Discord

OneDrive

Mac Only :(

CS225: Project Pokémon

Wonjoon Jun
Bonggyu Seo

TABLE OF CONTENTS

01

DEMONSTRATION OF THE GAME

02

UML

03

STYLE GUIDE

04

CODE DESCRIPTION

05

SOMETHING NEW!

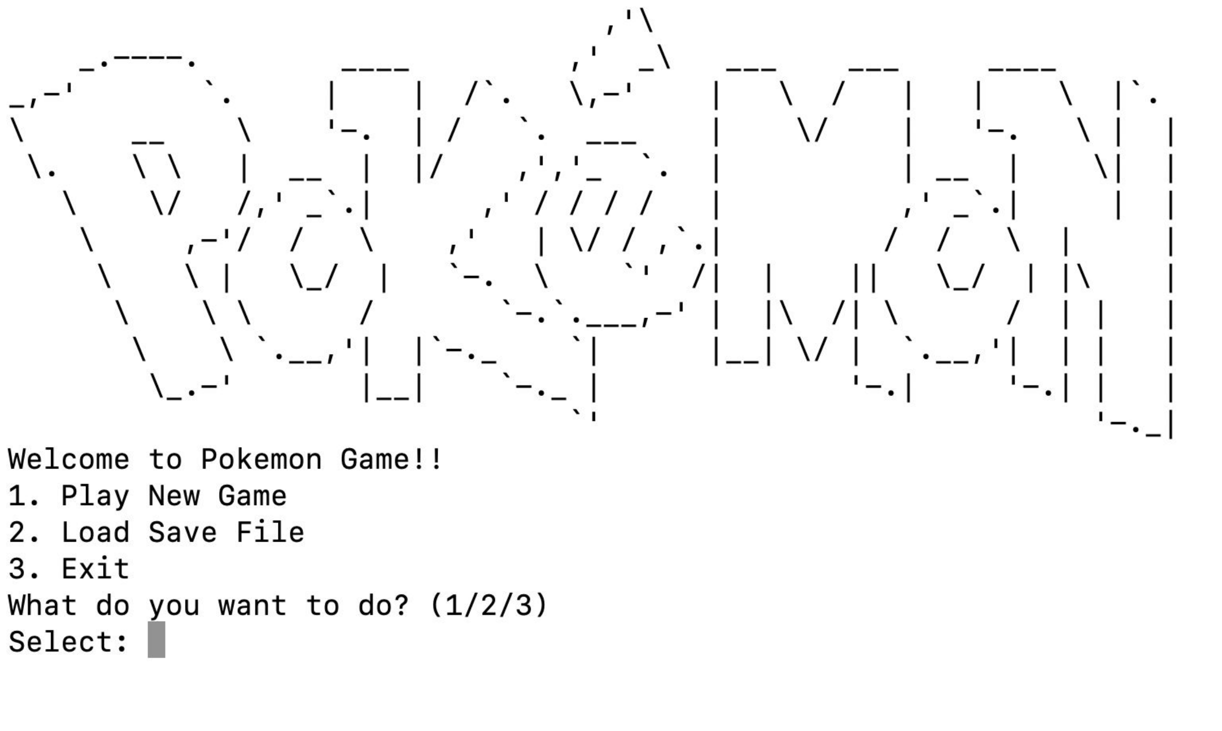
06

INTRODUCING MORE FEATURES ABOUT THE GAME
(IF TIME ALLOWS)

01

Demonstration

Demonstration of the Game



```
                                  ,'\
    _.----.        ____         ,'  _\   ___    ___     ____
_,-'       `.     |    |  /`.   \,-'    |   \  /   |   |    \  |`.
\      __    \    '-.  | /   `.  ___    |    \/    |   '-.   \ |  |
 \.    \ \   |  __  |  |/    ,','_  `.  |          | __  |    \|  |
   \    \/   /,' _`.|      ,' / / / /   |          ,' _`.|     |  |
    \     ,-'/  /   \    ,'   | \/ / ,`.|         /  /   \  |     |
     \    \ |   \_/  |   `-.  \    `'  /|  |    ||   \_/  | |\    |
      \    \ \      /       `-.`.___,-' |  |\  /| \      /  | |   |
       \    \ `.__,'|  |`-._    `|      |__| \/ |  `.__,'|  | |   |
        \_.-'       |__|    `-._ |              '-.|     '-.| |   |
                                `'                            '-._|
Welcome to Pokemon Game!!
1. Play New Game
2. Load Save File
3. Exit
What do you want to do? (1/2/3)
Select:
```

Demonstration

1

Creating a New Game

2

Choose Starting

3

Go to First Map and Battle

4

Go to Last Map and Catch the Pokemon

5

Sort The Pokemon By the strongest

6

Fight the Gym Leader

Demonstration

7

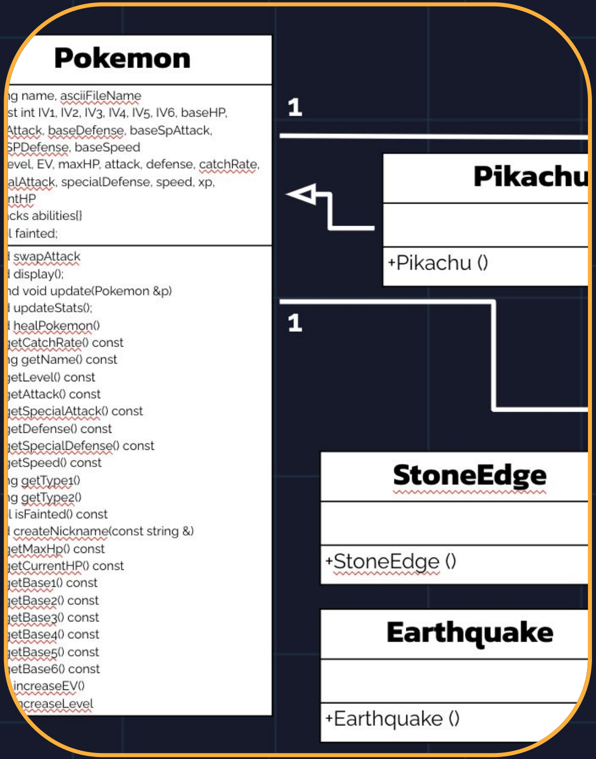
Save and Quit: Load game once more

8

Buy TM

9

Save The game



Pokemon

ıg name, asciiFileName
st int IV1, IV2, IV3, IV4, IV5, IV6, baseHP,
Attack, baseDefense, baseSpAttack,
SPDefense, baseSpeed
evel, EV, maxHP, attack, defense, catchRate,
alAttack, specialDefense, speed, xp,
ntHP
cks abilities[]
l fainted;

l swapAttack
l display();
nd void update(Pokemon &p)
l updateStats();
l healPokemon()
getCatchRate() const
ng getName() const
getLevel() const
getAttack() const
getSpecialAttack() const
getDefense() const
getSpecialDefense() const
getSpeed() const
ng getType1()
ng getType2()
l isFainted() const
l createNickname(const string &)
getMaxHp() const
getCurrentHP() const
getBase1() const
getBase2() const
getBase3() const
getBase4() const
getBase5() const
getBase6() const
increaseEV()
ncreaseLevel

