You missed!          You have: 15 Health out of 15          Monster has: 8 Health out of 10</p>	
8	<p>FIGHT!          You are fighting          A fast but lightly armoured Imp.          You will perform an action SECOND!</p>	
9	<p>FIGHT!          You are fighting          A goblin of balanced health and speed.          You will perform an action AT THE SAME TIME!</p>	
10	<p>FIGHT!          You are fighting          A slow but heavily armoured troll.          You will perform an action FIRST!          You will perform an action SECOND!</p>	
11	<p>You are able to move:          [8] North [6] East [2] South          Or press [5] to see statistics [3] to pray or [1] for the map.          2          You found the Treasure Room          Press any key to continue . . .</p>	
12	<p>You are able to move:          [8] North [6] East [2] South          Or press [5] to see statistics [3] to pray or [1] for the map.          You found the Treasure Room          Press any key to continue . . .</p>	
13	<p>The monster winds up for an attack...          The attack HIT and dealt 4 damage.          OH NO! You died.          Press any key to continue . . .</p>	

14      The monster winds up for an attack...  
           The attack HIT and dealt 4 damage.  
           OH NO! You died.  
           Press any key to continue . . .

## Failed Tests

FAILED TEST NUMBER	WHY IT FAILED	HOW TO FIX	IS IT FIXED?
5	I generated a dungeon with only 2 rooms.	I can add a Room Count to the dungeon. If Room Count is too low, generate a new dungeon. This will take too much time to be able to fix before the demo.	No
6	The dungeon size input must be given twice.	I can change the way the input is verified to ensure that all valid inputs are processed.	No
20	Error with the display for the troll.	The logic for printing the order needed to be changed as I left three if statements independent when they should be else if statements.	Yes
25	The prayers left counter never changes from 5	I forgot to add a statement to decrement the prayers left.	Yes
26	It counts every time I walk through a door.	Extra logic needs to be added to only increase it when the user clears a room, not when they enter.	Yes
27	The kill counter doesn't increase.	I forgot to use the method to set the kill.	Yes

## Evaluation

I came across many difficulties within the implementation of the solution. One of the biggest setbacks was that a method in C++ is unable to return an array, unless you use a pointer. As I was using a 2D array at the time I needed three pointers – E.G. `return ***ArrayName;` As I needed one pointer per dimension of the array and another for the Room class type – as Room must also be a pointer.

This required lots of changes to my plans as I didn't realize that this was an issue I would have. The change to a vector was necessary as they are more suitable for this implementation and can be easier to interact with between methods. However, using a 2D vector became complicated so I changed to use a 1D vector. The maths for this became problematic at times but was easier than always using two dimensions.

I believe the singleton was the best choice design pattern for the three classes I used it on but feel that I didn't exploit other design patterns to their fullest, making the project more difficult than it needed to be. I used singleton for the printing class I created as there will only ever be one file to output to, and it is better to store all required classes in that class, including the generation and storage of the filename. There are several print commands including a `println` and `print` to give more options for future development.

If I were to create a new implementation in the future, I would have a clearer idea about design patterns in mind as I never considered MVC or any other standard for programming which likely makes my code more difficult for another person to pick up and maintain in the future. I would also go into the development with a clearer plan of the order I would implement things in. I worked on base classes first and developed the implementation after. I thought this would be a good way to do this however I then needed to change lots of things within my classes to make them work. If I were to do this again, I would concentrate on one functionality and then make the classes for the next functionality.

I would also want to have a more visual interface, even if it is an interface drawn with computing symbols in a command line. However, as I am not a very artsy person I would need to source some online, which may be a copyright infringement outside of an educational environment.

I believe I did well on this project. The only real issue is that I can create a dungeon of only 2 rooms, however when a good dungeon is generated it is a long and decently populated dungeon. I added 2 more monster types and more can be easily added in the future and are implemented automatically with the way I generate the monsters in the rooms. Other than the possibility of generating a dungeon of 2 rooms, I believe I have fully met and exceeded the problem statement. I can generate 3 dungeon sizes, and more can be added in the future. I have created 3 types of monster that all play differently with the possibility of more being added in the future.