Hippo's Quest:

Game scenario:

When the game is loaded up you spawn in at a castle that is burning and on fire from dragon fire. So, the program will encourage you to explore the area in search of keys and random objects that may be required in order to allow you to escape the burning labyrinth.

Commands:

In the game you have the ability to move between rooms, as you can choose to either go north, south, west or east. But, the exit of each room is known, in order to make the experience more pleasurable for the user, to accommodate for varying skill levels.

Items:

During the game you will have to collect a water bucket in order to cross the burning bridge, a sword and a shield, in order to defeat the dragon, and you need to gather ingredients and equipment in order to create the perfect sandwich for a magical hippo.

To do this you will need to enter a room a type pick up in order to successfully store that item that has been found within your inventory.

However, there is also a limit to how many items can be carried in the backpack that the player is presented with, as they cannot carry more than 5 items, and so they will be forced to choose with items to carry and which to discard.

Win condition:

In order to beat the game you need to successfully obtain all the items from every location within the game to ensure that you can pass each location, and get past the three major obstacles within the game.

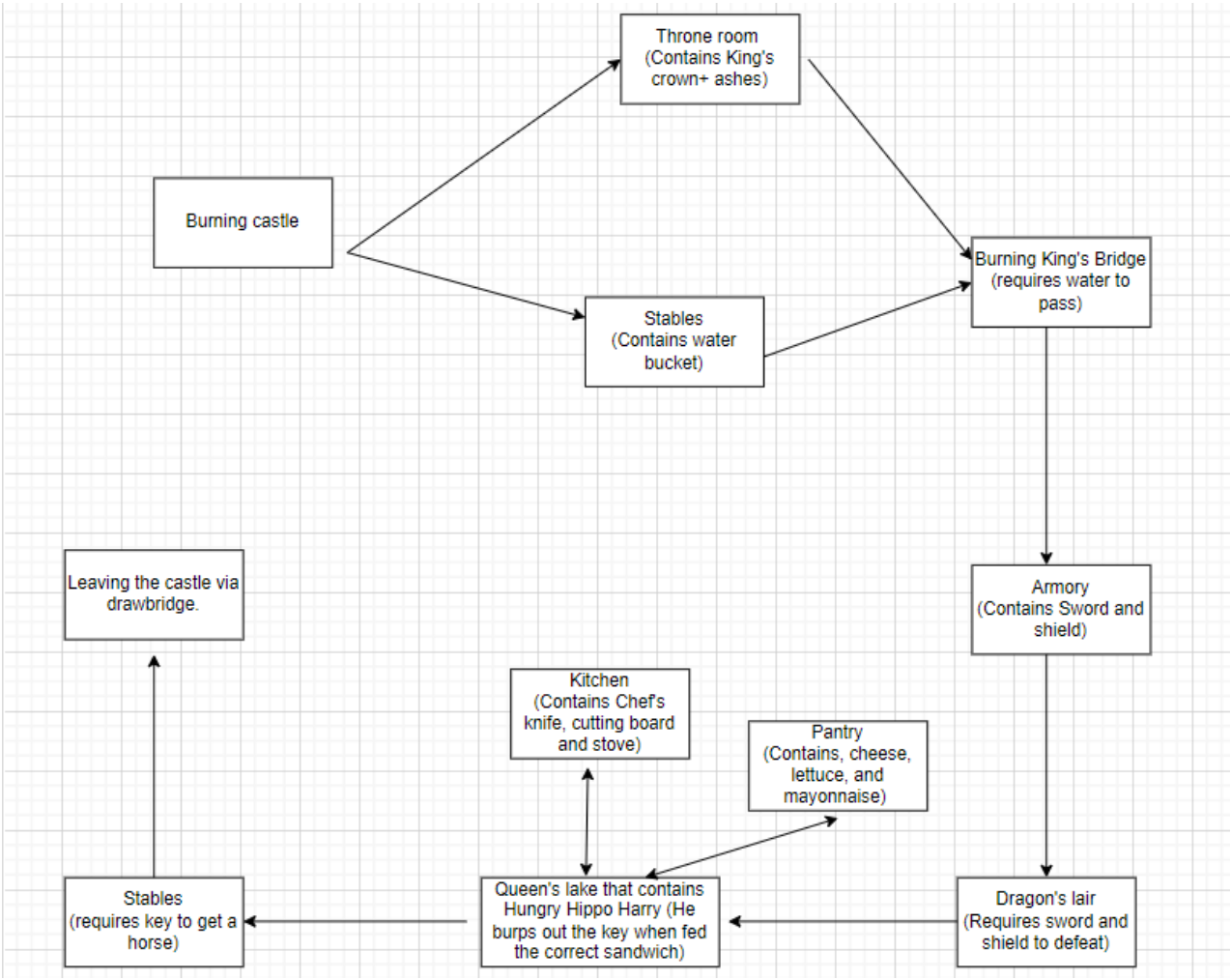
Locations:

The player has the ability to travel to 11 location within the game, and has the ability to move in whichever direction they would like. However, the exits to each location will only be usually in one direction, and some rooms contain branch off into multiple locations, as part of the quest.

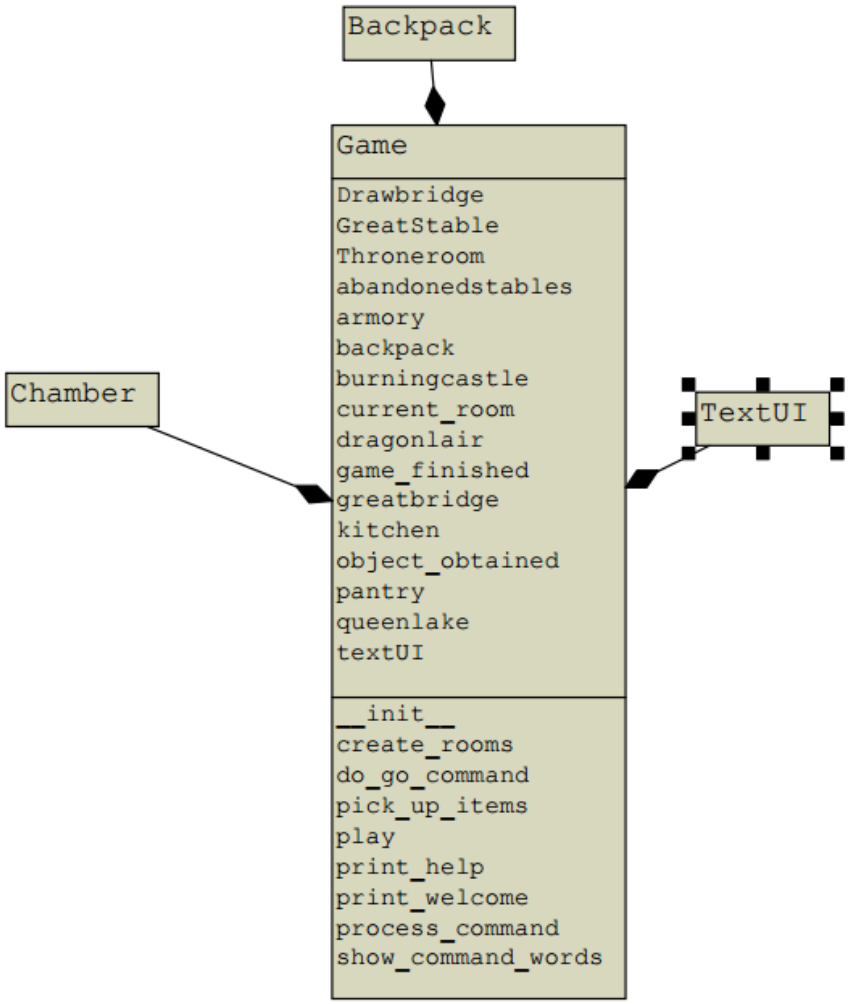
How to launch:

The game is run through PyCharm, so what you will need to do is simply run the game through clicking the run button, which can be found at the top right hand side of the application, once the game.py has been opened.

