Encapsulation & Inheritance & Polymorphism in C++

Lab 11

TA : Changmin Jeon, Minji Kim, Hyunwoo Jung,
Hyunseok Oh, Jingyu Lee, Seungwoo Jo

SEOUL NATIONAL UNIVERSITY

Objectives

- Get used to C++ project structure. (*.cpp, *.h)
- Practice OOP with C++
  - ✓ Inheritance
    - ➢ Class extending
    - ➢ Method overriding
  - ✓ Encapsulation
    - ➢ Access modifiers
  - ✓ Polymorphism
    - ➢ Operator overriding

Pocke*t*mon Game Application

- Some of you (Older students...) may have some memories of Pokemon games.
- We will make 2-player pocketmon game.

Enemy MILTANK
used ROLLOUT!

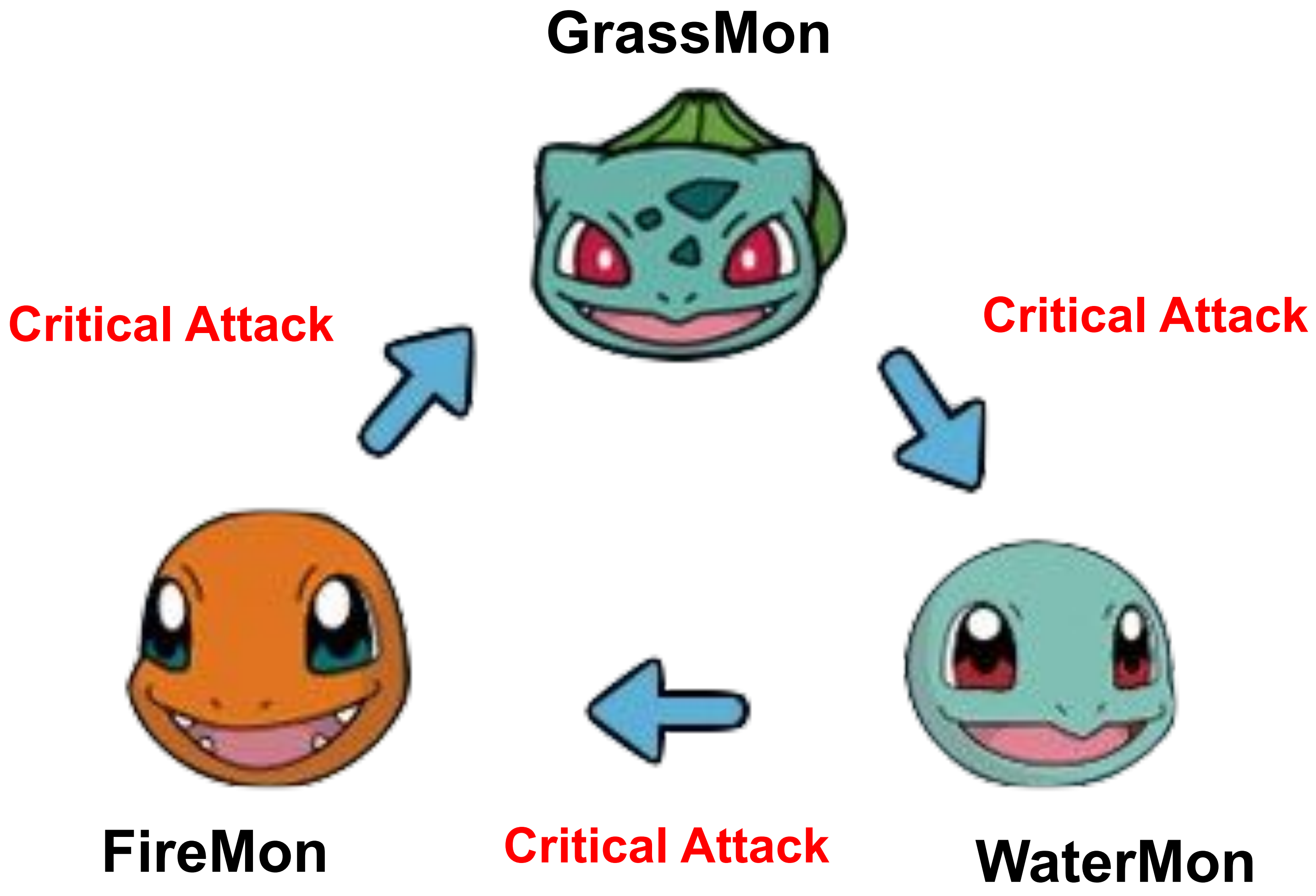
Pocket*t*mon Game Application

- There are two players in the game.
- Each player can have at most 6 monsters.
- There are three types of monster, WaterMon, FireMon, and GrassMon.
- At each round,
  a. Each player chooses one of the monster.
  b. The 1st player's chosen monster attacks the 2nd player's monster, and then the 2nd player's monster attacks back.
  c. If a monster's hp is below or equal to zero after a fight, the monster is excluded from the player.

