CPSC 2720
Spring 2023

SPACE RESCUE

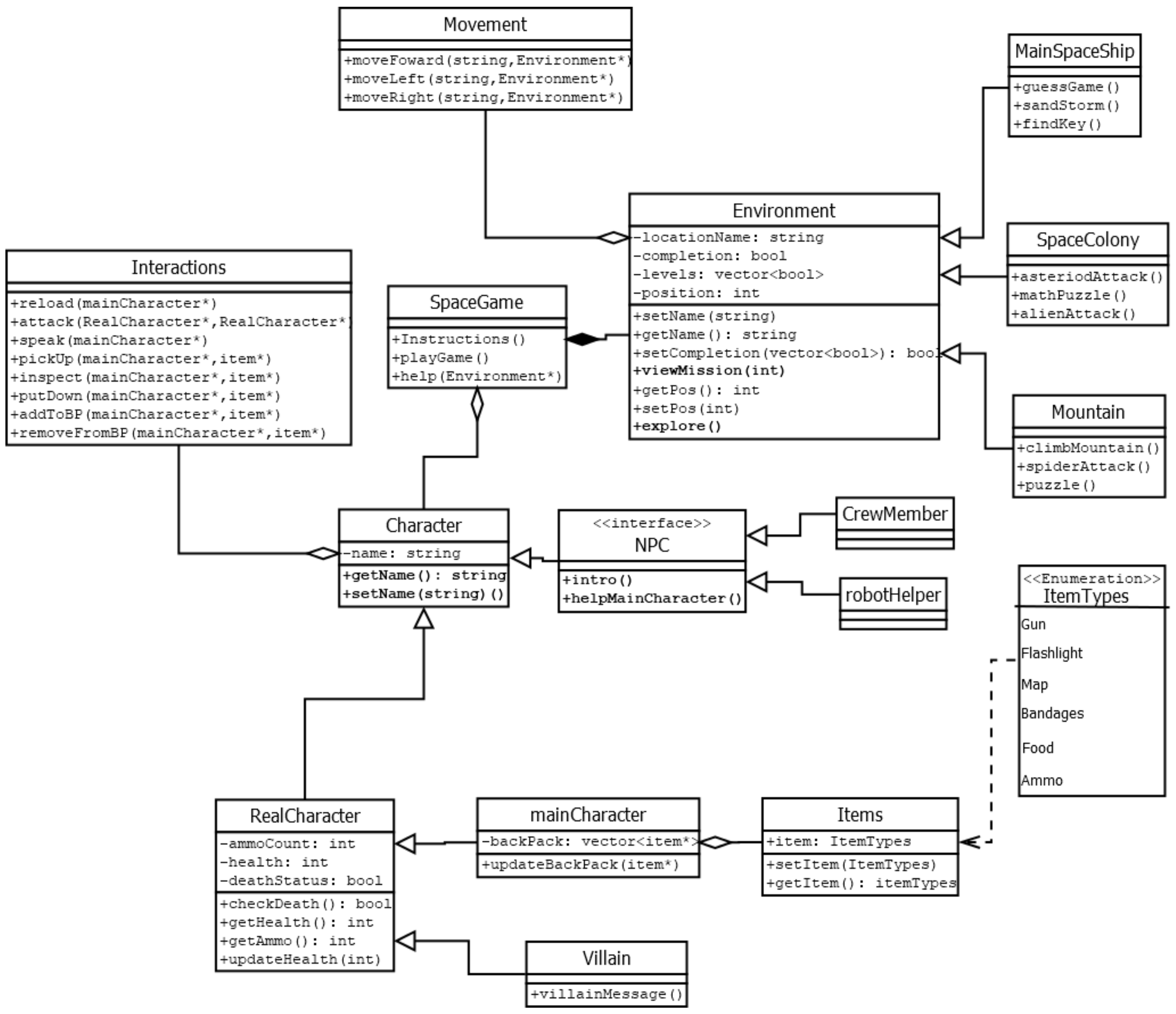
Team LEGION

Simon Rolfson
