Advanced Programming Concept Project Report

(Zork-style text based game)

Student Name: Weiyi Wang, Siyan Luo, Weilun Chen

1. Background

Zork-style text-based game is a type of interactive fiction game which relies on text-based input and output to immerse the player in a virtual world. To practice the theoretical knowledge learned from Advanced Programming Concept, we have designed a simplified Zork-style adventure game in Python. This project is made up of mainly two sections, which are the game (UML) design and code implementation.

2. Game design

2.1.Flowchart design

Our Zork-style text-based game places a strong emphasis on battling. The game consists of three different stages, each requires the player to beat enemies of varying strengths in order to gain experience, money, and items (weapons, shields, potions). These resources can be used to level up the player's characteristics and equip himself. The ultimate goal of the game is to defeat the big boss in the final stage, which makes the player win the game.

The design of this game is shown as follows. To help organize the game's structure and plan out its various paths and decision points, we have created a flowchart as Fig.1, which shows the challenges that the player will encounter, as well as the actions they can take and the various paths they can follow.

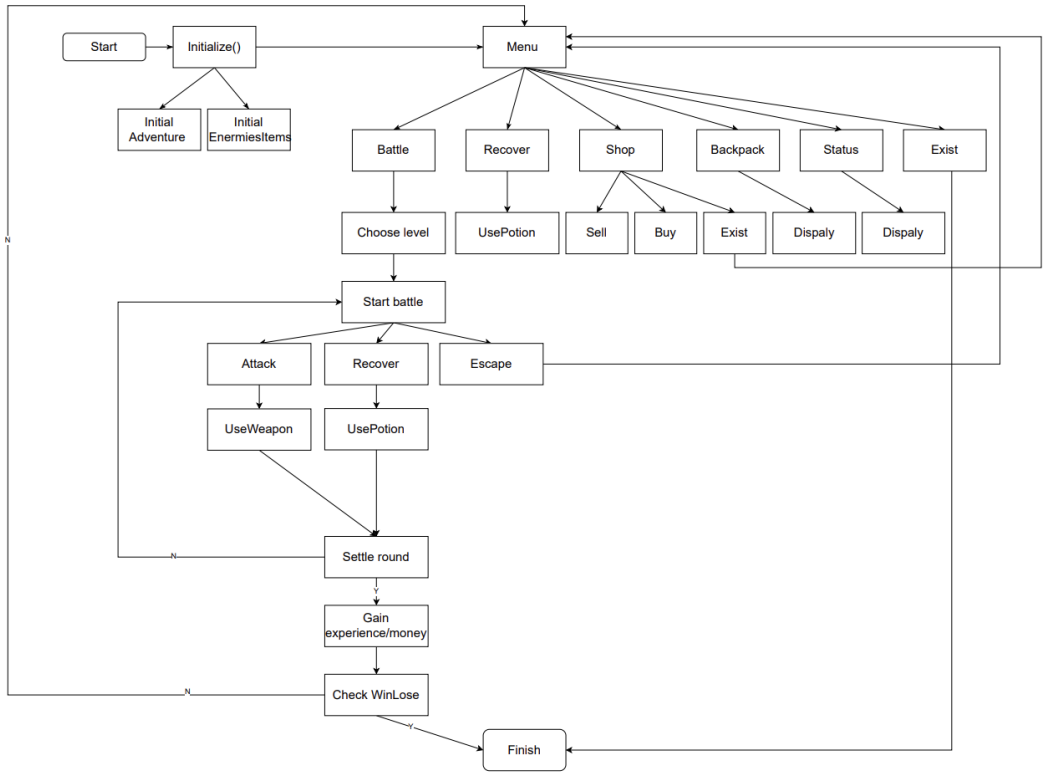

Fig.1 Workflow

When the player starts the game, the adventurer, enemies, and items will be initialized. The player will then be introduced to a main menu where they can choose from six different operations. These operations include "battle," "recover," "shop," "bag," "status," and "exist."

In the "battle" operation, the player will select a battle among the ones they have unlocked. During the battle, the player can choose to attack, recover, or run away. If the player successfully defeats the enemy, they will gain experience and items, then return to the main menu. However, if the player loses the battle, the game is over.

The "recover" operation allows the player to use hp and energy potions to recover their health power and energy levels.

In the "shop" operation, the player can purchase items such as weapons, shields, and potions. They can also sell their items to the shop to gain money.

The "bag" operation displays all the items currently in the player's possession. There is a maximum load of the bag, the player has to be cautious of that every time he makes the decision to add and drop an item.

