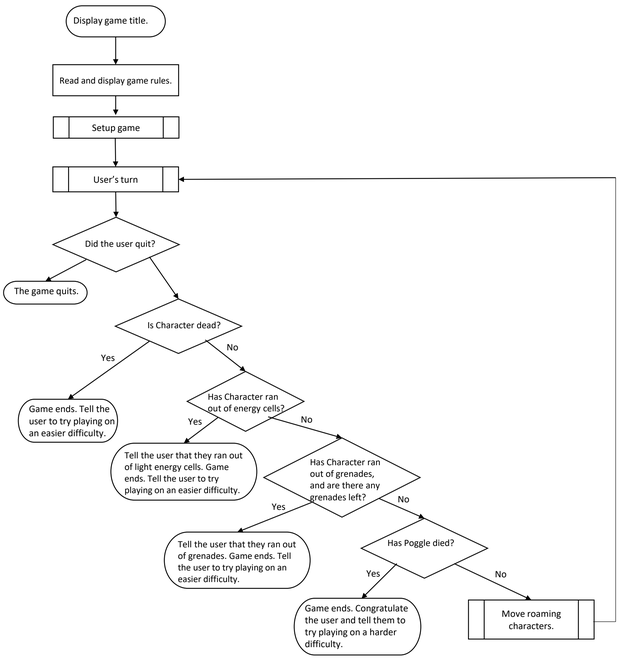
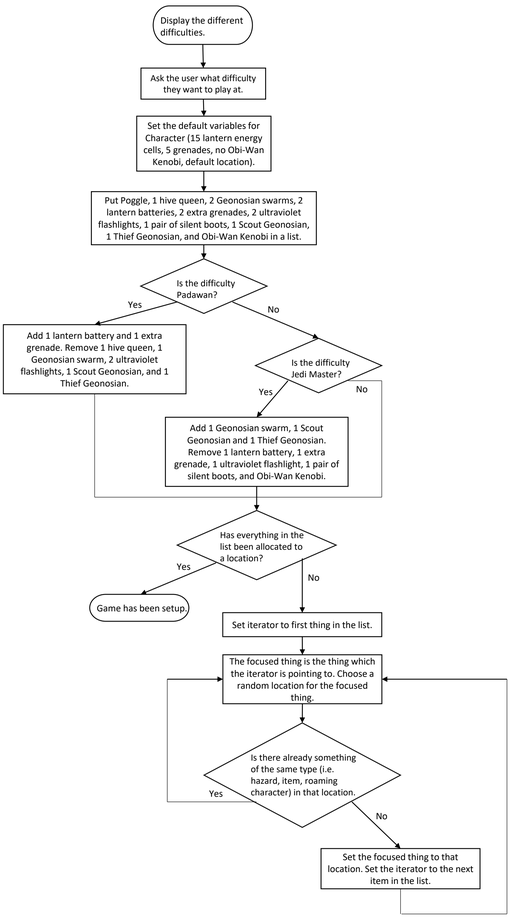
Josiah Schuller 31466400

Master game flowchart:



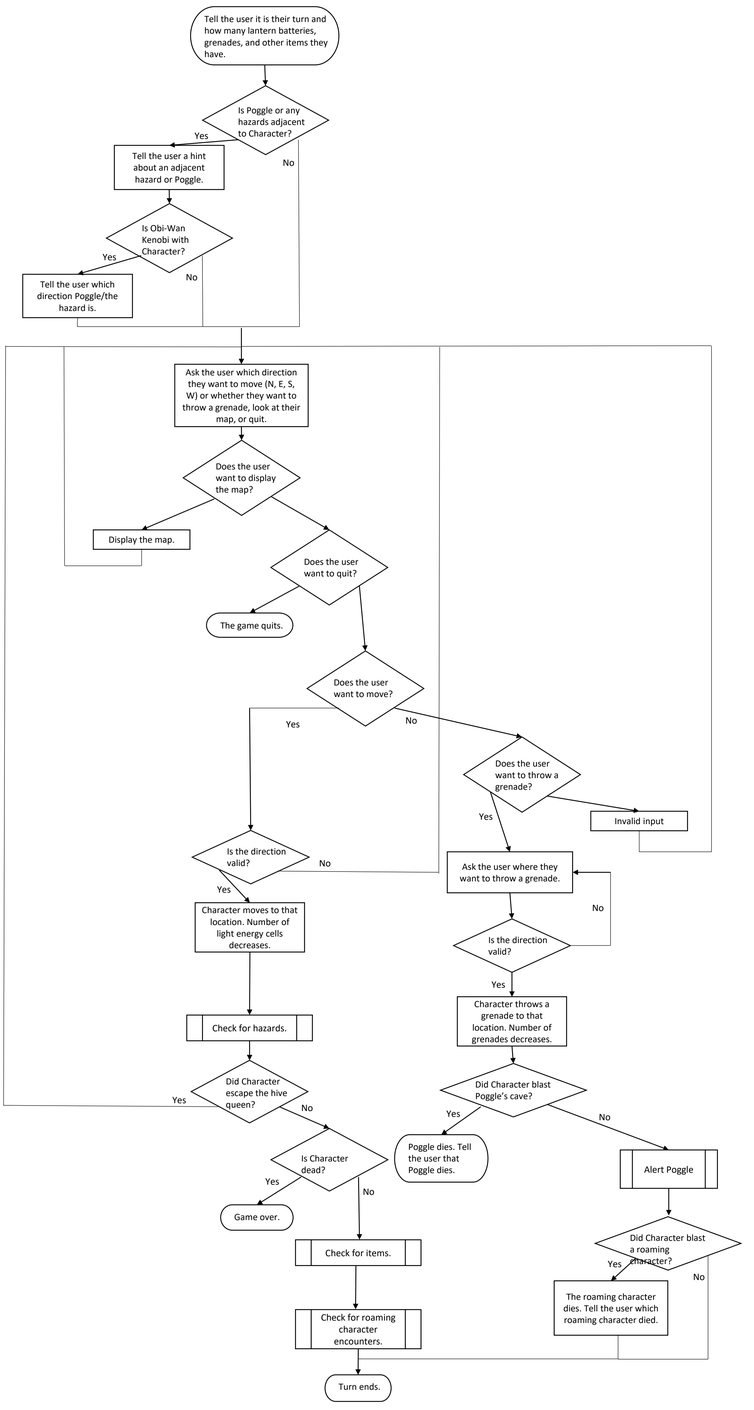
Josiah Schuller 31466400

Setup game flowchart:



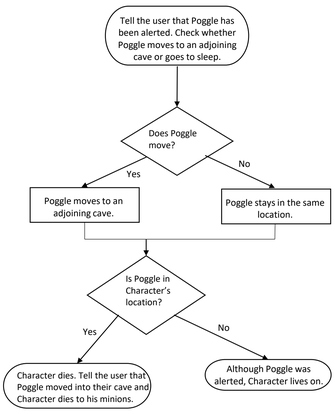
Josiah Schuller 31466400

User's turn flowchart:



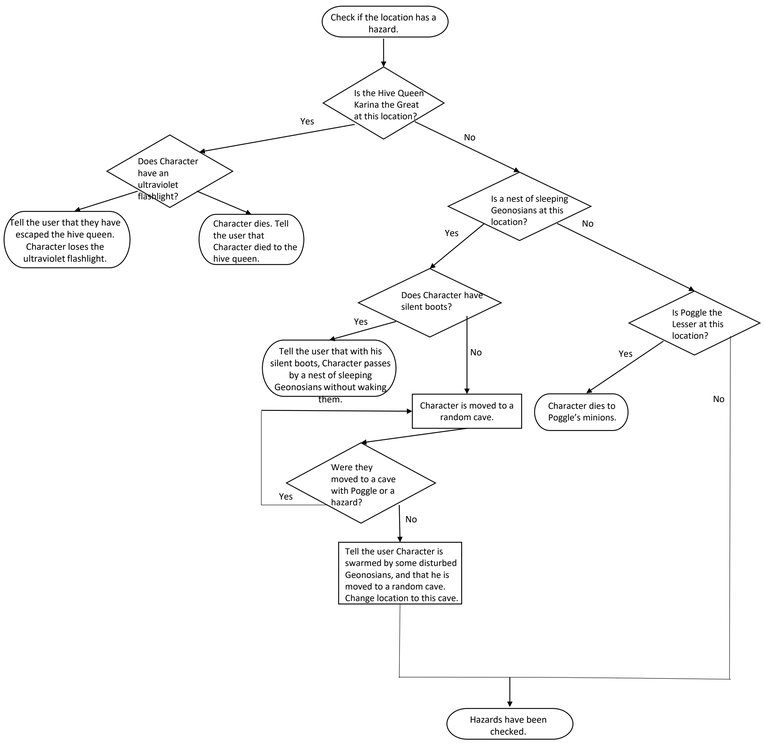
Josiah Schuller 31466400

Alert Poggle flowchart:

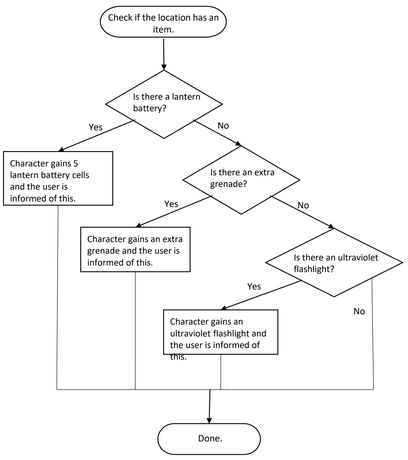


Josiah Schuller 31466400

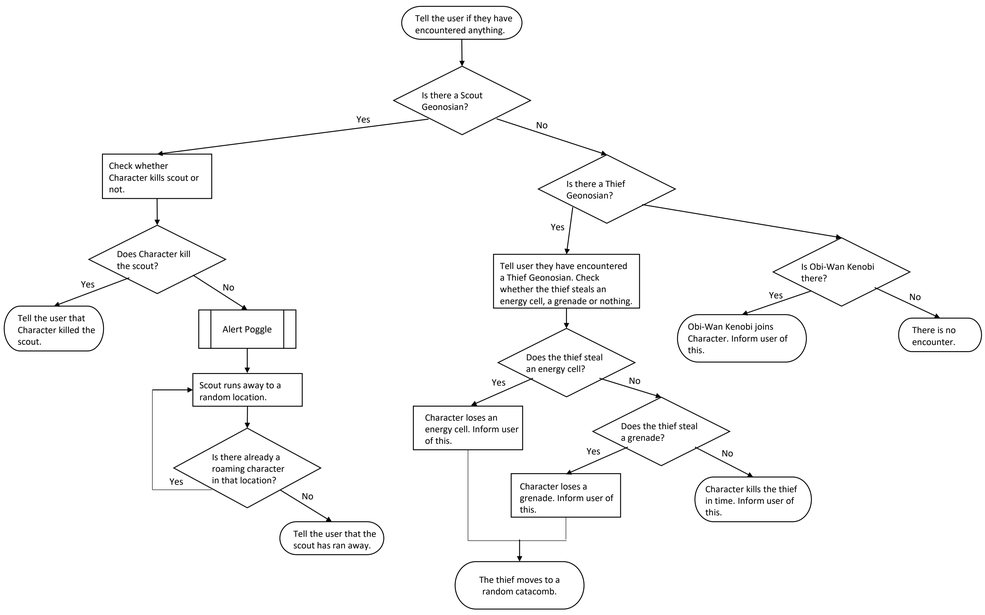
Check for hazards flowchart:



Check for items flowchart:

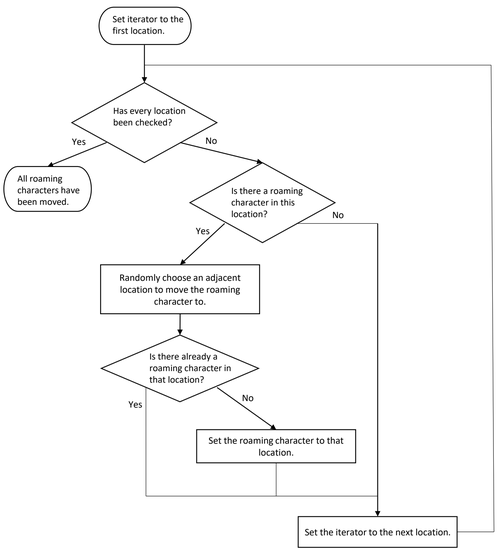


Check for roaming character encounters flowchart:



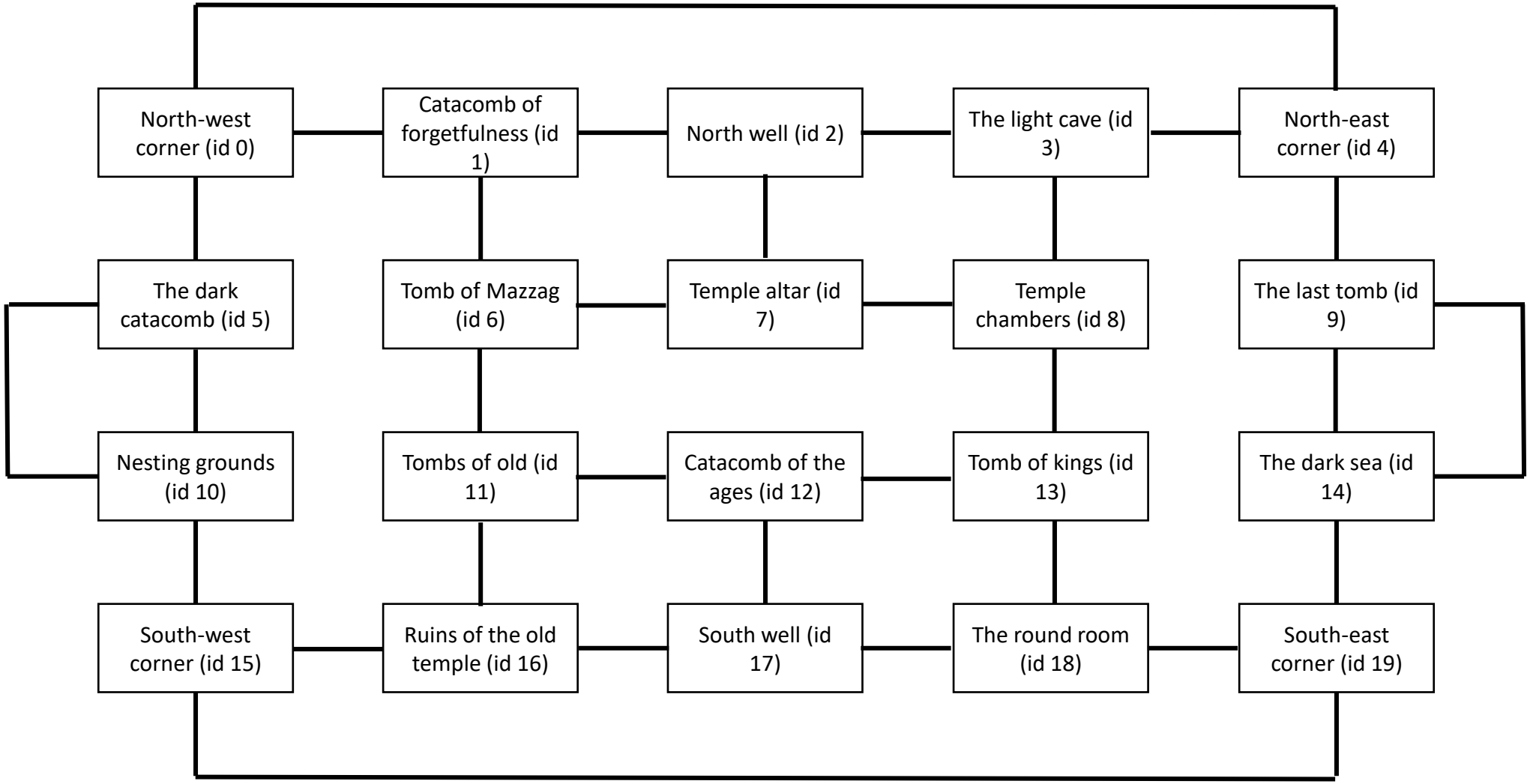
Josiah Schuller 31466400

Move roaming character flowchart:



**Map:**

Josiah Schuller 31466400



**UML Diagrams:**

| **Character** | | **Catacomb** |
|---|---|---|
| - string name<br>- int currentCatacomb<br>- int lanternEnergyCells<br>- int grenades<br>- vector<Item*> items<br>- bool hasKenobi<br>- bool gameEnded | | - string catName<br>- string catDescription<br>- string catExits<br>- Item* catItem<br>- Hazard* catHazard<br>- RoamChar* catRoamChar<br>- int northCatacomb<br>- int eastCatacomb<br>- int southCatacomb<br>- int westCatacomb |
| + Character()<br>+ Character(string newName)<br>+ ~Character()<br>+ string getName()<br>+ int getCatacomb()<br>+ int getLanternEnergyCells()<br>+ int getGrenades()<br>+ vector<Item*> getItems()<br>+ bool getHasKenobi()<br>+ bool getGameEnded()<br>+ void setCatacomb(int catacomb)<br>+ void setLanternEnergyCells(int cells)<br>+ void setGrenades(int grnds)<br>+ void setHasKenobi(bool kenobi)<br>+ void addItem(Item* item)<br>+ void removeItem(Item* item)<br>+ string displayInventory()<br>+ void gameHasEnded() | | + Catacomb()<br>+ Catacomb(string name, string description, string exits, int northIndex, int eastIndex, int southIndex, int westIndex)<br>+ ~Catacomb()<br>+ string getName()<br>+ string getDescription()<br>+ string getExits()<br>+ Item* getItem()<br>+ Hazard* getHazard()<br>+ RoamChar* getRoamChar()<br>+ int getNorthCatacomb()<br>+ int getEastCatacomb()<br>+ int getSouthCatacomb()<br>+ int getWestCatacomb()<br>+ void setItem(Item* item)<br>+ void setHazard(Hazard* hazard)<br>+ void setRoamChar(RoamChar* roamChar)<br>+ string getCatacombDetails() |
| | | |
| **Item** | | **Hazard** |
| - string name | | # string name<br># string hint<br># int currentCatacomb |
| + Item()<br>+ Item(string newname)<br>+ ~Item()<br>+ string getName() | | + Hazard()<br>+ Hazard(string newName, string newHint)<br>+ ~Hazard()<br>+ string getName()<br>+ string getHint()<br>+ int getCatacomb()<br>+ void setCatacomb(int catacomb) |

| **Poggle (child of Hazard)** | | **HiveQueen (child of Hazard)** |
|---|---|---|
| | | |
| + Poggle()<br>+ ~Poggle() | | + HiveQueen()<br>+ ~HiveQueen() |
| | | |
| **Nest (child of Hazard)** | | **RoamChar** |
| | | # string type |
| + Nest()<br>+ ~Next() | | + RoamChar()<br>+ RoamChar(string type)<br>+ ~RoamChar()<br>+ string getType()<br>+ string encounter() |
| | | |
| **Scout (child of RoamChar)** | | **Thief (child of RoamChar)** |
| | | |
| + Scout()<br>+ ~Scout()<br>+ string encounter() | | + Thief()<br>+ ~Thief()<br>+ string encounter() |
| | | |
| **ObiWanKenobi (child of RoamChar)** | | **Application** |
| | | Character* character<br>vector<Catacomb*> catacombList<br>bool testingMode |
| + ObiWanKenobi()<br>+ ~ObiWanKenobi() | | int main()<br>void displayTitle()<br>void displayRules()<br>void setupGame()<br>void setupCatacombs()<br>void assignHazard(Hazard* hazard)<br>void assignItem(Item* item)<br>void assignRoamChar(RoamChar* roamChar)<br>void playGame()<br>void userTurn(Character* character)<br>bool displayUI(Character* character)<br>void displayMap()<br>bool alertPoggle()<br>void moveRoamChars()<br>bool checkGrenades(Character* character)<br>string askForString(string question)<br>void cleanMemory() |

**Rules:**

It is the middle of the Clone Wars, a galactic war. You, Anakin Skywalker, a jedi knight, have gone in search of Poggle the Lesser, king of the Geonosians and key leader of the Separatists. Unfortunately, he has fled into a maze of cave-like catacombs underneath the great Geonosian temple. The catacombs are vast, unexplored and are very dangerous. You must search the catacombs and find Poggle but avoid the hazards that await you.