1

Pikachu

+Pikachu ()

1

StoneEdge

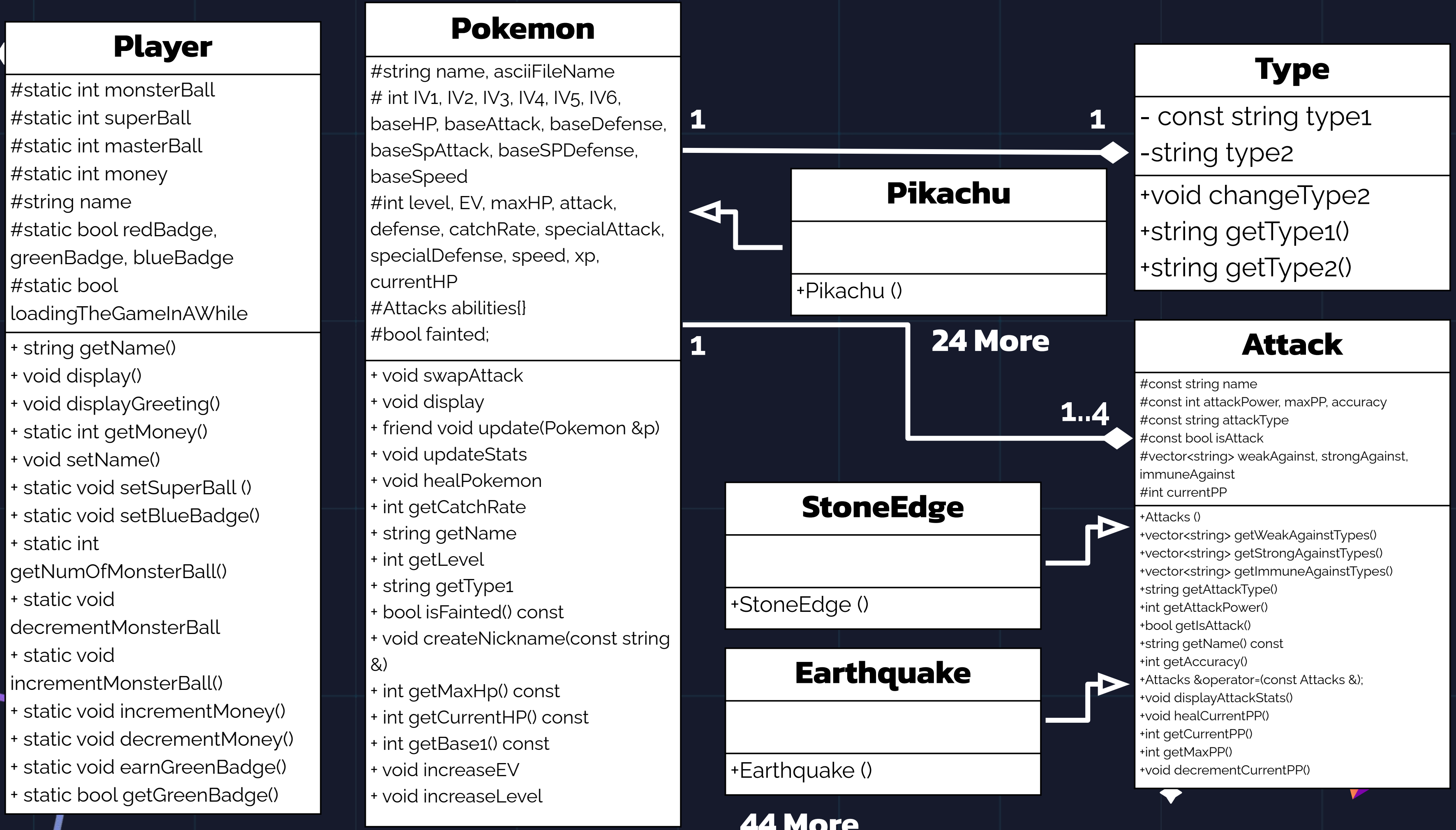
+StoneEdge ()

Earthquake

+Earthquake ()

02

UML



Player

#static int monsterBall
#static int superBall
#static int masterBall
#static int money
#string name
#static bool redBadge, greenBadge, blueBadge
#static bool loadingTheGameInAWhile

+ string getName()
+ void display()
+ void displayGreeting()
+ static int getMoney()
+ void setName()
+ static void setSuperBall ()
+ static void setBlueBadge()
+ static int getNumOfMonsterBall()
+ static void decrementMonsterBall
+ static void incrementMonsterBall()
+ static void incrementMoney()
+ static void decrementMoney()
+ static void earnGreenBadge()
+ static bool getGreenBadge()

Pokemon

#string name, asciiFileName
# int IV1, IV2, IV3, IV4, IV5, IV6, baseHP, baseAttack, baseDefense, baseSpAttack, baseSPDefense, baseSpeed
#int level, EV, maxHP, attack, defense, catchRate, specialAttack, specialDefense, speed, xp, currentHP
#Attacks abilities[]
#bool fainted;

+ void swapAttack
+ void display
+ friend void update(Pokemon &p)
+ void updateStats
+ void healPokemon
+ int getCatchRate
+ string getName
+ int getLevel
+ string getType1
+ bool isFainted() const
+ void createNickname(const string &)
+ int getMaxHp() const
+ int getCurrentHP() const
+ int getBase1() const
+ void increaseEV
+ void increaseLevel

Pikachu

+Pikachu ()

24 More

Type

- const string type1
-string type2

+void changeType2
+string getType1()
+string getType2()

## Attack

#const string name
#const int attackPower, maxPP, accuracy
#const string attackType
#const bool isAttack
#vector<string> weakAgainst, strongAgainst, immuneAgainst
#int currentPP

+Attacks ()
+vector<string> getWeakAgainstTypes()
+vector<string> getStrongAgainstTypes()
+vector<string> getImmuneAgainstTypes()
+string getAttackType()
+int getAttackPower()
+bool getIsAttack()
+string getName() const
+int getAccuracy()
+Attacks &operator=(const Attacks &);
+void displayAttackStats()
+void healCurrentPP()
+int getCurrentPP()
+int getMaxPP()
+void decrementCurrentPP()

StoneEdge

+StoneEdge ()