The "status" operation displays the player's current status, such as their health power, energy, and level.

Finally, the "exist" operation will end the game.

2.2.UML design

In order to visualize and organize our idea before implementation, we have designed the UML diagram as Fig.2.

This project includes 6 main classes: "Game," "Adventurer," "Monster," "Item,"

"Shop," and "Main." Additionally, there are two subclasses of the "Monster" class and four subclasses of the "Item" class, which inherit attributes from their parent classes. Each class has its own set of attributes and functions, all of which contribute to the game's overall flow.

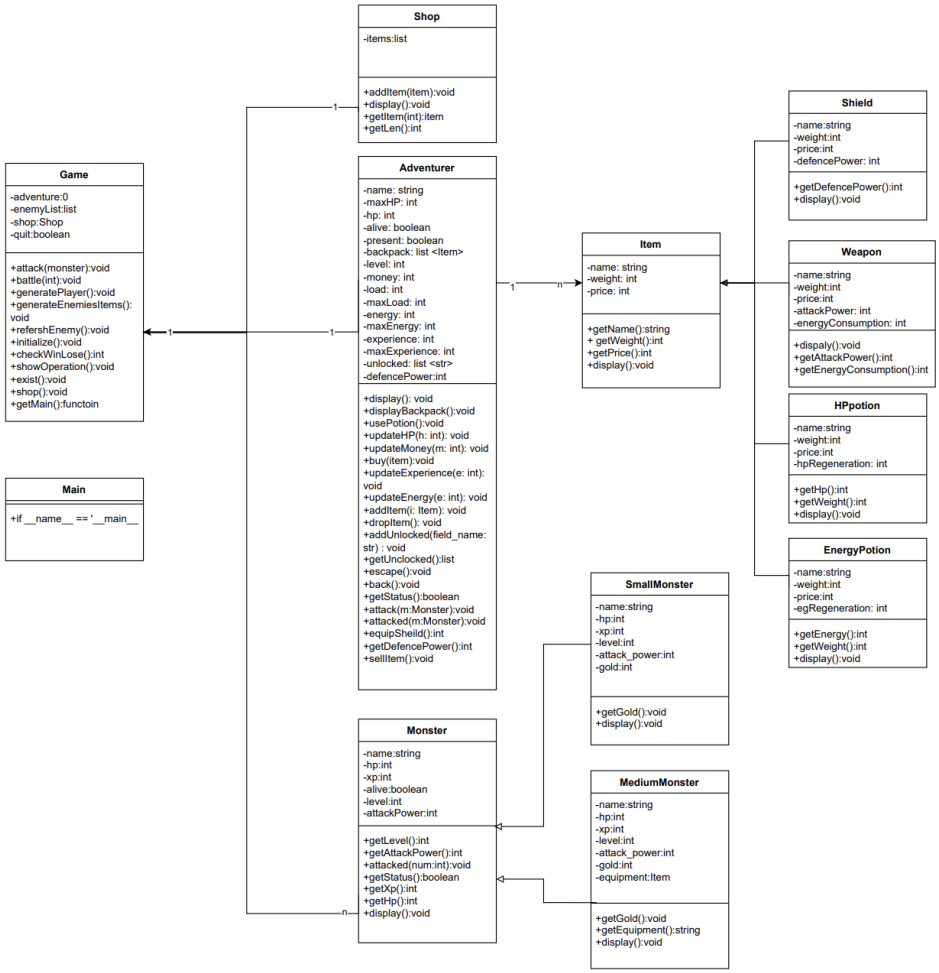

Fig.2 UML

3. Code Implementation Progress

We finished the code implementation by realizing all the functions covered in above UML diagram, using object-oriented programming principles, and creating multiple classes to represent different entities and functionalities in the game. For example, the game has classes such as Adventurer, Monster, Item, Game, Shop, and Main.

Besides, we ensured that more than one class was instantiated multiple times to represent multiple instances of a specific entity. For instance, the Monster class was instantiated multiple times to represent different enemies with varying strengths and characteristics. Similarly, the Item class was instantiated multiple times to represent different types of weapons, shields, and potions.

In terms of utilizing inheritance to establish base classes and derive subclasses with specific functionalities, and improve code reusability and promote a hierarchical

structure, we created different subclasses of enemies for different challenge levels that contain common attributes and methods from the Monster class. The Weapon, Shield and Potion classes then inherit from the Item class, inheriting the base attributes, like weight, price and so on, and also some common methods while also adding their unique characteristics, for example, Weapon class has attack power attribute, Shield class has defense power attribute.