Chandan Sharma
Parmeet Pannu

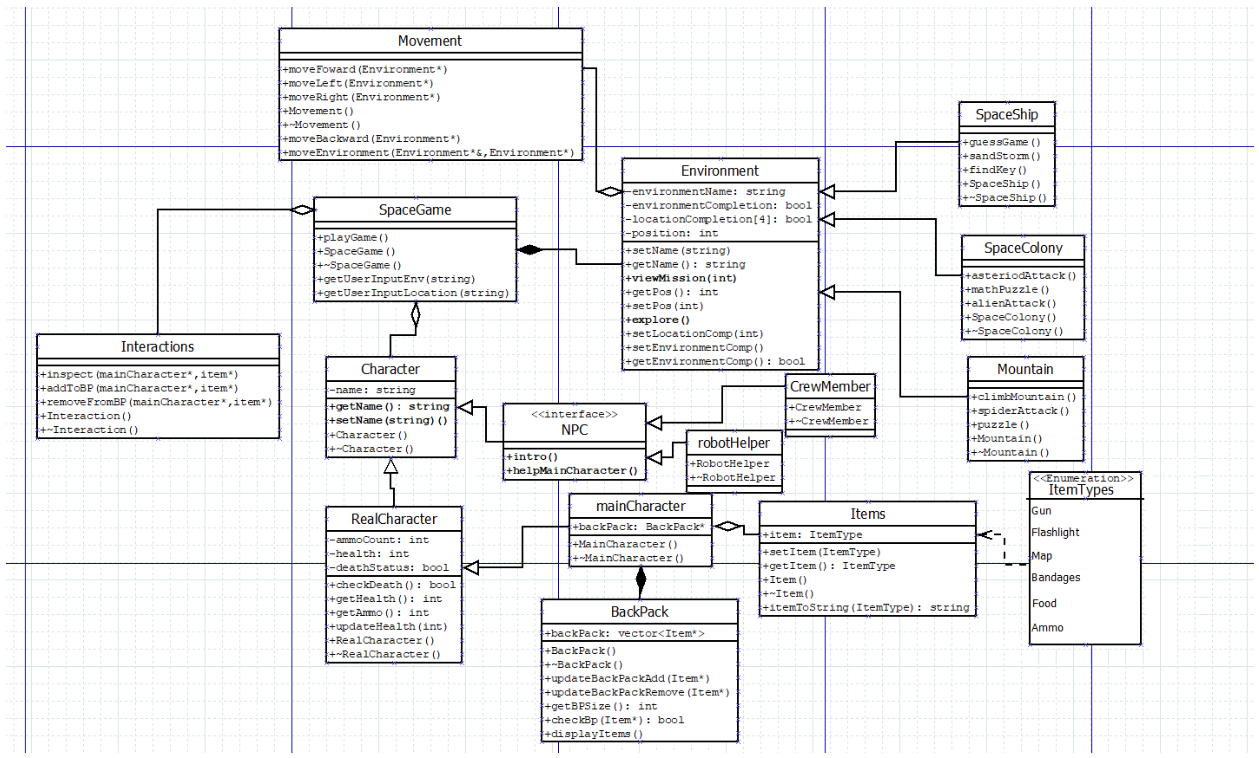
March 6th

Software Design:
<u>CLASS DIAGRAMS:</u>