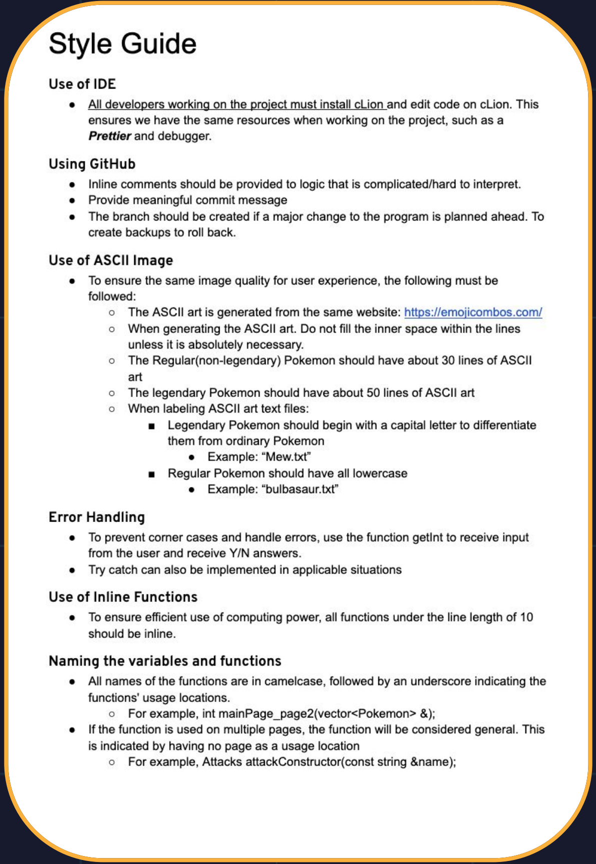
Earthquake

+Earthquake ()

44 More

03

Style Guide



Style Guide

Use of IDE

- All developers working on the project must install cLion and edit code on cLion. This ensures we have the same resources when working on the project, such as a ***Prettier*** and debugger.

Using GitHub

- Inline comments should be provided to logic that is complicated/hard to interpret.
- Provide meaningful commit message
- The branch should be created if a major change to the program is planned ahead. To create backups to roll back.

Use of ASCII Image

- To ensure the same image quality for user experience, the following must be followed:
  - The ASCII art is generated from the same website: https://emojicombos.com/
  - When generating the ASCII art. Do not fill the inner space within the lines unless it is absolutely necessary.
  - The Regular(non-legendary) Pokemon should have about 30 lines of ASCII art
  - The legendary Pokemon should have about 50 lines of ASCII art
  - When labeling ASCII art text files:
    - Legendary Pokemon should begin with a capital letter to differentiate them from ordinary Pokemon
      - Example: “Mew.txt”
    - Regular Pokemon should have all lowercase
      - Example: “bulbasaur.txt”

Error Handling

- To prevent corner cases and handle errors, use the function getInt to receive input from the user and receive Y/N answers.
- Try catch can also be implemented in applicable situations

Use of Inline Functions

- To ensure efficient use of computing power, all functions under the line length of 10 should be inline.

Naming the variables and functions

- All names of the functions are in camelcase, followed by an underscore indicating the functions' usage locations.
  - For example, int mainPage_page2(vector<Pokemon> &);
- If the function is used on multiple pages, the function will be considered general. This is indicated by having no page as a usage location
  - For example, Attacks attackConstructor(const string &name);

Style Guide

Use of IDE

All developers working on the project must install cLion and edit code on cLion. This ensures we have the same resources when working on the project, such as a Prettier and debugger.

Using GitHub

- Inline comments should be provided to logic that is complicated/hard to interpret.
- Provide meaningful commit message
- The branch should be created if a major change to the program is planned ahead. To create backups to roll back.

Use of ASCII image

- The ASCII art is generated from the same website: https://emojicombos.com/
- The Regular (non-legendary) Pokemon should have about 30 lines of ASCII art
- The legendary Pokemon should have about 50 lines of ASCII art

Style Guide

Error Handling

- To prevent corner cases and handle errors, use the function getInt to receive input from the user and receive Y/N answers.
- Try catch can also be implemented in applicable situations

Use of Inline Functions

- To ensure efficient use of computing power, all functions under the line length of 10 should be inline.

Naming the variables/functions

- All names of the functions are in camelcase, followed by an underscore indicating the functions' usage locations.
- on multiple pages, the function will be considered general. This is indicated by having no page as a usage location

Style Guide

Formatting the Code

- The code format is adjusted using the program **Prettier**, which is available by default on cLion.
- Prettier enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

Use of the Goto command

- The goto function can be used when the function's return type is void.
- The style guide encourages the usage of the Goto command to end the function. This decreases the complexity of the code and increases the readability.

Loops

- The range-based loop should be used in an applicable situation. This increases the readability of the code.
- Refrain from using while loops for fixed range usages. This is to prevent non-ending loops.

Style Guide

Use of the Global Variables

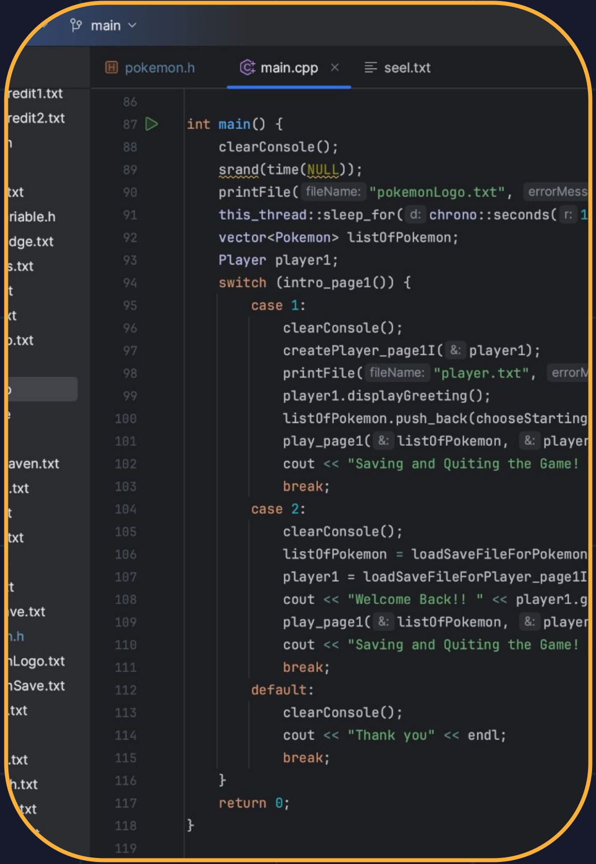
- Global variables are created in separate header files(globalVariable.h) to track what has been created.
- The global variables must be constant and cannot be altered within the code.
- Constant variables must be spelled with all capitalized letters. Therefore, all global variables should be capitalized.

Use of the ENUM

- Enum is replaced by const global variable to prevent the issue of having different int for the same variable name.