Normal & Critical Attack

- The default attack damage is given for each type of monster.
- At each attack, the health of the target is decreased by the amount of the damage.
- Each type of monster has a counter monster type.
  WaterMon > FireMon > GrassMon > WaterMon ...
  - For example a WaterMon performs critical attack to a FireMon.
- Each type of monster has its own critical attack.

Normal & Critical Attack



Overview

- Let’s look at the code.

Problem1

- Add `id` attribute for `Monster` class objects by using `num_monsters` attribute in `Monster` class.
- Example
  - ✓ First created Monster object `id` : 0
  - ✓ Second : 1
  - ✓ Third : 2
  - ✓ …

Problem2

- Implement `critical_attack` method for `Monster`, `WaterMon`, `FireMon`, and `GrassMon`. `XXMon` should override `critical_attack` method of `Monster` class.
- `void critical_attack(Monster *attacked_monster)`

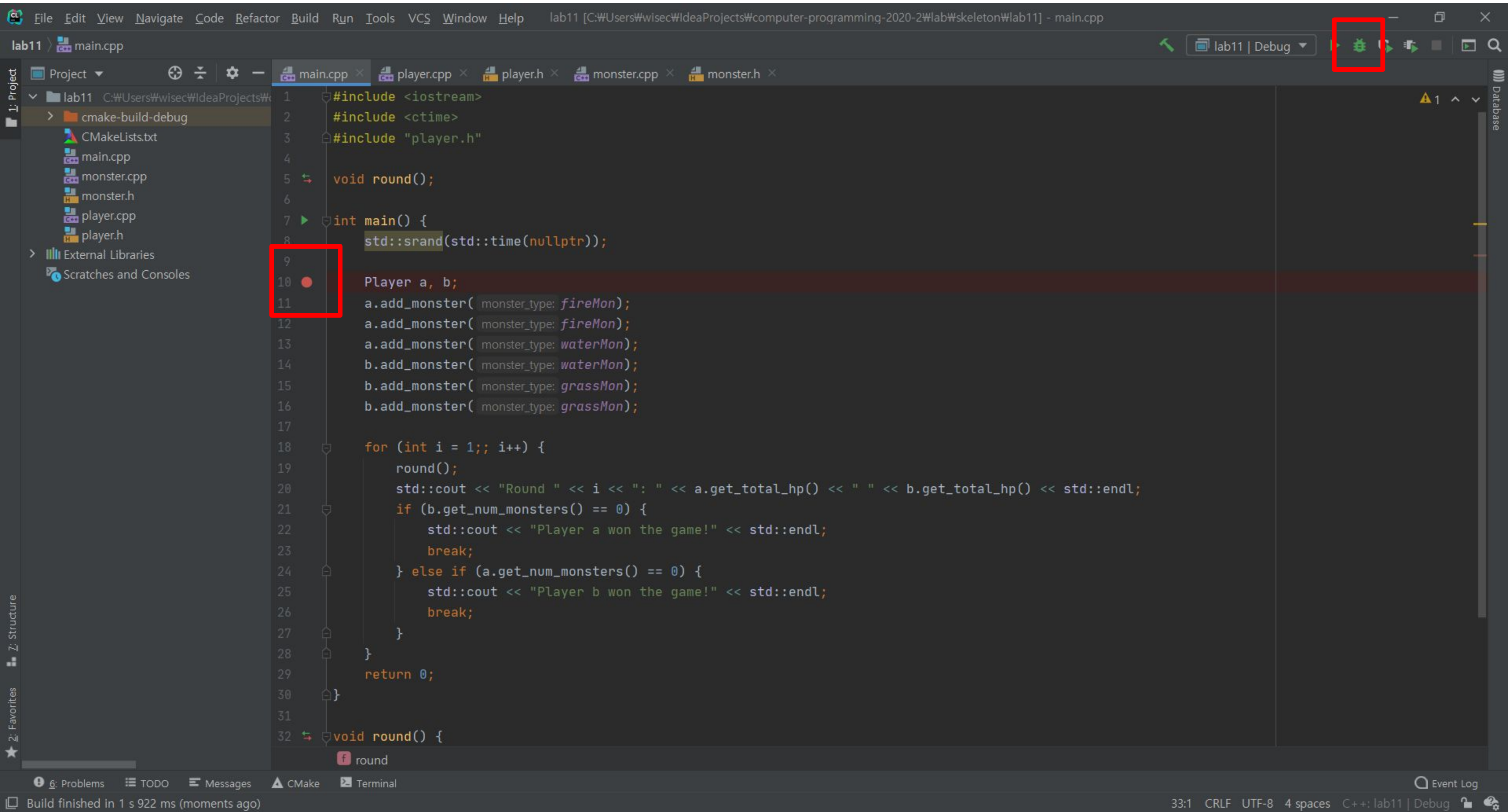
Monster	: 2 × damage
WaterMon	: damage^2 / 2
FireMon	: Random in range (0 ~ 10 × damage).
GrassMon	: 3 × damage