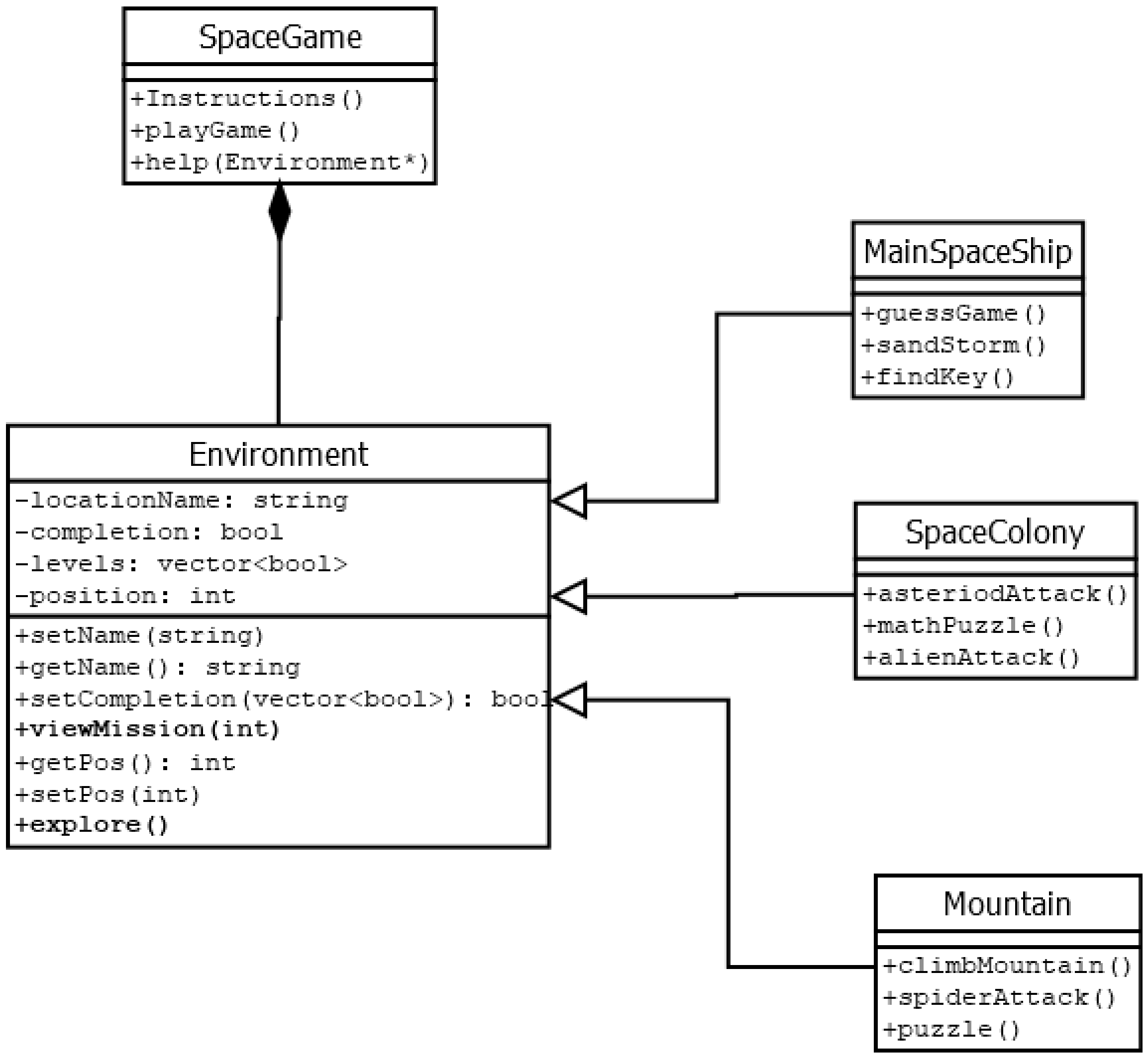
1. Whole class diagram: (showing interaction of all classes).