Docstring

- Docstring is in every cpp file
- Includes:
  - File name
  - Author
  - Purpose
  - Version
  - Resources

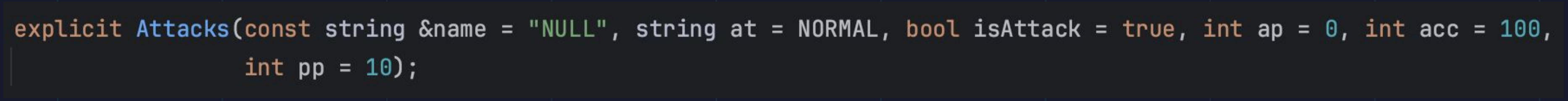


```
int main() {
    clearConsole();
    srand(time(NULL));
    printFile( fileName: "pokemonLogo.txt",  errorMess
    this_thread::sleep_for( d: chrono::seconds( r: 1
    vector<Pokemon> listOfPokemon;
    Player player1;
    switch (intro_page1()) {
        case 1:
            clearConsole();
            createPlayer_page1I( &: player1);
            printFile( fileName: "player.txt",  errorM
            player1.displayGreeting();
            listOfPokemon.push_back(chooseStarting
            play_page1( &: listOfPokemon,  &: player
            cout << "Saving and Quiting the Game!
            break;
        case 2:
            clearConsole();
            listOfPokemon = loadSaveFileForPokemon
            player1 = loadSaveFileForPlayer_page1I
            cout << "Welcome Back!! " << player1.g
            play_page1( &: listOfPokemon,  &: player
            cout << "Saving and Quiting the Game!
            break;
        default:
            clearConsole();
            cout << "Thank you" << endl;
            break;
    }
    return 0;
}
```

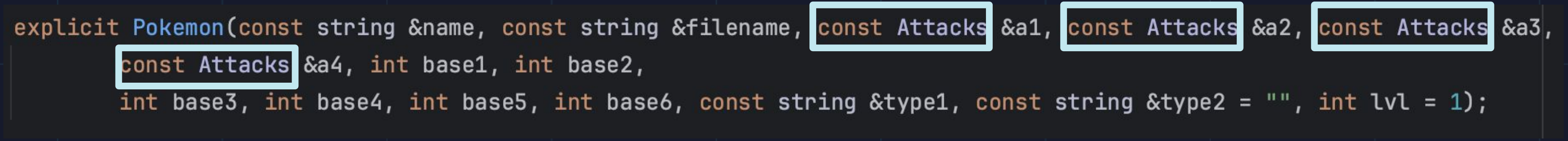
04

Code Description

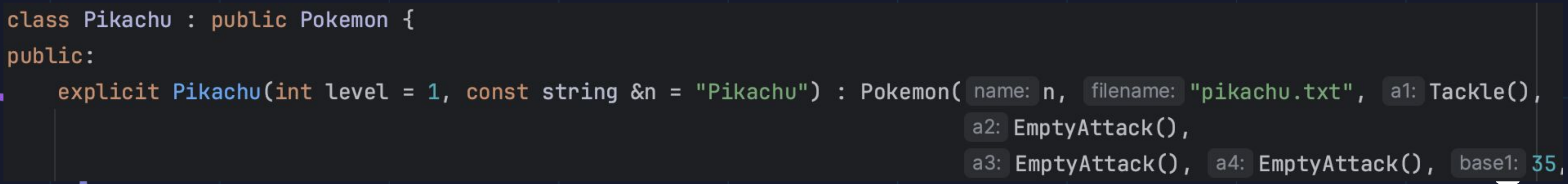
OBJECT COMPOSITION
INHERITANCE
CONSTRUCTORS



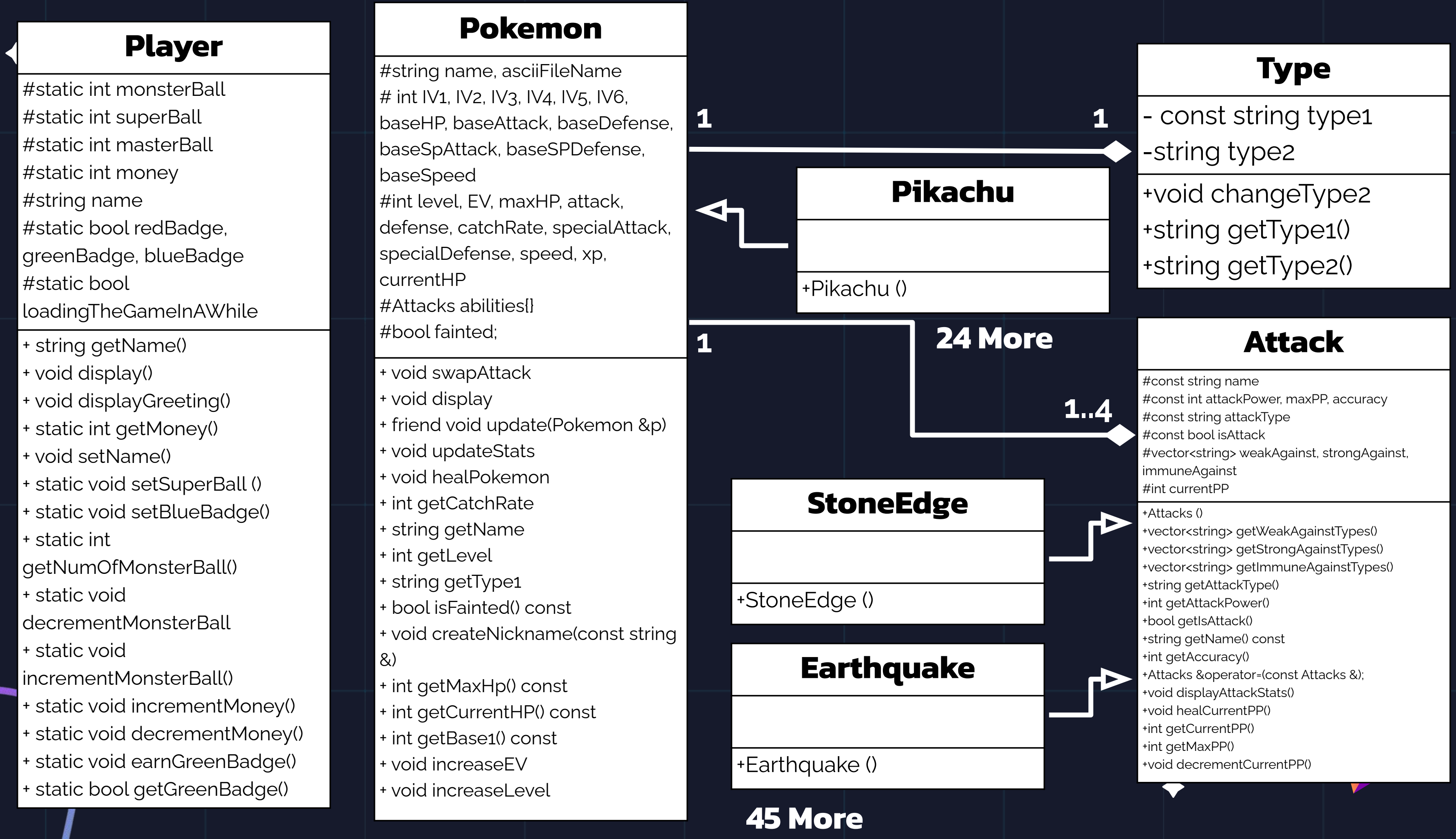
```
explicit Attacks(const string &name = "NULL", string at = NORMAL, bool isAttack = true, int ap = 0, int acc = 100,
                 int pp = 10);
```



```
explicit Pokemon(const string &name, const string &filename, const Attacks &a1, const Attacks &a2, const Attacks &a3,
        const Attacks &a4, int base1, int base2,
        int base3, int base4, int base5, int base6, const string &type1, const string &type2 = "", int lvl = 1);
```



```
class Pikachu : public Pokemon {
public:
    explicit Pikachu(int level = 1, const string &n = "Pikachu") : Pokemon( name: n,  filename: "pikachu.txt",  a1: Tackle(),
                                                                     a2: EmptyAttack(),
                                                                     a3: EmptyAttack(),  a4: EmptyAttack(),  base1: 35,
```



