BLG 252E - Object Oriented Programming
Assignment #3

Due: May, 12th 23:59

Introduction

In this assignment, you are expected to implement a simplified Pokemon Universe. The Pokemon Universe consists of two classes of Pokemons, namely, PsychicPokemons and ElectricPokemons. In addition, you are asked to design an arena on which one ElectricPokemon and one PsychicPokemon participate in a battle. For the purposes of this assignment, battles can only take place between one PsychicPokemon and one ElectricPokemon. This assignment is designed to reinforce your knowledge of inheritance.

For any issues regarding the assignment, please contact Meral Kuyucu (**korkmazmer@itu.edu.tr**).

Implementation Notes

The implementation details below are included in the grading:

- Please follow a consistent coding style (indentation, variable names, etc.) with comments. You are advised to revise the suggested coding practices from the course documents.
- Do not use any pre-compiled header files or STL commands.
- You may (and should) implement any getter and setter methods.
- Make sure that there is no memory leak in your code.
- Define required methods and variables as constanst in all necessary cases.
- Do not make any modifications in the given main.cpp. Your code will be evaluated using the given main.cpp and other alternatives.
- Ensure that your outputs match with the sample scenario for the given inputs.

Submission Notes

1. Your program should compile and run on the Linux environment provided to you. You should code and test your program on Docker's Virtual Environment.
2. You are expected to submit **pokemon.h, electricPokemon.h, psychicPokemon.h, pokemon.cpp, electricPokemon.cpp, psychicPokemon.cpp, arena.h and arena.cpp** containing all of the necessary declarations and definitions. In addition, you are expected to submit a report detailing your work.
3. You should compress your files into an archive file (`.zip`) and only submit necessary files. Also, please write your name and ID on the top of each document that you will upload as in following format:

```
/* @Author
* Student Name:  <student_name>
* Student ID : <student_id>
*/
```

4. Do not change included libraries and path to header files.
5. Submissions are made through only the Ninova system and have a strict deadline. Assignments submitted after the deadline will not be accepted. Do not risk leaving your submission to the last few minutes. Do not send your solutions by e-mail.
6. This is not a group assignment, and getting involved in any kind of cheating is subject to disciplinary actions. Your homework should not include any copy-paste material (from the Internet or from someone else's paper/thesis/project).

1 Implementation Details

This assignment requires that you design and implement four classes: **Pokemon, ElectricPokemon, PsychicPokemon and Arena.**

1.1 Pokemon

The Pokemon class contains a generic definition for a Pokemon. Each Pokemon should have the following defining characteristics:

- **Name:** How we refer to the Pokemon.
  - **Example = "Pikachu"**
- **Hit Points (HP) (Number):** How much damage a Pokemon can receive before fainting. This indicates the lifeline of the Pokemon. In each round of the battle, Pokemons take turns attacking eachother. The lifeline of a Pokemon drops by the amount of damage asserted by the opposing Pokemon.
  - **Example = 100**
  - To better illustrate, consider a Pokemon with HP = 100. If the opponent of the Pokemon has a Damage value of 10, the HP of the Pokemon taking the hit drops to 90 at the end of Round 1 (provided the battle is not on any particular terrain and neither Pokemon has used a PowerUp in this round).
- **Damage (Number):** How much damage a Pokemon will cause in a round of the battle. This value indicates the intensity of the Pokemon's attack. This number **cannot** be changed once it is defined for a Pokemon.
  - **Example = 10**
- **PowerUp Chance (Fraction/Percentage):** The chance that it will PowerUp on a given round of the battle. You can use random numbers to simulate the chance of PowerUp. This number **cannot** be changed once it is defined for a Pokemon.
  - **Example = 1/10 or 10%**
- **Recharge (Number of Rounds):** The number of rounds that must pass until a Pokemon has the chance to attempt to PowerUp again. This number cannot be changed once it is defined for a Pokemon.
  - **Example = 3**
- Additionally, each Pokemon should have other members which contain information about its state such as **whether or not it has fainted**, **whether or not it is currently in a PowerUp state** and **whether or not it is on its own terrain**. Consider the problem and design class members accordingly.

Whether or not a Pokemon is in a PowerUp state is determined in the same way for each Pokemon, however, the chance for PowerUp and the number of rounds it takes to recharge may change depending on the type of Pokemon. The way a Pokemon receives damage is also the same for each Pokemon. In the following subsection, characteristics of Pokemons that vary with type are provided in further detail.

1.2 ElectricPokemon

The ElectricPokemon **is a** special type of Pokemon. It causes damage to its opponent by means of electrocution. The attacking capabilities of ElectricPokemons are specialized. They are able to **PowerUp 20%** of the time. After using a PowerUp in one round, they must **wait for three** rounds to reattempt. A PowerUp enables the ElectricPokemon to generate **300% of its regular damage in one round**.

The ElectricPokemon is **prone to confusion** (not all pokemons are). When it is confused, it may attempt to perform attacks and PowerUps as usual, however, the attacks should result in no damage to its opponent for three rounds.

If an ElectricPokemon is on **electric terrain**, then it is **immune to confusion attacks** generated by PsychicPokemons.

