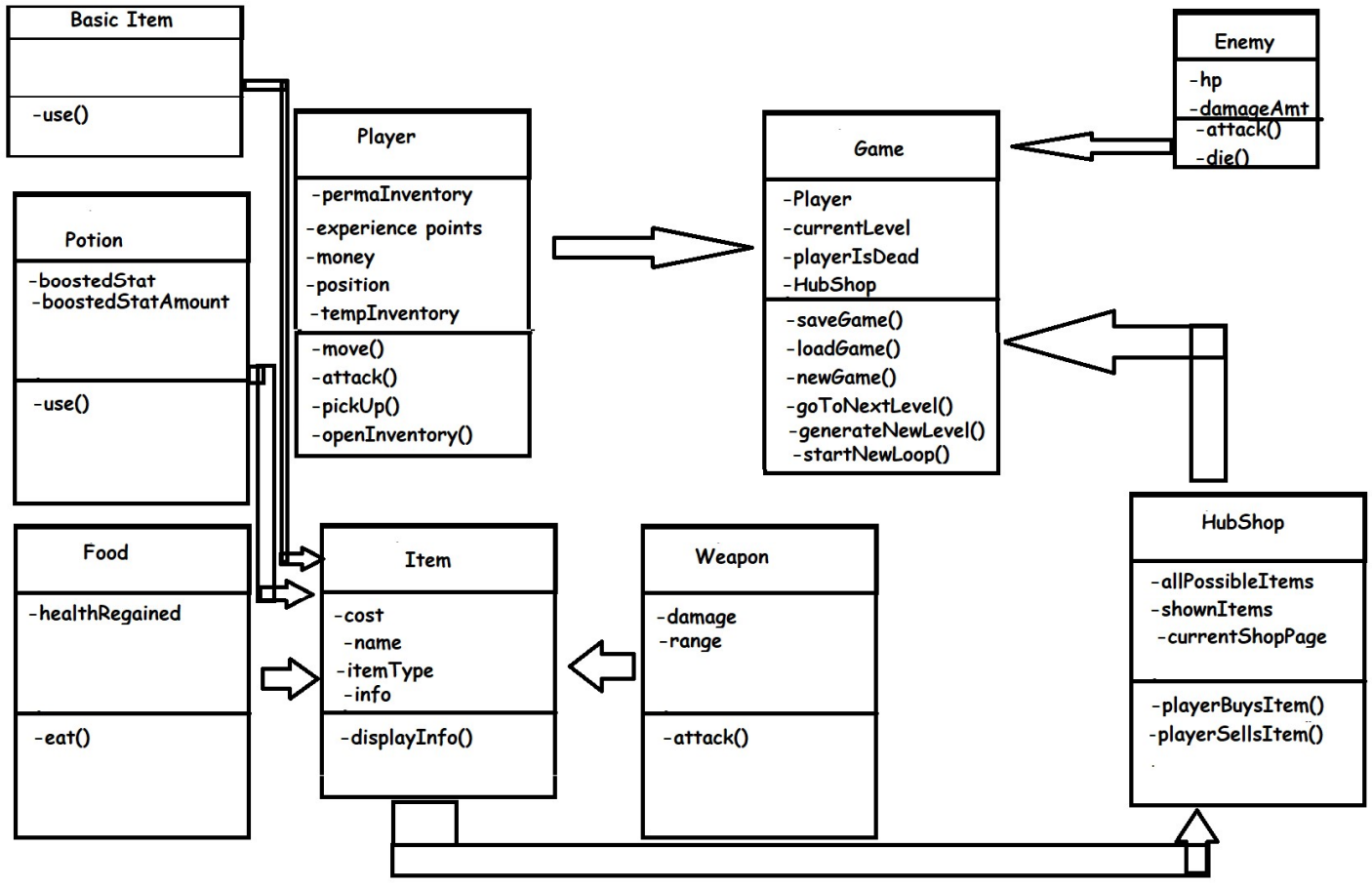
Emily Perry, CSCI 200 Section A

Rogue-Like Adventure Game

Problem Description:

This program will allow the user to play a rogue-like game where each level is randomly generated. There will be items such as weapons, potions, food, and keys that the player can find and use to advance in the game. Upon the player's death, the player will return to a "hub" screen where they get rewarded based on how far they got in the game. In this hub, the player can buy upgrades and items to help them get farther, faster in the game. There will be a preset final level after a certain number of rounds that will be the end of the game, but it will be unlikely that the player gets there on their first few attempts. The game can save and be continued through file I/O so it does not have to be done in one sitting, as it might take a while to finish. The game will have a primitive display that allows the user to see their environment, inventory, and stats.

Data Description:



-
- Lists will be used in this program in the form of vectors to keep track of the player's inventory.
- Files will be used to keep track of the player's stats, inventory, and progress in between sessions so that the game does not have to be played in one sitting. As such when the player selects the "load from file" option upon startup, the program will be able to read important data from the selected file. And, at the end of a session, there will be a save and quit option that writes all the important data to file.

Procedural Description:

```
/*
Game class{

public interface:

saveGame();
loadGame();
newGame();
goToNextLevel();
generateNewLevel();
startNewLoop();

private interface:

Player object
currentLevel data string
playerIsDead bool
HubShop Object
}

writes the player's permaInventory, XP and money to a file
saveGame()

reads in from a saved game file to restore the player's permaInvetory, XP and money
loadGame()

creates a new file to save to
newGame()

resets the player's position and calls generateNewLevel to simulate level advancement
goToNextLevel()

uses rand to create new rooms, items, and enemies
generateNewLevel()

exit the hub and calls goToNextLevel
startNewLoop()
```

```
*/

/*
Player class{

public interface:
move();
attack();
pickUp();
openInventory();

private interface:
vector of player's permanent inventory
int XP
int money
array of positon X and Y
vector of player's temp inventory
}

changes the player's position in the direction they indicate
move()

uses equipped weapon
attack()

picks up the item in front of them
pickUp()

displays all items in the player's inventory
openInventory()
*/

/*
class HubShop{

public interface:
playerBuysItem();
```

```
playerSellsItem();

private interface:
array of allPossibleItems
vector of shownItems
int currentShopPage
}

subtracts the cost of an item from the player's inventory and adds it to their permaInventory and takes it from shownItems
playerBuysItem()

adds the cost of an item to the player's inventory and adds it to shownItems and subtracts it from their permaInvetory
playerSellsItem()
*/

/*
class Enemy{
public interface:
attack();
die();

private interface:
int hp
int damageAmt which is how much damage they do upon each attack
}

deals damageAmt to player
attack()

deletes enemy from map and ends encounter
die()
*/

/*
class Item{
public interface:
displayInfo();
```

```
private interface:
int cost
string name
itemType which is what class of item it inherits
string info
}

prints out the information of the item
displayInfo()
*/

/*
class Food{
  public interface:
  eat();

  private interface:
  int healthRegained
}

gives the player back HP equal to healthRegained and deletes from inventory
eat();
*/

/*
class Potion{
public interface:
use();

private interface:
boostedStat string
boostedStatAmount int
}

adds the stat boost to the player and deletes from inventory
use()
```

```
*/

/*
class Weapon{
  public interface:
  attack();

  private interface:
  int damage
  int range
}

deals damage equal to damage to enemy if they are within range
attack()
*/

/*
class BasicItem{
  private interface:
  use();

  public interface:
  null
}
*/
```

Special Needs/Concerns:

- I don’t think this will be particularly difficult, maybe just figuring out how I display the map and environment and inventory n such. I have experience in making larger projects like this too and I enjoy coding so I believe I will be able to complete it.
- No, I do not plan to use any third party libraries