Player

#static int monsterBall
#static int superBall
#static int masterBall
#static int money
#string name
#static bool redBadge, greenBadge, blueBadge
#static bool loadingTheGameInAWhile

+ string getName()
+ void display()
+ void displayGreeting()
+ static int getMoney()
+ void setName()
+ static void setSuperBall ()
+ static void setBlueBadge()
+ static int getNumOfMonsterBall()
+ static void decrementMonsterBall
+ static void incrementMonsterBall()
+ static void incrementMoney()
+ static void decrementMoney()
+ static void earnGreenBadge()
+ static bool getGreenBadge()

Pokemon

#string name, asciiFileName
# int IV1, IV2, IV3, IV4, IV5, IV6, baseHP, baseAttack, baseDefense, baseSpAttack, baseSPDefense, baseSpeed
#int level, EV, maxHP, attack, defense, catchRate, specialAttack, specialDefense, speed, xp, currentHP
#Attacks abilities[]
#bool fainted;

+ void swapAttack
+ void display
+ friend void update(Pokemon &p)
+ void updateStats
+ void healPokemon
+ int getCatchRate
+ string getName
+ int getLevel
+ string getType1
+ bool isFainted() const
+ void createNickname(const string &)
+ int getMaxHp() const
+ int getCurrentHP() const
+ int getBase1() const
+ void increaseEV
+ void increaseLevel

Pikachu

+Pikachu ()

Type

- const string type1
-string type2

+void changeType2
+string getType1()
+string getType2()

Attack

#const string name
#const int attackPower, maxPP, accuracy
#const string attackType
#const bool isAttack
#vector<string> weakAgainst, strongAgainst, immuneAgainst
#int currentPP

+Attacks ()
+vector<string> getWeakAgainstTypes()
+vector<string> getStrongAgainstTypes()
+vector<string> getImmuneAgainstTypes()
+string getAttackType()
+int getAttackPower()
+bool getIsAttack()
+string getName() const
+int getAccuracy()
+Attacks &operator=(const Attacks &);
+void displayAttackStats()
+void healCurrentPP()
+int getCurrentPP()
+int getMaxPP()
+void decrementCurrentPP()

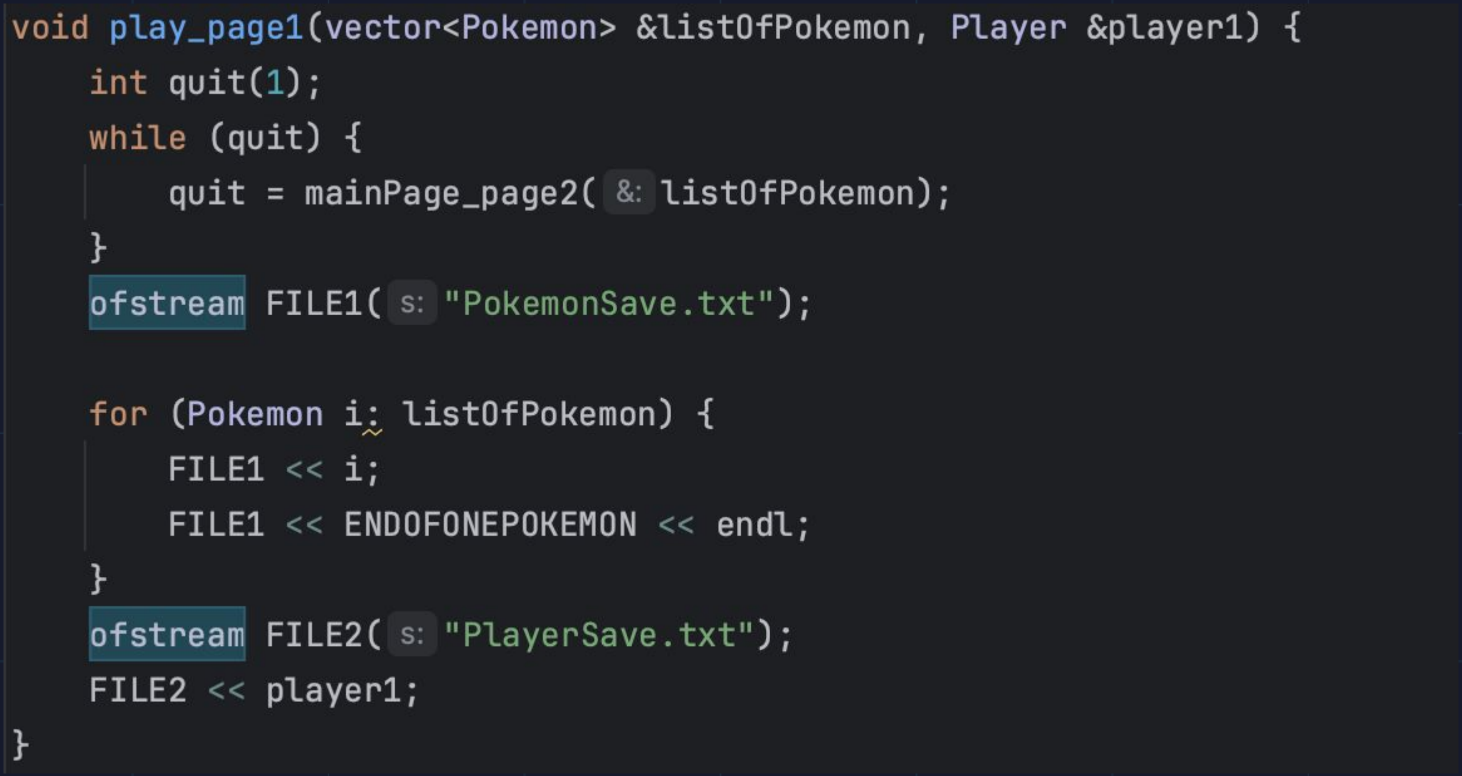
StoneEdge

+StoneEdge ()

