CMSC 202 Spring 2024

Project 2 – UMBC Pokémon

Assignment: Project 2 – UMBC Pokémon

Due Date: Tuesday, March 12th at 8:59pm on GL

Value: 80 points

1. Overview

In this project, you will:

- Practice basic C++ syntax including branching structures,
- Write classes and instantiate those classes using a constructor,
- Use arrays to hold objects,
- Use simple file input,
- Practice breaking projects into multiple files, and
- Using a makefile to compile a project.

2. Background

For this project, we are going to be designing a simple implementation of the famous Pokémon franchise. If you are not familiar with Pokémon, it is a game where you act as a trainer of pocket monsters (Pokémon). You search the world in hopes of finding powerful Pokémon. Additionally, you can train the Pokémon once you have caught them. Finally, after you have found some Pokémon, you can battle them against other people.

There are two pieces of information that are important in this game. The first is the list of all of the Pokémon that are available. Currently, there are 151 Pokémon available in the **`proj2_pokeDex.txt`** file (which is provided). The information about each of the Pokémon provided includes their number (1-151), their name, their maximum combat points, and their rarity (1-3). ****For Pokémon fans - the numbers used in this project are somewhat random and do not represent actual Pokémon values!**

The second piece of important information is the list of Pokémon that the user has managed to catch. A user has the ability to go out and try and catch a

Pokémon but the more powerful the Pokémon is, the harder they are to catch! Once a user has caught a Pokémon, the information such as name, combat points (CP), and their rarity (1-3) will be tracked.

For our implementation of the game, we will be designing some basic functionality including: listing all available Pokémon, listing your collection of Pokémon, trying to catch Pokémon, training your Pokémon, and finally, battling against other Pokémon.

3. Assignment Description

There are two classes that are used in this project. Take a few minutes to review the header files provided (**`Pokemon.h, and Game.h`**) as well as the project driver (**`proj2.cpp`**). Your job will be to implement the corresponding .cpp files (**`Pokemon.cpp and Game.cpp`**) based on the provided files. The first class, **`Pokemon`**, should be very easy to implement. Just copy each of the functions from the header file and implement them in the .cpp file. Then you can run the make function for just **`Pokemon.o`** which would just be "**`make Pokemon.o`**". This will allow you to implement each of the functions incrementally. After you have run **`Pokemon.o`**, you can start to work on **`Game.cpp`**.

For **`Game.cpp`**, you will need to read in a list of Pokémon and the information related to those monsters from a file and load them into the existing array. The list of Pokemon is static, and you can assume that the size is stored in a constant (**`TOTAL_POKEMON`**). There are a number of constants provided in both **`Pokemon.h`** and **`Game.h`** and you should be using all of them. Make sure to test each of the arrays to make sure they are properly loaded.

Once the file is loaded, you can start to work on the menus and then the display, search, train, and battle functionality.

4. Requirements:

This is a list of the requirements of this application. For this project, you will be provided with header files to start you in the right direction. For you to earn all the points, however, you will need to meet all the defined requirements.

- You must follow the coding standard as defined in the CMSC 202 coding standards (found on Blackboard under course policies). This includes comments as required.
- The project must be turned in on time by the deadline listed above.
- The project must be completed in C++. You may not use any libraries or data structures that we have not learned in class. Libraries we have learned include **`<iostream>`**, **`<fstream>`**, **`<iomanip>`**, **`<vector>`**, **`<cmath>`**, **`<ctime>`**, **`<cstdlib>`**, and **`<string>`**. You may not use vectors – everything must be implemented in arrays. You should only use **`namespace std`**.
- You **must** use the provided files, **`Pokemon.h`**, **`Game.h`**, **`makefile`**, **`proj2_pokeDex.txt`**, and **`proj2.cpp`**, to create the application. Do not add or change any constant, variables, or functions in these files. Do not add member variables to any class.
- To copy all of the starting files, navigate to your project 2 folder and type:
  ```
  cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj2/* .
  ```
- All user input must be validated. For example, if a menu allows for 1, 2, or 3 to be entered and the user enters a 4, it will re-prompt the user. However, the user is expected to always enter the correct data type. i.e. If the user is asked to enter an integer, they will. If they are asked to enter a character, they will. **You do not need to worry about checking for correct data types.**
- There is one input file for this project named, "**`proj2_pokeDex.txt`**". The file name is passed in **`proj2.cpp`**. Look at the code in **`proj2.cpp`** for additional details. The files are already provided in Prof. Dixon's course folder on GL.
- For this project, **pointers are not to be used.**
- Each team can have between 0 and 4 Pokémon. You add Pokémon by searching for them. You can try to find a common (1), uncommon (2), or ultra rare (3) Pokémon.
- Have a main menu that asks if the user wants to:

  1. Display Complete PokeDex

- Displays (using proper widths) all Pokémon loaded into the game from the input file.

