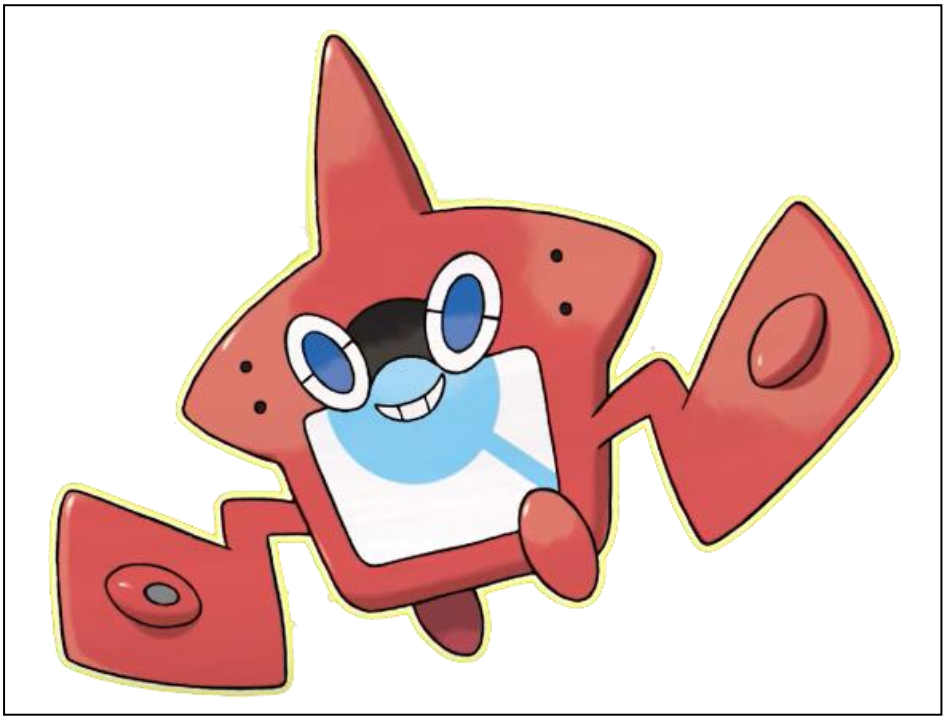
2021



OOP Major Practical

PROJECT SPECIFICATION

XUHUI WANG

RENYUAN FEI

YUANCHEN XIANG

Contents

Project Specification - Major Practical

Introduction

This project implements a custom Pokémon battle system for Pokémon fans. The idea is from the Pokémon games. The system allows players to customize Pokémon: freely choose Pokémon's types and names and add Pokémon to PokéDex. This system also includes the Pokémon Battle functionality, players need to choose the Pokémon with the suitable type to fight against enemies with different types. After customizing the Pokémon and adding it to PokéDex, the relevant information of the Pokémon will be automatically stored in the player's local area and allowed to be used when the player enters the game next time.

Design Description

Assessment Concepts

Memory allocation from stack and the heap

Arrays:

We will use serial dynamic arrays using new to add in some initial Pokémon and subject entries and allow more Pokémon to be added.

Strings:

Strings are used extensively through this major practical, such as Pokémon names, types, and skills.

Objects:

PokeDex, Grass, Water, Fire.

User Input and Output

• I/O of different data types:

Command-Line. Users are required to choose to add a Pokémon when starting the game and enter the Pokémon's serial number-this will determine the locations that the Pokémon will be stored in PokeDex and select the type and name according to the prompts.

The serial number entered must be an integer, and if a new Pokémon is added after entering the same serial number as the previous operation, the old Pokémon will be released.

You can view existing Pokémon through the console.

After entering the battle, the Pokémon held by the player has an attribute restraint relationship with the Wild Pokémon, which will affect the damage caused.

Object-oriented programming and design

• Inheritance:

There are different types of Pokémon. When start a battle, it will be clearly reflected in the damage caused to wild Pokémon. For example, when the wild Pokémon is of the grass type, if the player's Pokémon is of the fire type, it will cause a higher damage value; but if the player's Pokémon is of the water type, it can only cause minor damage.