Earthquake

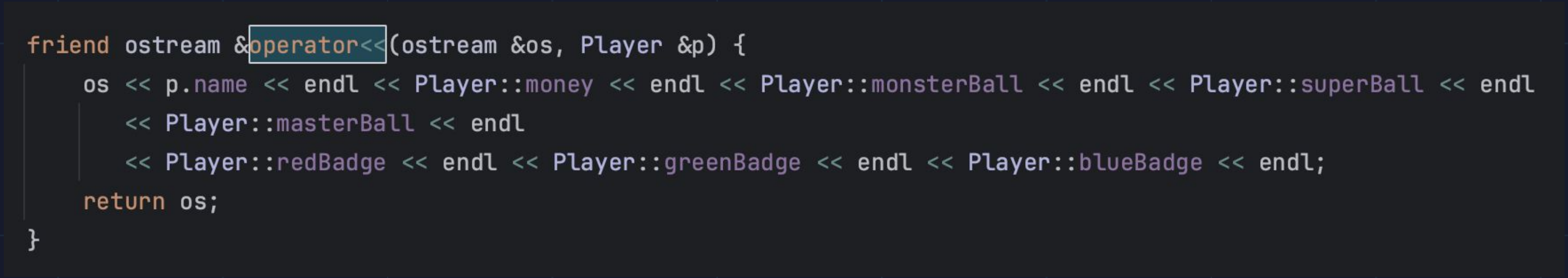
+Earthquake ()

FileI/O && OPERATOR OVERLOADING



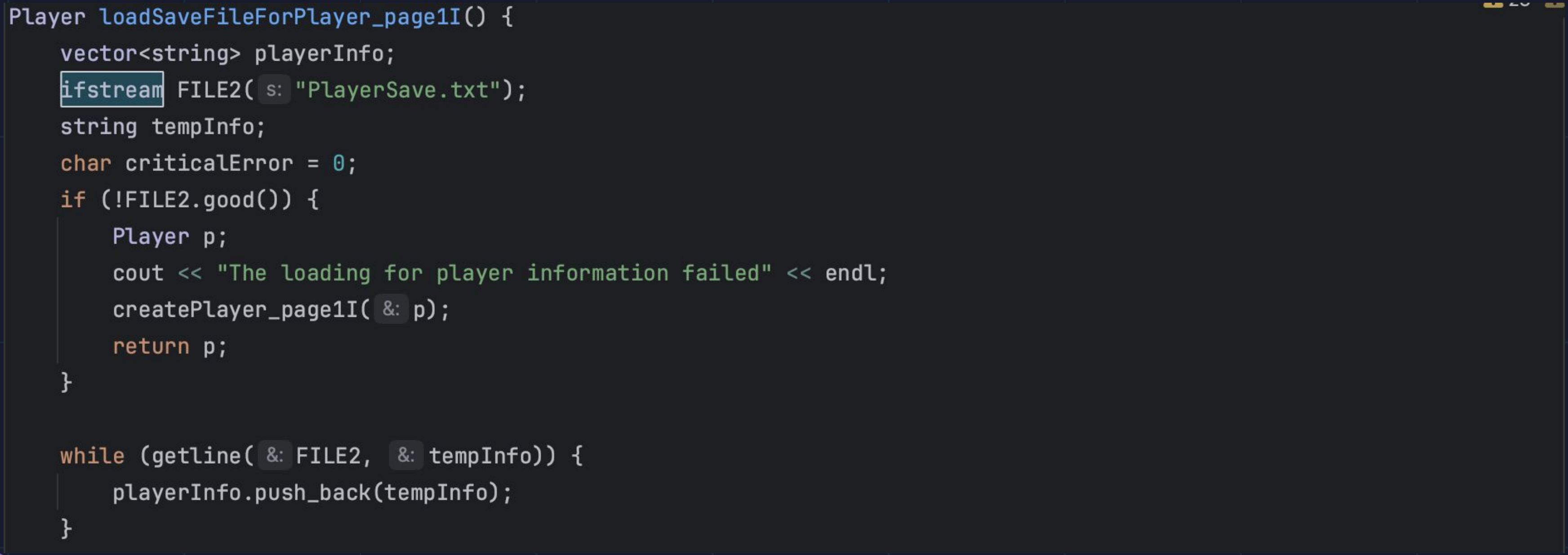
```
void play_page1(vector<Pokemon> &listOfPokemon, Player &player1) {
    int quit(1);
    while (quit) {
        quit = mainPage_page2( &: listOfPokemon);
    }
    ofstream FILE1( s: "PokemonSave.txt");

    for (Pokemon i: listOfPokemon) {
        FILE1 << i;
        FILE1 << ENDOFONEPOKEMON << endl;
    }
    ofstream FILE2( s: "PlayerSave.txt");
    FILE2 << player1;
}
```

FileI/O && OPERATOR OVERLOADING



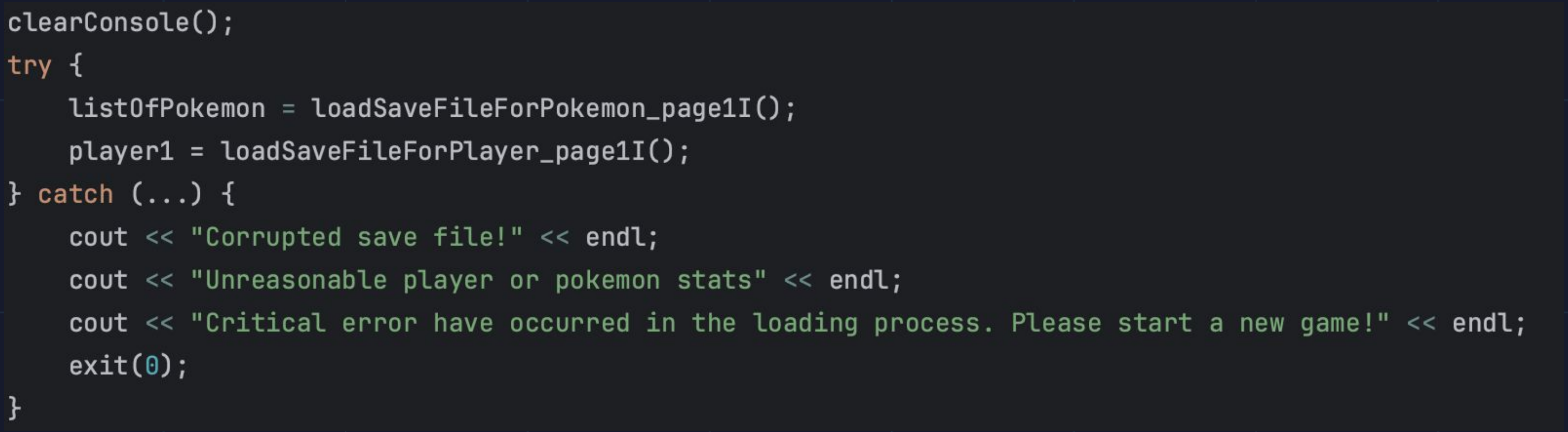
```
friend ostream &operator<<(ostream &os, Player &p) {
    os << p.name << endl << Player::money << endl << Player::monsterBall << endl << Player::superBall << endl
       << Player::masterBall << endl
       << Player::redBadge << endl << Player::greenBadge << endl << Player::blueBadge << endl;
    return os;
}
```