The cave system is made up 20 catacombs, each one connecting to 3 other catacombs (in a north, east, south or west direction). You have a map, a lantern with a limited amount of energy cells which runs out after wandering around for too long (your lightsabre is too loud to use as a light), and you have 5 grenades. You can throw a grenade into an adjacent cave. If you blast the catacomb with Poggle in it, he dies (and you are successful). If Poggle is not in that catacomb, you alert him. When you alert him, there is a 75% chance that he will move to an adjoining catacomb and a 25% chance that he stay in the same catacomb. If Poggle moves into your catacomb after being alerted, his minions will capture you (you lose). Also if you blunder into the catacomb where Poggle is, you lose to Poggle and his minions.

When entering a new catacomb, if you are one catacomb away from Poggle or a hazard, your jedi senses will sense what is close (but not the direction of where it lies). One such hazard is the Hive Queen Karina the Great and her horde of undead. If you stumble into her cave, you will die (unless you have an ultraviolet flashlight to blind her). If you find yourself in a cave where a nest of Geonosians are sleeping, they will swarm you and carry you to a random catacomb.

Some caves may have items in them for you to take. These include: a lantern battery adds 5 energy cells to your lantern, an extra grenade, silent boots (which allows you to walk past sleeping Geonosian swarms without waking them) or an ultraviolet flashlight which temporarily blinds the Hive Queen Karina the Great, allowing you to escape her.

There may also be roaming characters in the caves, both friendly and hostile. If you encounter a Scout Geonosian, you have a 50% of killing him. If you do not manage to kill him, Poggle will be alerted. If you encounter a Thief Geonosian, there is a 50% chance of him stealing a lantern energy cell from you, a 25% chance of stealing a grenade and a 25% chance of not stealing anything. If you encounter Obi-Wan Kenobi, he will join you. With his more experienced jedi senses, he will be able to sense the direction of where the hazards and Poggle are when you are adjacent to them.

**Program Reflection:**

In the first instance of the project (i.e. Assignment 2), I had not developed a strong understanding of classes at that point, so the design was quite simplistic. The Hazard class did not have any child classes for the types of hazards, and the Roaming Character class did not have any child classes for the types of roaming characters. So in this first instance, there were no class dependencies or parent-child structures. If I were to conduct this project again, I would change the design to include the child classes for the Hazard and Roaming Character classes right from the start.

In my design of the Catacomb class (i.e. Location class), I wanted there to be some kind of reference to the adjacent catacombs. After toying with different ideas, I decided to reference the adjacent catacombs by their index in the vector list of catacombs. This made creating the classes very simple (I did not have to create instances of all the catacombs before assigning their adjacent catacombs). The Catacomb class also contained objects of the Item, Hazard and Roaming Character classes if they existed in that catacomb.

In the second instance of the project (i.e. Assignment 3), I Implemented the child classes for Hazard (Poggle, HiveQueen and Nest) and Roaming Character (Scout, Thief, ObiWanKenobi).
Initially, the child classes of Roaming Character had their own separately named methods for their interactions with the player. However, there was no way to call these methods when the child classes are listed as Roaming Character classes. To fix this issue, I changed my design such that the Roaming Character class had a virtual method “encounter”. Then each of the child classes also had an “encounter” method, which overrides the parent class’s method. This change worked perfectly. If I were to start over again, I would make use of virtual methods from the start.

Note that I did not create child classes for the Item class, even though there are different types of items. This is because the only thing that the Item class needed to track was its name/type. Thus child classes for the Item class would have been redundant and useless.

Every single time I created an object of a class, it was created as a pointer. I did this so that there would not be any issues with referencing objects. By always using pointers, I know that any time I reference a pointer, it will always reference the same exact object.

Overall, the entire design took some time to implement. There were many inter-dependencies between the classes, which required a lot of brain power to get my head around. I made many realisations throughout the process of working on my project. If I were to start over, I would have implemented these changes right from the start.