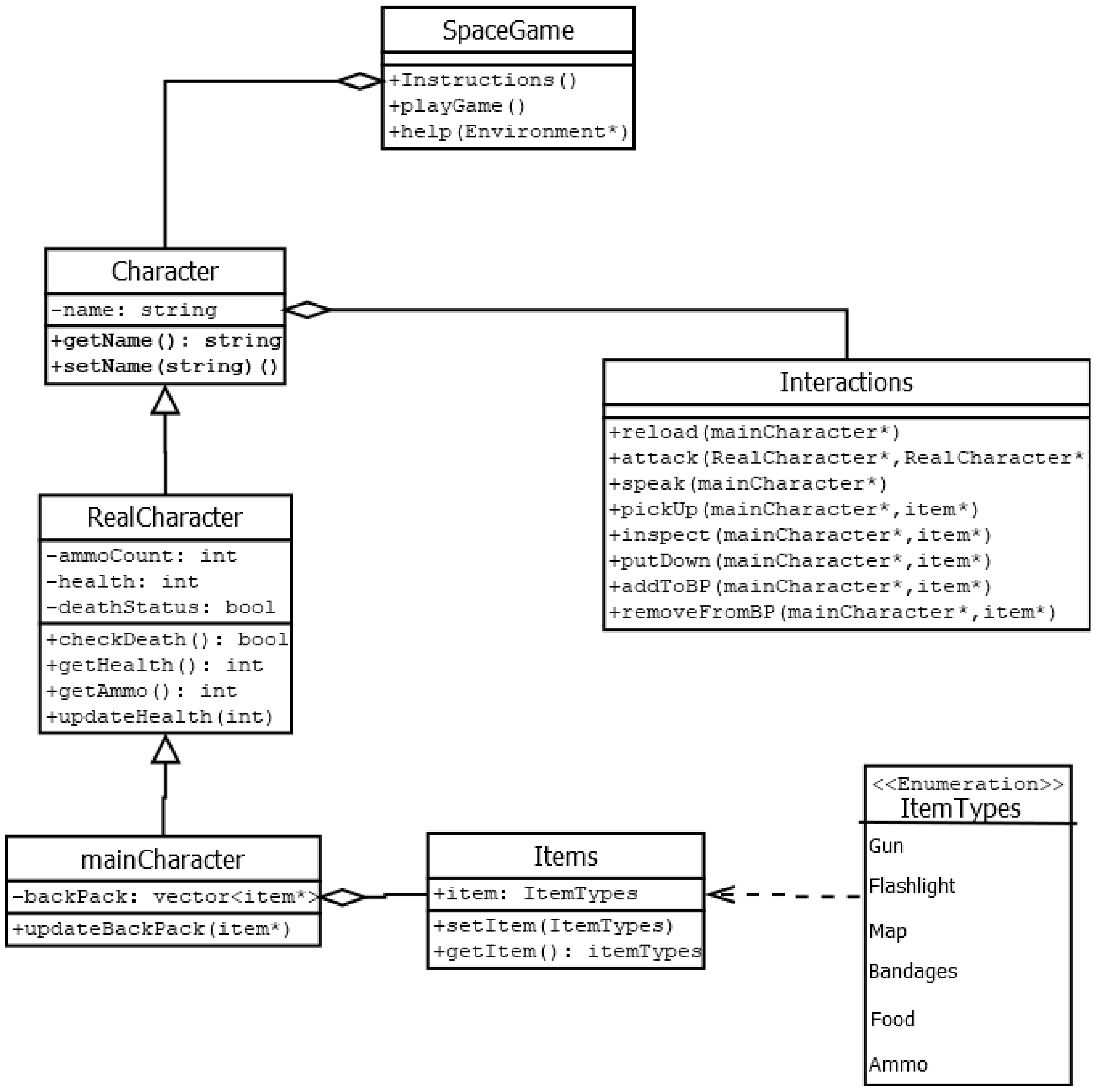
<u>New design</u>



2. Environment classes

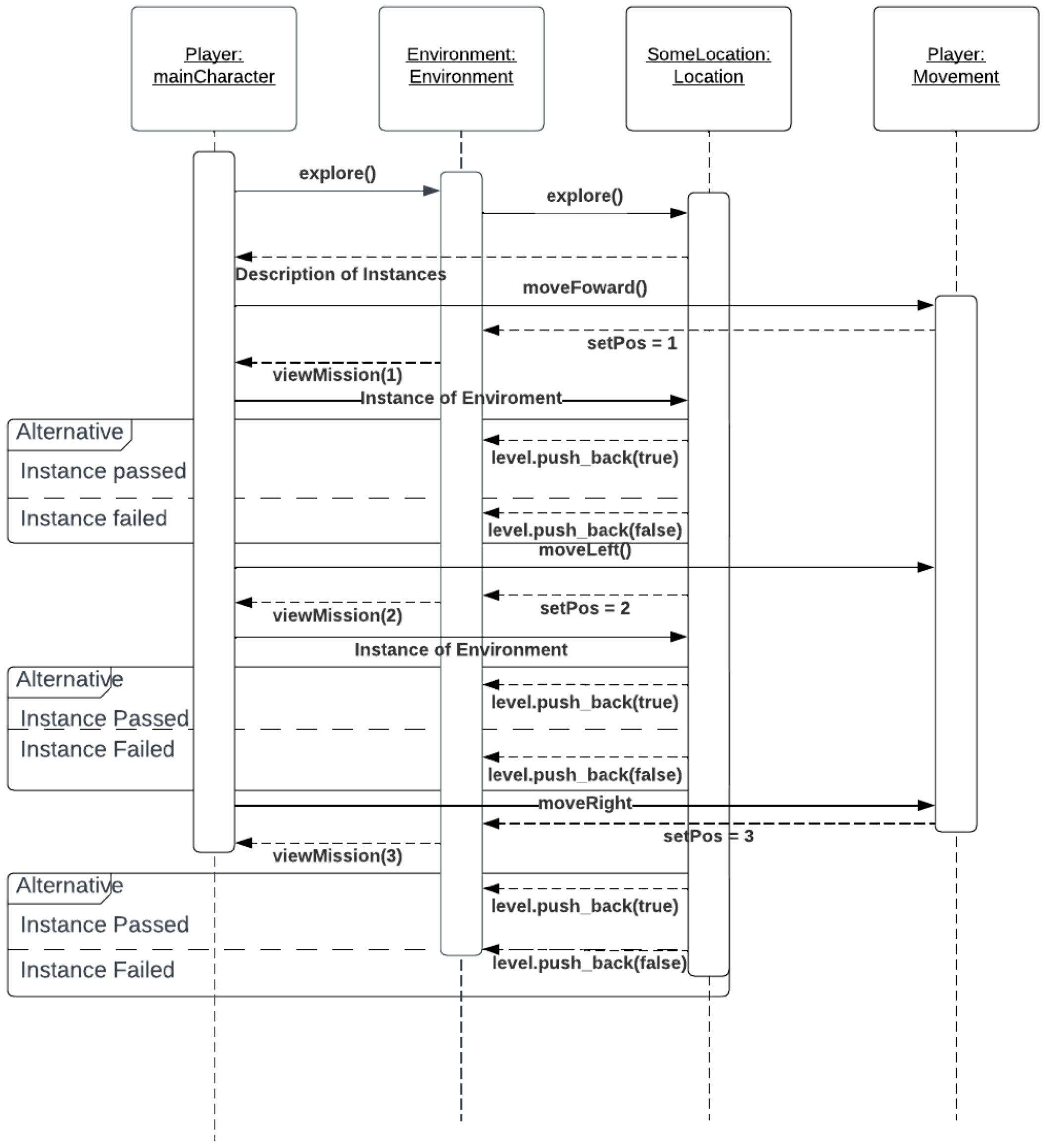


3. mainCharacter class and its related classes

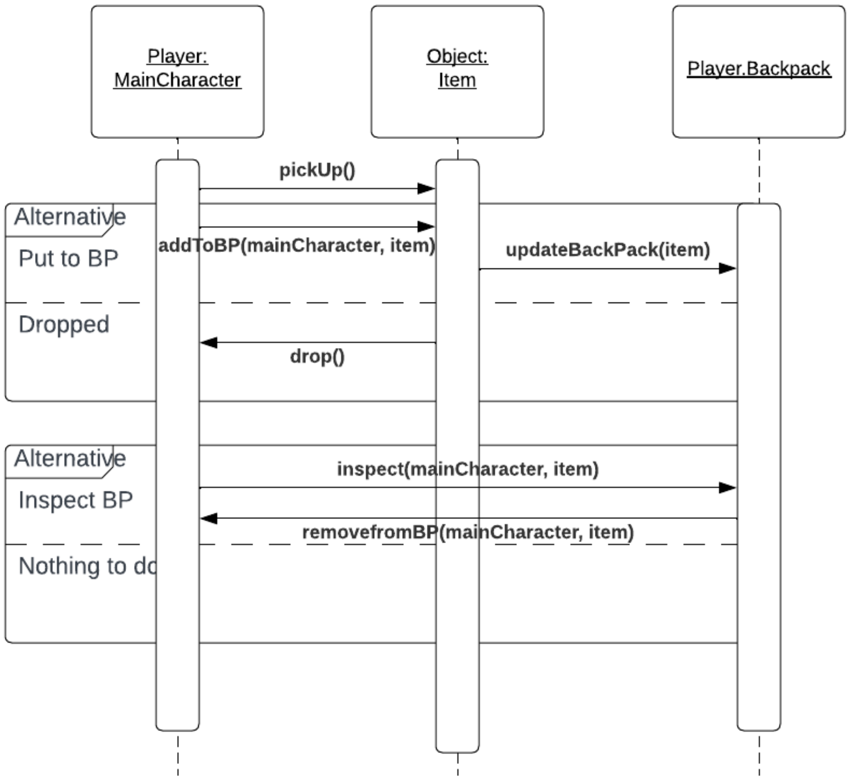