2. Display your Team
   - Checks to make sure team size is greater than 0.
   - Displays (using proper widths) all Pokémon that the player has successfully caught and added to their team.
3. Search for a new Pokémon
   - Allows user to choose a rarity of Pokémon to search for (between 1-3). Then identifies if one is found based on this table:
     Common = 45% chance of finding
     Uncommon = 25% chance of finding
     Ultra Rare = 1% chance of finding
   - If found, tries to add new Pokémon to team.
     - First, puts new Pokémon in any open slots.
     - If no open slots, adds to first lower slot with a lower CP.
     - If no open slots and no lower CP, does not add
4. Battle your Pokémon
   - Checks to make sure team size is greater than 0.
   - Randomly chooses a Pokémon from complete pokeDex to battle. Assigns random Pokémon a CP between 0 – maxCP for that Pokémon + 200.
   - Allows user to choose one team member to battle.
   - Compares CP between enemy and team member.
   - If team member wins, tells user and returns to main menu.

- If team member loses, sets CP = 0 and returns to main menu.

5. Train your Pokémon
   - Checks to make sure team size is greater than 0.
   - Allows user to choose a team member to train.
   - Passes the maxCP from the pokeDex for that corresponding Pokemon and calls Train for that team member.
   - Will not allow team member to train above max CP.
6. Exit

5. Recommendations

You must use the provided header files (**`Pokemon.h`** and **`Game.h`**) additionally, we provided you with the **`makefile`**, **`proj2_pokeDex.txt`**, **`proj2_sample1.txt`** and the **`proj2.cpp`**.

Here are some general implementation recommendations (do not follow these verbatim – these are GENERAL suggestions):

- Read each of the header files in detail. Read through the project document in detail. Use paper to take notes. Each header file has tons of comments.
- Design the solution (part is already designed, so make sure you understand how it works) on PAPER.
- Read through the provided **`Pokemon.h`**. Think about how you can use the starting file **`proj2_pokeDex.txt`** to populate an array of Pokémon. **`Train`** is the only tricky function in this class. Even if you don't use all of the functions, please implement all of them.
- Once **`Pokemon.cpp`** is written, start on **`Game.cpp`** – start with loading in the Pokémon from the **`proj2_pokeDex.txt`** file. This shouldn't be too hard because none of the Pokémon have a name that is more than one word (you don't need **`getline`** for this project).
- After the file is successfully loaded (and displayed) work on the main menu.

- The display functions (**`DisplayPokeDex`** and **`DisplayTeam`**) just iterate over those two arrays and display any entries available. If **`DisplayTeam`** is empty, it just indicates this to the player.
- There are several functions that are used with the idea of Searching for a new Pokémon.
    - **`CatchPokemon`** is the main function for trying to catch the Pokémon.
    - **`CatchMenu`** is just the menu itself and returns what the user chooses.
    - **`FoundPokemon`** identifies specifically which random Pokémon of the chosen rarity is found. If the user chooses rarity 1, they can only find a rarity 1 Pokémon. This function also reduces the starting CP of the Pokémon found by 30 – 50%. So if the CP in the pokeDex is 1000 then the Pokémon caught will have a CP between 500 – 700. This is to leave space to train!
    - **`AddPokemon`** attempts to insert the Pokémon into m_team. If the team isn't full, it inserts the new Pokémon into the first available slot. If the team is full, it inserts the new Pokémon into the first slot where the CP of the new Pokémon is higher than an existing team member. If the team is full and the new Pokémon's CP doesn't exceed any of the existing team members, the Pokémon is released.
- **`BattlePokemon()`** is the most difficult single function to write so expect to spend some time finishing it.
- **`TeamSize`** is a helper function that tracks the size of the team by iterating over it and seeing if it has a name or not.
- **`FindPokemon`** is a helper function that is passed the name of a Pokémon and returns the index if that name is found. If a Pokémon with that name is not found, this function returns a -1.

6. Sample Input and Output

The input file for this project includes a data file that has all 151 of the original Pokémon listed. The file, **`proj2_pokeDex.txt`** is the text file used in this project. The data is num, name, maximum CP, and rarity.

You can type "**`make run`**" and it should use a built in macro in the provided makefile.