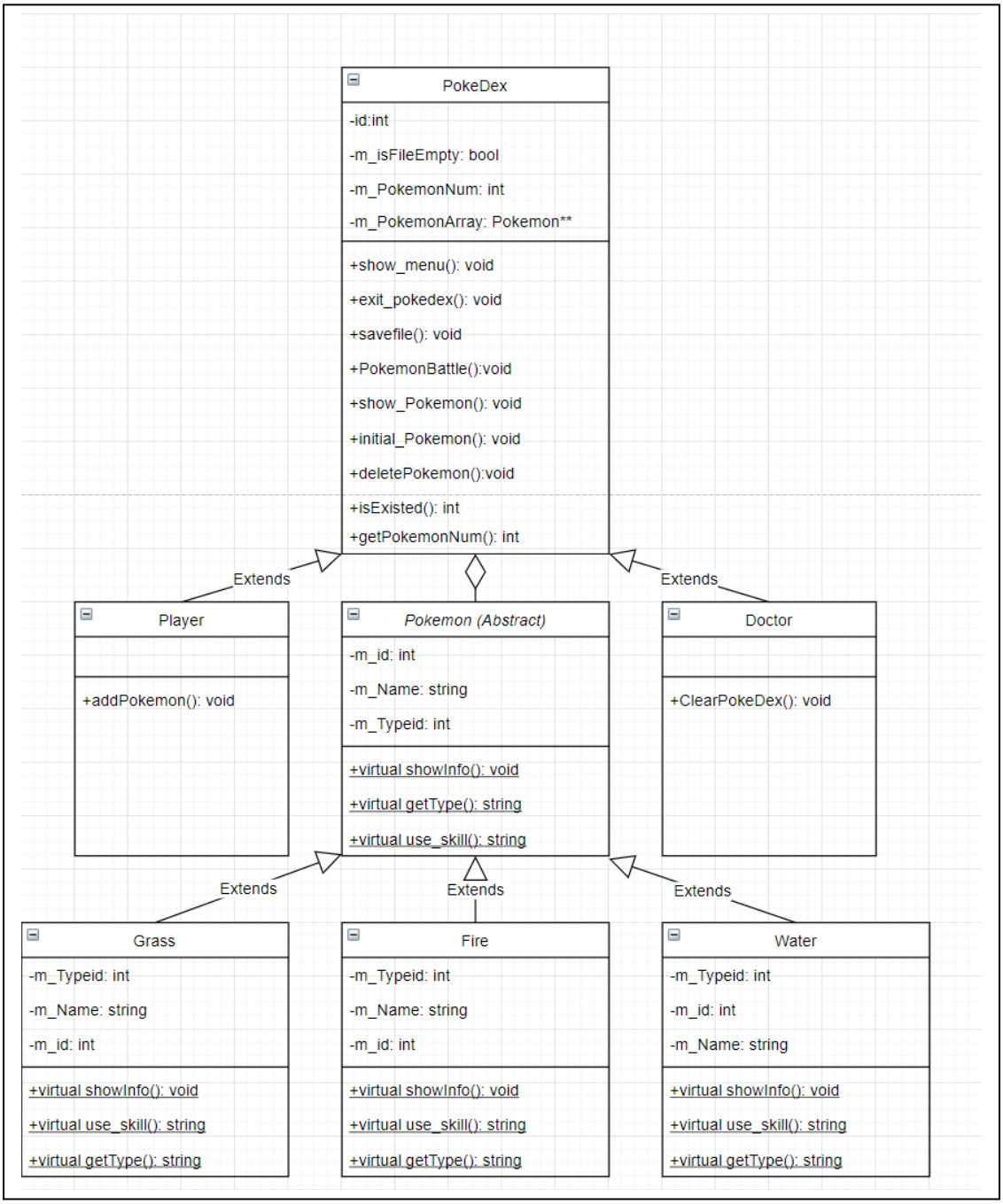
•Polymorphism:

Both Doctor and Player are the sub-classes of PokeDex, they all extended the basic functionalities from the parent class, PokeDex. However, the Polymorphism is shown by the Player can be called to battle with wild Pokémon, Doctor cannot; Doctor can clear the PokeDex- delete all the Pokémon in the PokeDex- for Player, the Player do not have the permission with this behaviour.

•Abstract Class:

Through this system, Doctor and Player plays different roles with different functionalities, which has to be purely isolated with based classes which are the abstract classes, interfaces.

Class Diagram



Class Descriptions

PokeDex

PokeDex itself integrates a large number of functions.