<u>SEQUENCE DIAGRAMS:</u>

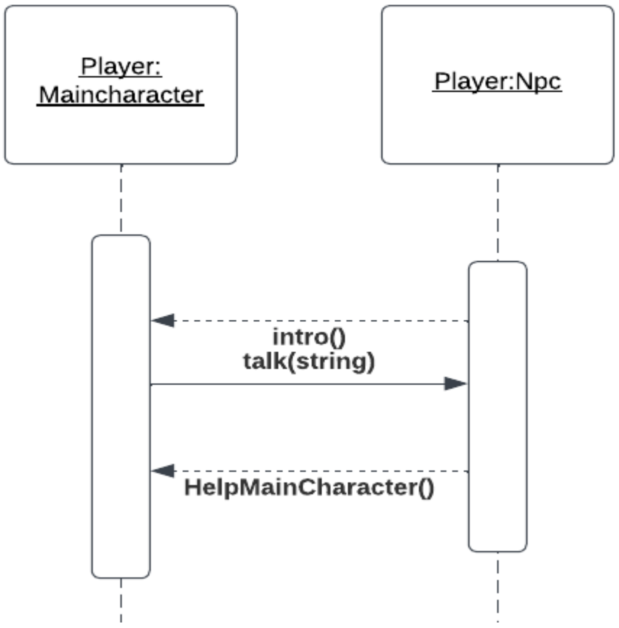
1. Player interacting with environment



2. Player interacting with Object



3. Player Interacting with NPC



Class Descriptions

Class:

1. Character- Implements the common features between all of the subsequent sub-classes including the name and the getName()(gets the name of character), setName()(sets name of character) methods.