FileI/O && OPERATOR OVERLOADING



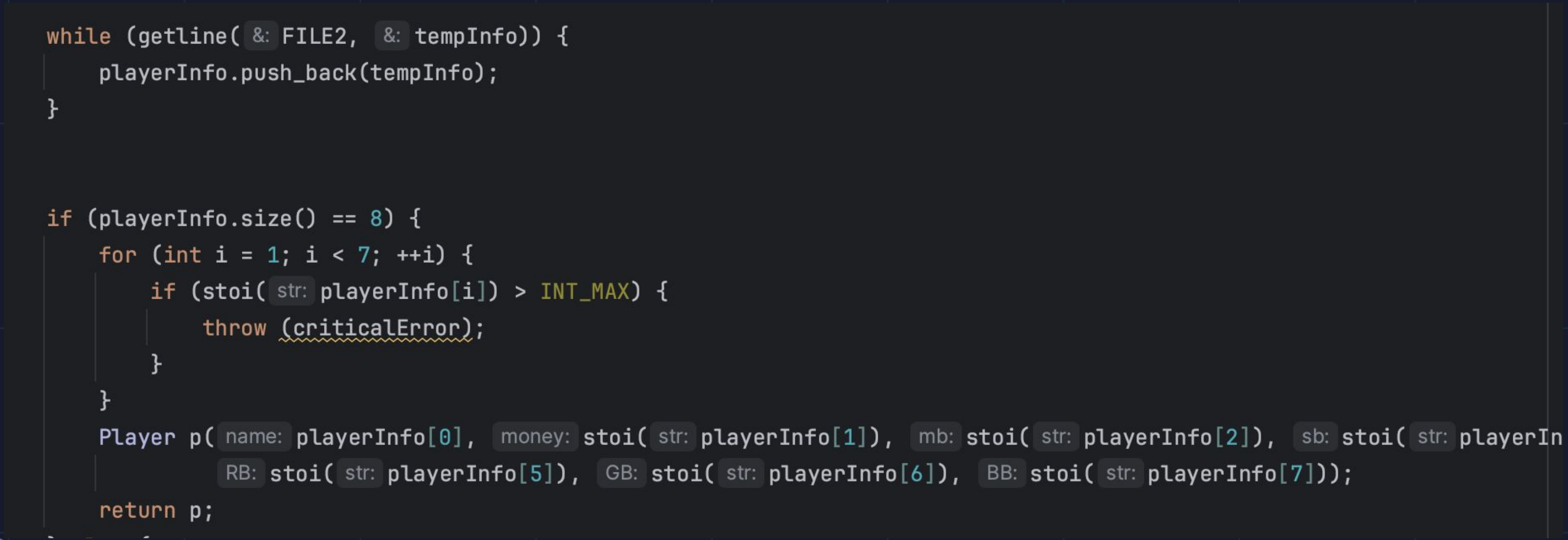
```
Player loadSaveFileForPlayer_page1I() {
    vector<string> playerInfo;
    ifstream FILE2( s: "PlayerSave.txt");
    string tempInfo;
    char criticalError = 0;
    if (!FILE2.good()) {
        Player p;
        cout << "The loading for player information failed" << endl;
        createPlayer_page1I( &: p);
        return p;
    }

    while (getline( &: FILE2,  &: tempInfo)) {
        playerInfo.push_back(tempInfo);
    }
```

Exceptions (try and catch)



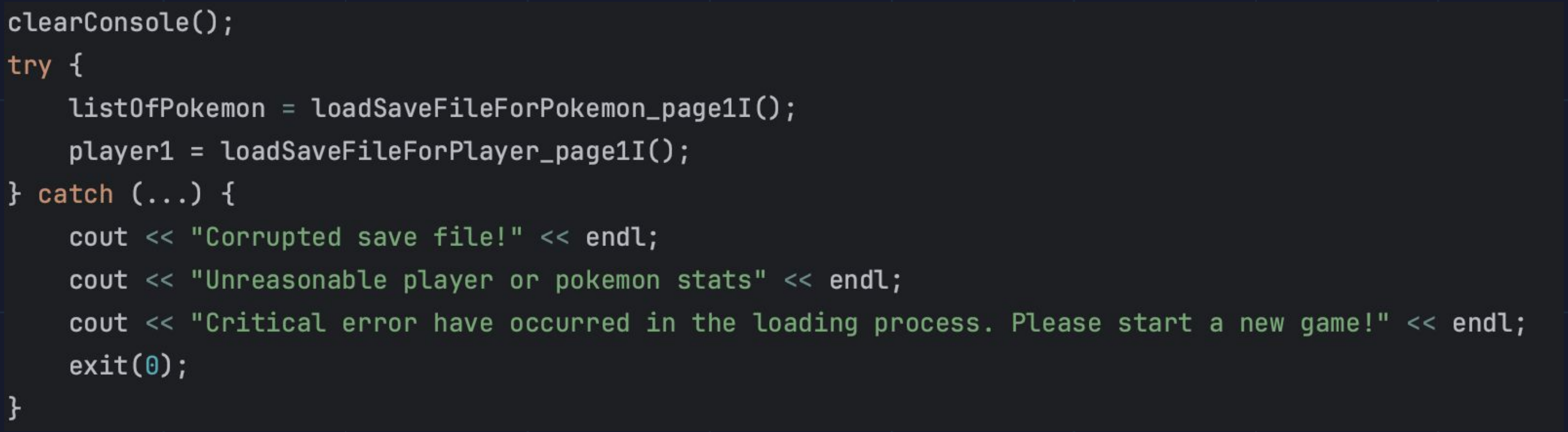
```
clearConsole();
try {
    listOfPokemon = loadSaveFileForPokemon_page1I();
    player1 = loadSaveFileForPlayer_page1I();
} catch (...) {
    cout << "Corrupted save file!" << endl;
    cout << "Unreasonable player or pokemon stats" << endl;
    cout << "Critical error have occurred in the loading process. Please start a new game!" << endl;
    exit(0);
}
```

Exceptions (try and catch)



```
while (getline( &: FILE2,  &: tempInfo)) {
    playerInfo.push_back(tempInfo);
}



if (playerInfo.size() == 8) {
    for (int i = 1; i < 7; ++i) {
        if (stoi( str: playerInfo[i]) > INT_MAX) {
            throw (criticalError);
        }
    }
    Player p( name: playerInfo[0],  money: stoi( str: playerInfo[1]),  mb: stoi( str: playerInfo[2]),  sb: stoi( str: playerIn
             RB: stoi( str: playerInfo[5]),  GB: stoi( str: playerInfo[6]),  BB: stoi( str: playerInfo[7]));
    return p;
```

Exceptions (try and catch)



```
clearConsole();
try {
    listOfPokemon = loadSaveFileForPokemon_page1I();
    player1 = loadSaveFileForPlayer_page1I();
} catch (...) {
    cout << "Corrupted save file!" << endl;
    cout << "Unreasonable player or pokemon stats" << endl;
    cout << "Critical error have occurred in the loading process. Please start a new game!" << endl;
    exit(0);
}
```