4. Game map


Throne room
(Contains King's crown+ ashes)
Burning castle
Burning King's Bridge (requires water to pass)
Stables
(Contains water bucket)
Leaving the castle via drawbridge.
Armory
(Contains Sword and shield)
Kitchen
(Contains Chef's knife, cutting board and stove)
Pantry
(Contains, cheese, lettuce, and mayonnaise)
Stables
(requires key to get a horse)
Queen's lake that contains Hungry Hippo Harry (He burps out the key when fed the correct sandwich)
Dragon's lair
(Requires sword and shield to defeat)

5. UML Diagram:



6. How I modified the starter code:

I made minimal modifications to the starter code that was provided to me, in order to allow the complexity of the game to shine through using other coding techniques, and to allow for easy modification.

To begin in the initialise definition I added create room to allow for the creation of rooms, and the current room to make it clear which room it was the user begins from. Also, I set backpack capacity to ensure the capacity begins at 0, to ensure the game only finishes when required I set game_finished to false, to use later.

Within the game I also added logging in order to keep track of all the changes that are made throughout the game by the player. So, if the player chooses to go in a particular direction that will be logged.

Furthermore, within the room.py code I modified that to chambers.py, as I included room label, and room_items in order to allow each room to have a name, and to

allow each room to store items inside of it when they have not been picked up by the user, and then inserted the relevant information for each chamber into game.py.

Additionally, within the game.py I created the exits for each chamber to allow one chamber to link to the other, and to allow the user to travel to more than one location from a particular location, due to the branching off of multiple rooms, in some instances.

In addition, I made some minor modification to the welcome messages that were displayed to be unique to my game setting.

Furthermore, I added pick up items as a command that the user can perform. Then, when defining this I made modifications to the process command in order to ensure that this runs smoothly under different conditions, by using if and else if statements, for multiple rooms.

Moreover, I created a definition for pick up items to ensure that the appropriate items are being picked up within each room, and also I coded this to ensure that the appropriate text is displayed for the user, according to the item obtained, whilst ensuring that the backpack capacity does not exceed 5.

7. interesting features:

One interesting feature I coded into the game, was the ability to move back to certain rooms. This ensures that the individual picks up the correct item before they are allowed to progress onto the next room, as they remained locked without the acquisition of such items.

Additionally, when the user is seeing the giant hippo they are able to move between multiple rooms, and obtain multiple items in order to satisfy the hippo, which will then allow them to pass him.

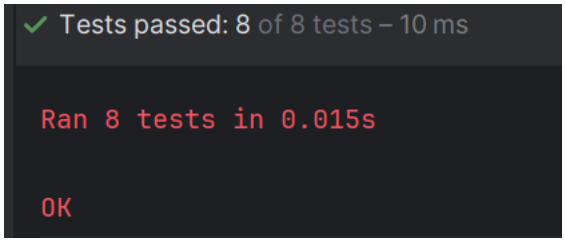
8. Automated unit testing:

I performed 8-unit tests, in order to fully test that each function was working appropriately, but I have attached this to the end submission also.

```
import unittest
from chamber import Chamber
from backpack import Backpack
from gamez import Game

class MyTestCase(unittest.TestCase):
    def setUp(self):
        self.b1 = Backpack(6)
        self.c1 = Chamber( room_label: 'burning castle', description: "You have entered the burning castle", room_items: [])
        self.c2 = Chamber( room_label: 'abandoned stables', description: "You have entered the abandoned stables", room_items: ["water bucket"])
        self.c1.set_exit( direction: "east", self.c2)
        self.c2.set_exit( direction: "west", self.c1)
        self.g1 = Game()

    def tearDown(self):
```


Tests passed: 8 of 8 tests – 10 ms
Ran 8 tests in 0.015s
OK