Finally, after completing all the functions, we optimized the code and rationally used filter, map, list compensation to make the code more concise. For example: In battel function, we use lambda to find an enemy object from the enemy list self._enemyList that satisfies the condition that its level is equal to gameStage. Using next and filter to implement if an eligible enemy is found, assign it to the variable m, otherwise assign the default Monster object to m. As Fig.3 shows blue block is our original code, red block is after optimization.

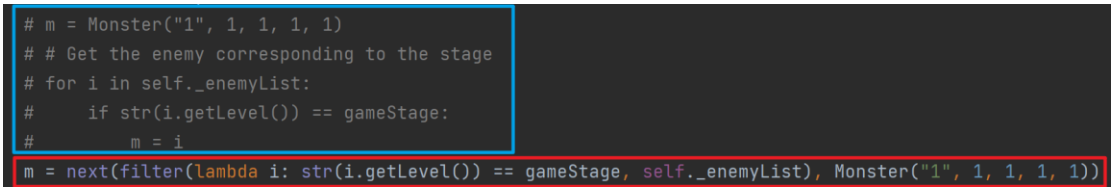

Fig.3 Code Optimization

4. User Manual

4.1 Game Installation

As our game is design to be text based, the game running environment is entirely on the console of PyCharm, alternatively, in order to improve the user-friendliness, we have also converted the py. document into exe. document, which means the users do not have to install any integrated development environment for running this game.

4.2 Game Introduction

When opening the application file, the user can read a brief introduction of the text-based game before starting it, as marked with red box in Fig.4.

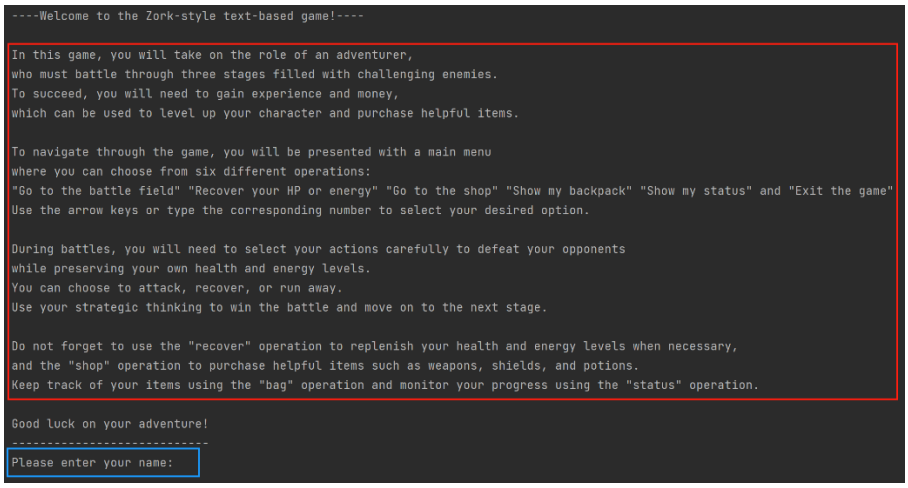

Fig.4 Display of Game Introduction

4.3 Game Operation

At the beginning of each game, the user is required to input the adventure name as marked with blue box in Fig.1. Upon starting the game, the player will be presented with the main menu as Fig.5. The main menu provides various options to navigate the game, for example, the player can input the index “1” of “Go to the battle field”, to make the adventurer start their challenge.

```
Welcome to the main menu! Here you can choose one of the options below:
---------------MENU---------------
1. Go to the battle field.
2. Recover your HP or energy.
3. Go to the shop.
4. Show my backpack.
5. Show my status.
6. Exit the game.
------------------------------------
Which option would you like to choose:
```

Fig.5 Display of Menu Operations

In the " Go to the battle field" operation, the player can choose a battle from the unlocked options as Fig.6.

```
Battle fields you have unlocked:
['1.1']

Please enter the level you want to challenge:
```

Fig.6 Display of Battle Field Level to Choose

During battles, the player has the following choices as Fig.7:

Attack: Engage in combat and attempt to defeat the enemy.

Recover: Use hp and energy potions to restore the adventurer’s health power and energy levels.