1.3 PsychicPokemon

The PsychicPokemon is another specific **type of Pokemon** with psychic abilities. It is able to confuse, and therefore temporarily hinder the attacking capabilities of its opponent in addition to causing damage to its opponent's HP. The PsychicPokemon itself **is not prone to confusion**. In this universe, PsychicPokemons will not be battling any other PsychicPokemons anyways. PsychicPokemons also have the ability to **PowerUp with a 30% chance**. On a PowerUp, they are able to confuse their opponents and cause as much damage as their regular Damage value. After a PowerUp, the recharge time is **5 rounds** for a PsychicPokemon.

If a PsychicPokemon is battling on **psychic terrain**, it is able to induce damage its opponent **twice in one round**.

1.4 Arena

The Arena houses battles between Pokemons. It can contain up to two pokemons. One PsychicPokemon and one ElectricPokemon can enter an arena for battle (any other combination of Pokemons is not possible). The arena also has a type of terrain indicated by a string. During a battle, an Arena can have three different types of terrain, namely, Psychic, Electric, and None. The terrain can vary throughout a battle and the state of terrain is reset at the beginning of each battle. The Arena is responsible for:

- **Adding Pokemons for Battle:** A battle can be conducted if one PsychicPokemon and one ElectricPokemon are in the arena. The addPokemon method takes as arguments the type of Pokemon ('e' for electric, 'p' for psychic), the name of the Pokemon, the HP and damage.
  - void addPokemon(char type, string name, int hp, int damage);
- **Simulating Battles:** Pokemons take turns attacking each other until one faints (runs out of HP). The first Pokemon to attack is determined randomly at the beginning of each round. Use the function prototype provided below:
  - void simulateBattle();
- **Spawning Terrains:** The Arena can spawn one of three types of terrains: electric, psychic or none. Electric and psychic terrains can be spawned with 20% chance each and there is a 60% chance that the terrain type is none. If either of electric or psychic terrains have been spawned, the arena retains this terrain for 5 rounds. At the end of the 5 rounds, a terrain is spawned again. If the terrain spawned is none, then the spawn attempt is repeated in the next round without retention. In the case that the terrain is none, the pokemons continue battling without the added advantages of battling on their own terrains.

  If new Pokemons enter the arena for a battle, the terrain must be reset. Even if the terrain was electric for the last two rounds of the previous battle, a new terrain spawn attempt is made with 20% electric, 20% psychic and 60% none chances. Note that the performance of Pokemons vary with the type of terrain they stand on as described above.

Depending on the type of terrain, the arena is responsible of informing the Pokemon. For example, if the terrain is electric, the arena must inform the ElectricPokemon that it is on electric terrain. Use the function prototype provided below:

  - void spawnTerrain();
  - void restartTerrain();

- Don't forget to print the stats at the end of each round in the format provided in **output.txt**. The printRoundStats method prints the status of the battle at the end of each round. The round argument is the round number. Heads is a random number generated in simulateBattle to determine which Pokemon goes first in the round.

  - void printRoundStats(int round, int heads);
  - void printMatchResults();

Warning: Make sure you use `const` modifier whenever necessary. Do not implement any function that is not required.

2 Main Method

A `main.cpp` file is provided along with this document. Please do not change anything. The output your code should generate for the main method provided is given in `output.txt`.

In the main method, an object of the Arena class is created. Then, two pokemons are added to the arena. The inputs of the AddPokemon method are the type ('e' for electric, 'p' for psychic), name, HP, and damage of the pokemon respectively. These pokemons should be created in the Arena class. After one electric and one psychic pokemon are added to the arena, the simulateBattle method is called. After the battle comes to an end, two other pokemons enter the arena in the same way and another battle is simulated between the new pokemons.

Warning: In the `main.cpp` file given to you, the random generator seed is constant. Therefore, the same sequence of random numbers will be generated. However, depending on your design and implementation, the sequence of operations that the random numbers will be used for may vary. Therefore, you may not generate the exact same output as the one given to you. It is important that your output is consistent within itself, and the output your code generates is in the same format as the one given to you so that it is traceable when grading.

3 Report

Draw a **UML class diagram** that represents the relationships between the classes you are asked to design and implement. Provide all necessary details of your design including but not limited to parameter visibility and name hiding.

4 Compilation and Execution

You can build your code with the following command:

```
$ g++ -Wall -Werror src/pokemon.cpp src/arena.cpp src/electricPokemon.cpp
src/psychicPokemon.cpp src/main.cpp -I include -o assignment3

$ ./assignment3
```

5 For Independent Practice:

After completing this assignment and polymorphism has been covered in lectures, you will have laid the foundations for the implementation of polymorphic code. In your own time, you are advised to try your hand at incorporating polymorphism into your implementation. Consider increasing the number of Pokemons that can enter the arena and varying the combinations of Pokemons that can participate in battle. You can even try implementing three-way (or even n-way) battles!

"The true sign of intelligence is not knowledge but imagination." Albert Einstein

Warning: This assignment is not for testing your knowledge on polymorphism. **No polymorphic implementation should be submitted.** This section is only included to advise practicing **in your free time after submitting the required code described above.**