In the sample output below, user input is colored blue for clarity. After compiling and running proj2, the output would look like this:

```
[jdixon@linux1 proj2]$ make run
./proj2 proj2_pokeDex.txt
Welcome to the UMBC Pokemon
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

If you were to display all the Pokémon, it would look like this (On your program all 151 would be display but we truncated them, so it didn't waste paper):

```
Welcome to the UMBC Pokemon
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
1
     1.    Bulbasaur
     2.      Ivysaur
     3.     Venusaur
     4.   Charmander
```

```
     5.   Charmeleon
**6-149 Omitted for space**
   150.       Mewtwo
   151.          Mew
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

If the user would choose 2, Display your Team, then the output would look like this:

```
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
2
You have no team yet. Maybe search for a Pokemon?!
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

If the user would choose 3, Search for new Pokemon, then 2, Display your Team, the output would look like this:

```
What would you like to do?:
1. Display Complete PokeDex
```

```
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
3
What rarity of Pokemon would you like catch?:
1. Common (High Probability)
2. Uncommon (Normal Probability)
3. Ultra Rare (Extremely Low Probability)
1
You start to search.
You found a Chansey
Chansey added to your team!
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
2
1.      Chansey   371
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

If you were to choose to train your Pokémon, it would look like this:

```
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
```

```
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
5
Which of your pokemon would you like to use?:
1.      Chansey   371
1
Your Chansey's CP goes up!
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
2
1.      Chansey   381
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

If you were to choose to battle your Pokémon, it would look like this:

```
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
4
You are going to fight a Kangaskhan
The enemy has a CP of 698
```

```
Which of your pokemon would you like to use?:
1.      Chansey   381
1
You lost.
Your Chansey can't lift its head.
You should replace it.
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
2
1.      Chansey     0
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

Here are some example runs where additional input validation is being shown and where the user is entering invalid numbers:

```
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
2
1.      Chansey     0
What would you like to do?:
1. Display Complete PokeDex
```

```
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
0
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
7
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
3
What rarity of Pokemon would you like catch?:
1. Common (High Probability)
2. Uncommon (Normal Probability)
3. Ultra Rare (Extremely Low Probability)
0
What rarity of Pokemon would you like catch?:
1. Common (High Probability)
2. Uncommon (Normal Probability)
3. Ultra Rare (Extremely Low Probability)
4
What rarity of Pokemon would you like catch?:
1. Common (High Probability)
2. Uncommon (Normal Probability)
3. Ultra Rare (Extremely Low Probability)
```

```
2
You start to search.
You found a Alakazam
Alakazam added to your team!
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
4
You are going to fight a Metapod
The enemy has a CP of 337
Which of your pokemon would you like to use?:
1.      Chansey     0
2.     Alakazam  1269
2
You won!!
What would you like to do?:
1. Display Complete PokeDex
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
5
Which of your pokemon would you like to use?:
1.      Chansey     0
2.     Alakazam  1269
0
3
2
Your Alakazam's CP goes up!
What would you like to do?:
1. Display Complete PokeDex
```

```
2. Display your Team
3. Search for a new Pokemon
4. Battle your Pokemon
5. Train your Pokemon
6. Exit
```

There is a longer run of the program available as **`proj2_sample1.txt`** with the other files.

7. Compiling and Running

We have provided you with a sample **`makefile`** which should help you compile all of the classes and the program itself.

Once you have compiled using the provided **`makefile`**, enter the command **`make run`** or **`./proj2`** to run your program. If your executable is not proj2, you will lose points. It should look like the sample output provided above.

8. Completing your Project

When you have completed your project, you can copy it into the submission folder. You can copy your files into the submission folder as many times as you like (before the due date). We will only grade what is in your submission folder.

For this project, you should submit the following files to the **`proj2`** subdirectory:

`Pokemon.cpp, Pokemon.h`

`Game.cpp, Game.h`

`proj2.cpp`

You do not need to submit the **`makefile`**.

As you should have already set up your symbolic link for this class, you can just copy your files listed above to the submission folder. You can also make a rule in your makefile to copy all of the files using make submit if you would like (although it is not provided).

A. cd to your project 2 folder. An example might be cd **`~/202/projects/proj2`**

B. **`cp Pokemon.h Pokemon.cpp Game.h Game.cpp proj2.cpp ~/cs202proj/proj2`**

You can check to make sure that your files were successfully copied over to the submission directory by entering the command:

`ls ~/cs202proj/proj2`

Make sure that the required files are submitted by the deadline. If the copy command provided does not work, it is your responsibility to figure out what is wrong and that all required files have been submitted.

You can check that your program compiles and runs in the **`proj2`** directory, but please clean up any **`.o`** and executable files. Again, do not develop your code in this directory and you should not have the only copy of your program here. Uploading of any **`.gch`** files will result in a severe penalty.

IMPORTANT: If you want to submit the project late (after the due date), you will need to copy your files to the appropriate late folder. If you can no longer copy the files into the proj2 folder, it is because the due date has passed. You should be able to see your proj2 files but you can no longer edit or copy the files in to your proj2 folder. (They will be read only)

- If it is 0-24 hours late, copy your files to **`~/cs202proj/proj2-late1`**
- If it is 24-48 hours late, copy your files to **`~/cs202proj/proj2-late2`**
- If it is after 48 hours late, it is too late to be submitted.