Run Away: Attempt to flee from the battle. Note that this may not always be successful.

```
Hello adventurer, here is the enemy you are going to challenge:
-----------------------------------------------------------------
Monster name: Goblin
Health points: 50
Carried experience: 200
Attack power: 10
Carried gold: 100
-----------------------------------------------------------------
Which action would you like to take: 1. Attack; 2. Recover; 3. Escape.
Please enter your option:
```

Fig.7 Display of Options During a Battle

The "Recover" operation allows the adventurer to use hp and energy potions to replenish the adventurer’s health power and energy levels. Use this option strategically to restore the adventurer’s resources during the game.

Visit the shop to purchase weapons, shields, and potions. The adventurer can also sell the their items to the shop and earn money as Fig.8. Spend the adventurer’s resources wisely and choose items that will enhance the adventurer’s chances of success in battles.

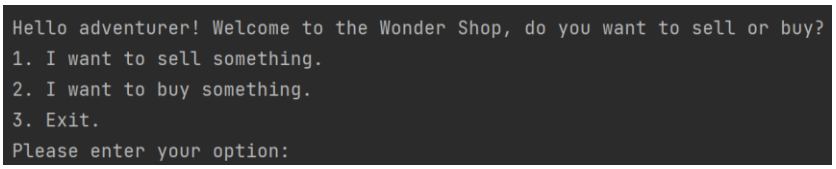

Fig.8 Display of Shop for Buying/Selling Items

The "Show my backpack" operation displays all the items currently in the adventurer's possession as Fig.9. Be mindful that there is a maximum load for the adventurer's backpack. Consider the weight of items before adding or dropping them to ensure the adventurer has the necessary resources at all times.

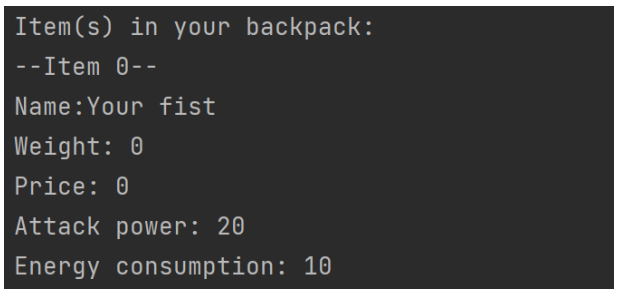

Fig.9 Display of Items in the Backpack

The "status" operation provides an overview of the adventurer's current character status as Fig.10. It includes information such as the adventurer's health power, energy, and level. Stay updated on the adventurer's progress and make informed decisions based on the adventurer's status.

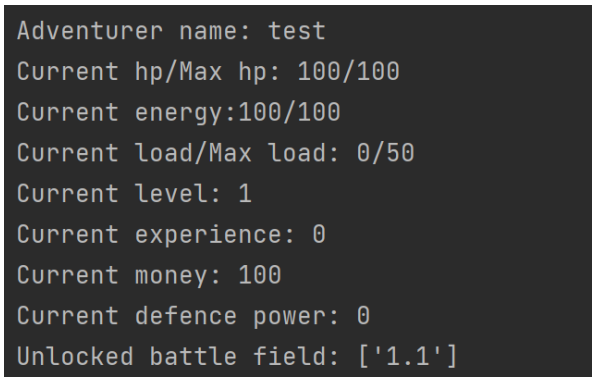

Fig.10 Display of Adventurer Status

The "exist" operation allows the player to exit the game. Use this option when the player wants to stop playing the game.

4.4 Tips and Strategies

Plan the adventurer's battles carefully, considering the strength of the enemies and the resources at the adventurer's disposal.

Use experience points to level up the adventurer's character and improve their abilities.

Manage the adventurer's resources efficiently, balancing the adventurer's inventory and available funds.

Strategize and make smart decisions to overcome challenges and progress in the game.

Appendix

Appendix 1. Packages

In this project, we referenced external packages and internal classes The following are detailed names:

Type	Name
Package	Random
Class	main Game Adventurer Monster SmallMonster MediumMonster Shop Item HPpotion EnergyPotion Shield Weapon

Appendix 2. Workload Division

UML Design	For the initial design of UML, both of us participated in and made lots of modifications to decide on the final version.
Code Implementation	Each of us mainly contribute for following class. Weiyi: main, Game, Monster, SmallMonster, MediumMonster Siyan: main, Game, Adventurer, Monster, Shop Weilun: main, Game, Shop, Item, HPpotion, EnergyPotion, Shield, Weapon The integration of code was done by Siyan. For the testing, debugging and optimization we worked together.
Documentations	Siyan wrote the code documentation. Weiyi and Weilun wrote the report.