You can use `std::rand()/RAND_MAX` in <cstdlib>.
`std::rand()` will return integer value between 0 ~ `RAND_MAX`.

Problem3

- Implement `round()` function in main.cpp.
- You can change the signature of the function.
- At each round,
  - Each player chooses one of the monster.
  - The 1st player's chosen monster attacks the 2nd player's monster, and then the 2nd player's monster attacks back.
  - If a monster's hp is below or equal to zero after a fight, the monster is excluded from the player.
- You can use `Player::select_monster`, `Monster::attack`, and `Player::delete_monster` methods.

Debugging in C++



```cpp
#include <iostream>
#include <ctime>
#include "player.h"

void round();

int main() {
    std::srand(std::time(nullptr));

    Player a, b;
    a.add_monster( monster_type: fireMon);
    a.add_monster( monster_type: fireMon);
    a.add_monster( monster_type: waterMon);
    b.add_monster( monster_type: waterMon);
    b.add_monster( monster_type: grassMon);
    b.add_monster( monster_type: grassMon);

    for (int i = 1;; i++) {
        round();
        std::cout << "Round " << i << ": " << a.get_total_hp() << " " << b.get_total_hp() << std::endl;
        if (b.get_num_monsters() == 0) {
            std::cout << "Player a won the game!" << std::endl;
            break;
        } else if (a.get_num_monsters() == 0) {
            std::cout << "Player b won the game!" << std::endl;
            break;
        }
    }
    return 0;
}

void round() {
```

Build finished in 1 s 922 ms (moments ago)

Shortcut

Ctrl + F8	: Toggle Breakpoint
Shift + F9	: Start Debugging
F8	: Step Over
F7	: Step Into
Shift + F8	: Step Out
F9	: Resume Program

Submission

- Download skeleton files from eTL
- Compress your Project directory into a zip file.
- Rename your zip file as 20XX-XXXXX_{name}.zip
  - for example, 2020-12345_KimMinji.zip
- Upload it to eTL - Lab 11 assignment.