Based on this category, the player can view the main menu and enter the corresponding serial number to perform the corresponding function. This includes exiting PokeDex, joining Pokémon battles, adding new Pokémon, and viewing the current number and specific information of Pokémon in PokeDex. Unless the player chooses to delete the corresponding Pokémon or clear all Pokémon in PokeDex, even if the player quits the game, the existing Pokémon will be saved to the player's local area in the form of a txt file and can continue to be used when the player back the game.

When inputting invalid content, such as decimals, negative numbers, character strings, and the sequence number range of the executed behaviour, the system will report an error and ask the player to re-enter it.

Player

Players can perform most of the operations in PokeDex, including operations such as adding/deleting/naming/selecting types, and can also choose to play against wild Pokémon. However, the player does not have the permission to clear all Pokémon in PokeDex.

Doctor

Doctor cannot add new Pokémon, nor can he play Pokémon battles. However, the Doctor can clear all Pokémon in PokeDex, perhaps because the player has died.

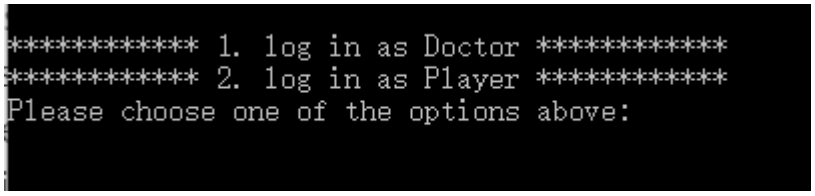
Pokémon

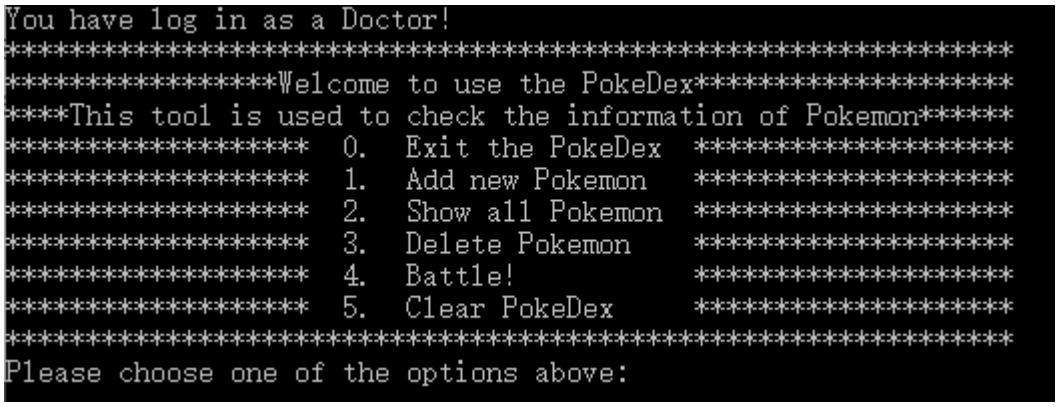
Pokémon is an abstract type, including the types of fire, water and grass Pokémon, including setting the name and choosing the type for new Pokémon, and using skills.

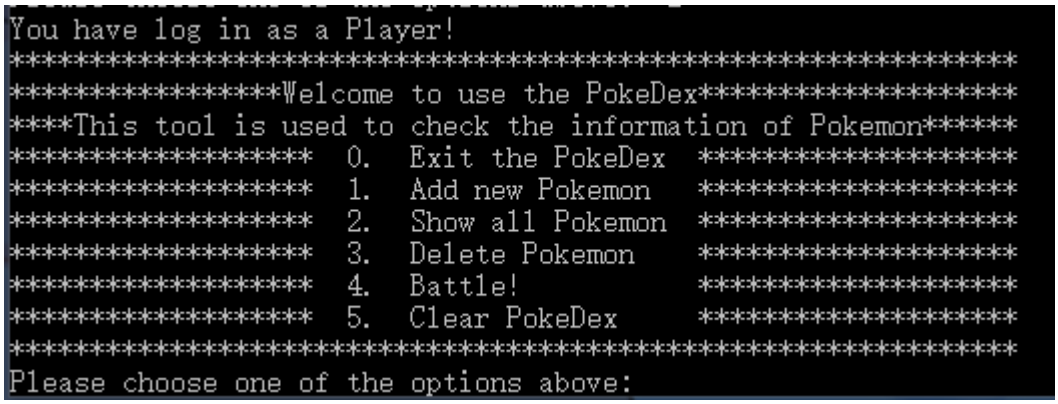
Grass/ Fire/ Water

When the user makes a choice among the three types, the corresponding class will be called. These three types of Pokémon have a corresponding restraint relationship in battle. When fighting with restrained Pokémon, they can cause higher damage, and vice versa. The user needs to name it when adding a new Pokémon.