- **a. RealCharacter-** This class represents the characters that actually do things, for example, they can die, deathStatus and checkDeath() work together to describe whether a character has died or not. They have health through the use of health variables, getHealth() which returns the health, and updateHealth(), which updates the health of the RealCharacter. They also have ammo, and a getAmmo() method which returns the amount of ammo they have.
  - **i. mainCharacter-** This class inherits all the functionality of the RealCharacter, but also provides the use of a backPack, which is a vector of item*. This backpack can be updated by updateBackPack(item*).
    1. In the actual design, we decided a BackPack class should implement the functionality of a backpack, and not this class, so a Backpack class was created.
  - **ii. Villain-** This class also inherits everything from real character, but also has villainMessage(), something they will say when fought.
    1. A Villian class was never created. A RealCharacter implements all the functionality a villain needs.
- **b. NPC-** This class serves as an interface for two methods, each of which will be implemented in the respective subclasses. The methods are intro(), which introduces the NPC, and helpMainCharacter(), which does what you think.
  - **i. Crew Member**- Provides an implementation of the two methods mentioned above.
  - **ii. Robot helper-** Provides an implementation of the two methods mentioned above, but the reason for interface is to allow variation.

2. Movement- This class directly associates with the Environment class and has functions for the movement of the main character.

1. The movement examples include: moveLeft(), moveRight(), moveForward(), moveBackward(), and moveEnvironment();
2. A string is passed into each one of these methods, as well as a pointer to the respective environment. This is because each environment is represented as a 3 int grid(all environments make up a 3x3 grid), so moving left, right, and forward just means a change in the integer relative to the current position. Just a environment* is passed into each one of these methods except for moveEnvironment(), which actually take a environment*& as well.

3. Items- Items are declared using setItem(ItemTypes). ItemTypes is an enumeration of all the possible Items in the game, so an item can only be an ItemType. A method getItem(), returns the respective item type.

4. Interactions- this class is responsible for all the actions that RealCharacters can perform. The methods of the class include the following:

1. reload(MainCharacter*) - allows the mainCharacter to reload weapon
2. attack(RealCharacter*, RealCharacter*) - allows for an attack between two characters
3. speak(MainCharacter*) - allows the mainCharacter to speak
4. pickUp(mainCharacter*,item*)- allows the mainCharacter to pick up an item
5. inspect(mainCharacter*,item*)- Once an item has been picked up, you can inspect it.
6. putDown(mainCharacter*,item*)-allows mainCharacter to put down an item
7. addToBP(mainCharacter*,item*) - Once an item has been picked up, a main character can choose to add to BP
8. removeFromBP(mainCharacter*,item*)-Once an item is in the BP, it can be removed, if wanted

Up to this point, only three of these methods have been implemented. We plan on implementing the rest during the final phase.

5. Environments- The Environments class will represent the different places the main character can travel. In this base class, each environment will have a name, along with a method to getName() and setName(). Each environment will have completion monitored by a boolean, and the setCompletion(vector<boolean>) tracks this by going through each instance(location) of the environment. As mentioned in the movement class, each instance of every Environment can be represented by an int, and moved between. This is called the position, and has a getPos() and setPos() method. viewMission(int) can view the instance, or mission, of the subclasses. Explore() list all possible instances.
This class undertook the greatest redesign. It now contains an environmentCompletion bool, the tells the user if all the levels of the respective environment are completed. This is done through an array of booleans. Each location in the environment represents a bool, so after the user completes a location, the bool in the array corresponding to that position is set to true. After all the positions are complete, the environment is complete.

a. **MainSpaceShip-** has guessGame(), sandStorm(), and findKey() as missions, or instances. Also has implementations for viewMission(int) and explore().
b. **SpaceColony-** has asteroidAttack(), mathPuzzle(), and alienAttack() as missions, or instances. Also has implementations for viewMission(int) and explore().
c. **Mountain-** has climbMountain(), spiderAttack(), and puzzle() as missions, or instances. Also has implementations for viewMission(int) and explore().