05

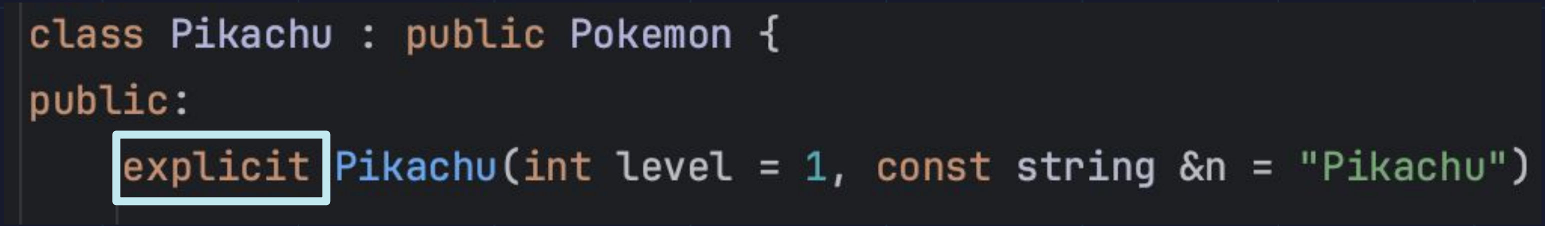
Something New!

explicit

Keyword: Explicit

Demonstration using *Online GDB:*

- Add them before the constructors
- Prevents Implicit conversion and assignment to the class

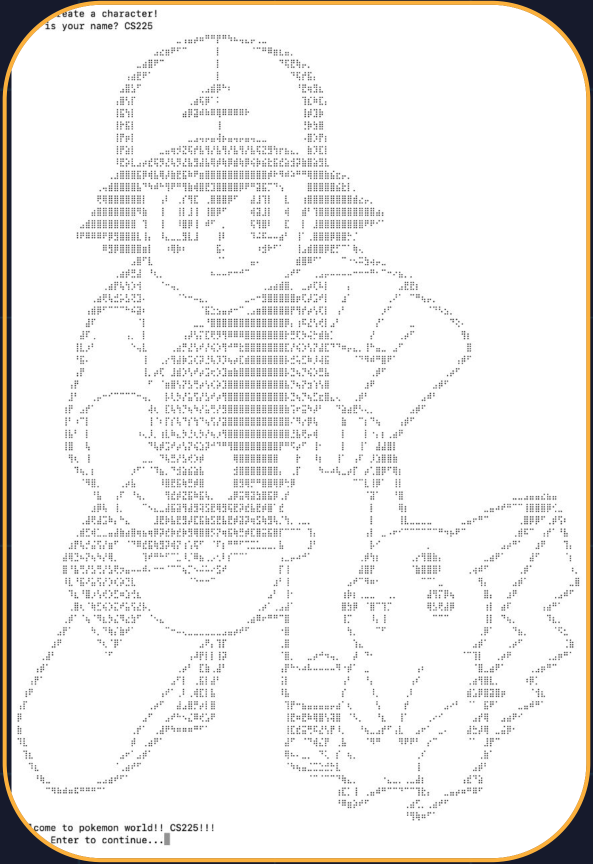


```
class Pikachu : public Pokemon {
public:
    explicit Pikachu(int level = 1, const string &n = "Pikachu")
```

Sample codes

```
#include <iostream>
using namespace std;

class MyClass {
public:
    // Constructor without explicit keyword
    MyClass(int value) : data(value) {}

    int getData() const {
        return data;
    }

private:
    int data;
};

void processObject(const MyClass& obj) {
   cout << "Data: " << obj.getData() << std::endl;
}

int main() {
    MyClass obj1 = 42; // This compiles successfully, but may
lead to unexpected behavior

    processObject(obj1); // This works, but it might not be
what you intended

    return 0;
}
```

```
#include <iostream>
using namespace std;

class Demo{
    public:
        explicit Demo(int n){
            demo1 = n;
        }
        int getDemo(){
            return demo1;
        }
    private:
        int demo1;
};

void getDemoExternally(Demo demo){
    cout << demo.getDemo();
}
// Driver Code
int main()
{
    getDemoExternally(10);
    return 0;
}
```



```
eate a character!
is your name? CS225

come to pokemon world!! CS225!!!
 Enter to continue...
```

06

More Playtime!

Thank You!
Any Questions?

References

ASCII Art Pokemon - asciiart.eu. (n.d.). ASCII Art Pokemon - Asciiart.eu. Retrieved November 20, 2023, from https://www.asciiart.eu/video-games/pokemon

Bing AI image generator. (n.d.). Bing. Retrieved November 17, 2023, from https://www.bing.com/create

Catch rate - Bulbapedia, the community-driven Pokémon encyclopedia. (n.d.). Bulbapedia.bulbagarden.net. https://bulbapedia.bulbagarden.net/wiki/Catch_rate

Chattopadhyay, S. (2022, November 16). *What is Explicit in C++?* Scaler Topics. https://www.scaler.com/topics/cpp-explicit/

DeFreitas, C., McQ, J., Madrigal, H., & Moreupdated, 421. (2012, September 4). *Pokemon Types - Pokemon Gold, Silver and Crystal Guide*. IGN. https://www.ign.com/wikis/pokemon-gold-silver-crystal-version/Pokemon_Types

References

Gym Leader - Bulbapedia, the community-driven Pokémon encyclopedia. (2023, October 29). Bulbapedia.bulbagarden.net. https://bulbapedia.bulbagarden.net/wiki/Gym_Leader

Image to Dot Art Generator (Text Art Maker). (n.d.). Emojicombos.com. https://emojicombos.com/dot-art-generator

Lachlan, A., & Caroline. (n.d.). *Emoji Combos*. Emojicombos.com. Retrieved November 17, 2023, from https://emojicombos.com/

Long, J. (2023, November 17). *prettier/prettier*. GitHub. https://github.com/prettier/prettier

Nintendo. (n.d.). *Pokémon moves: list of attacks*. Pokemondb.net. Retrieved November 17, 2023, from https://pokemondb.net/move/all

OpenAI. (2023, November 2). *ChatGPT*. Chat.openai.com; OpenAI. https://chat.openai.com/

Pokémon Pokédex: list of Pokémon with stats. (n.d.). Pokemondb.net. https://pokemondb.net/pokedex/all

References

Pokémon Sun and Moon IVs explained - how to judge Pokémon stats and get max, 31 IVs in HP, Attack, Defense, Special Attack, Special Defense and Speed IVs. (2017, December 15). *Eurogamer.net*. https://www.eurogamer.net/pokemon-sun-and-moon-competitive-training-guide-how-to-raise-the-best-strongest-pokemon-for-competitive-play-4925?page=3#:~:text=IVs%20(Individual%20Values)%20%2D%20the

Pokémon type chart: strengths and weaknesses. (n.d.). Pokemondb.net. Retrieved November 17, 2023, from https://pokemondb.net/type

Stat - Bulbapedia, the community-driven Pokémon encyclopedia. (n.d.). Bulbapedia.bulbagarden.net. https://bulbapedia.bulbagarden.net/wiki/Stat

The Official Pokémon Website. (2023, November 17). Www.pokemon.com. https://www.pokemon.com/us

References

Welcome to Marriland.com! • Marriland.com. (2023, September 15). Marriland.com. http://marriland.com