User Interface







User Interface are presented with a list of options to choose from command-line. The user can choose to log in as the player / doctor by entering 1/2, then there are more sub-options after the user log into the system. After entering the option, the user used to type Enter button on the keyboard to continue to the next step.

Code Style

Since the code is completed using the Windows version of Sublime as the function of the text editor, with the help of the automatic line-wrapping function of the text editor, the code appears in the shape of "Wave", and in each CPP file There are blank lines between functions. It not only increases the readability of the code, it also makes it easier to find problems during the Debugging process later.

Testing Plan

Our unit test will cover all the functions in the illustration. Each function has a test file, such as addPokemontest.cpp. We will use input.txt to test the input of the file to ensure that the input value is limited, and no error will be reported. After that, we will conduct integration tests. We will match the output to know whether the output function is complete and whether the desired output is obtained. For all tests, we will use Google Test (Unit Testing) and using Makefile to automate the process. In addition, each time the changes made, it will keep up with the pace of regression testing.

Google Testing (Unit Testing)

1. Regarding the function with only printing function like showmenu, we will use EXPECT_EQ (val1, val2) to compare the output of the function with the expected output to judge whether the output is normal.

2. Regarding the addPokemon function that inputs parameters after calling the function, we try to use Google test, and input the parameters in the input.txt file by editing the SH file, and judge whether the test is passed by whether it will report an error

Preparation plan (due to the high difficulty of writing SH file, we are not sure whether it can be written successfully. So, we prepared the second set of test plan, write an independent test, add the run command in the makefile, and add input.txt, passed Judge whether there is an error to check whether the test passes)

3. Regarding the show_Pokemon function that automatically reads and prints data, we will pre-set several Pokémon objects inside the test, and let the function read these Pokémon, and then use EXPECT_EQ (val1, val2) to compare the printed content and Expected output to determine whether the function can be read and printed normally.

Regarding google test, it is currently not possible to directly install third-party library files into the system on cs50. We now have two options.

1. Write a link to refer to google test in the makefile, but do not install it.

(Due to the current limited knowledge, the degree of difficulty is relatively high)

2. Replace other test frameworks (such as the catch2 test framework that comes with C++).

Main Function Test

1. Start program, enter 1 —> login Doctor
2. Start program, enter 2 —> login Player
3. enter 0 —> Exit the PokeDex and end the program
4. enter 1 —> (Player)Add new Pokémon

   (Doctor) Prompt no permission
5. enter 2 —> Print the ID, name and type ID of all Pokémon
6. enter 3 —> Delete the specified Pokémon
7. enter 4 —> (Player)Play against wild Pokémon

   (Doctor) Prompt no permission
8. enter 5 —> (Player) Prompt no permission

   (Doctor) Clear Delete Pokémon

Schedule Plan

Stretch Goals

We plan to design a PokéDex system that PokéDex system can store Pokémon information. You can use PokéDex system to add Pokémon information, delete Pokémon information, show all the Pokémon information you have, and clear all the Pokémon information. You also can battle with other Pokémon to see your different Pokémon have different skills. Last but not least, you can choose to exit the system.

Features Checklist

1. Log in as Doctor / Player
2. Exit
3. Add / initialize / name / check Pokémon
4. Save Pokémon
5. Pokémon Battle with wild Pokémon
6. Different Types of Pokémon (with different damage)
7. Delete Pokémon (by Player)
8. Clear PokéDex (by Doctor)
9. Pokémon reuse (save data in local device)
10. User Interface

Week 8 - Preparing for assessment in Week 9

- Choose the topic
- Share the ideas
- Pseudo code sharing
- Make the group and add all members in
- Set up the Coding environment
- Prepare for the makefile document

Week 9

- Create SVN structure
- Make the Testing Plan
- Head files and classes: PokeDex, Pokemon (Abstract), Grass/Water/Fire
- Finish makefile (the first version)
- Finish coding (the first version)
- Make Test Plan

Week 10

- Input Validation
- Testing (Google Test, main function Test, automated test)
- Test cases
- Add more classes (e.g., Player, Enemy, wild Pokemon and etc.)
- Improve functionalities
- Makefile for testing

Week 11

- Check the Goals
- Grade
- Improve Command-Line UI design.
- Self-check with the rubric