

Criterion C - Development

The product is a Java program. It takes in texts from the Pokemon Showdown game and inputs from the user. It saves the Pokemon teams and notes inputted by the user and uses it to process the advice and prediction in the game based on input texts from the game.

-List of techniques-

- File saving
- JSON file reading
- For loop
- While loop
- Arrays
- ArrayList
- Searching (linear search)
- Error handling
- GUI tabs
- Simple and compound selection (if/else)
- Making an array of objects
- User-defined objects made from an OOP "template" class
- Opening a file to a table
- Methods returning a value
- Parameter passing

-Notes-

The user types in the notes about a specific scenario in the game in the note tab. The note will be saved by using a FileWriter variable and .write() method to write a file with data from the note.

Sample Note Text-File

```
1 Xerneas
2 Zygarde
3 Swap to either a full health PDon or a steel type Pokemon. After that just hit it until it dies
4 Arceus, Groudon, Solgaleo, Necrozma-Dusk Mane
5 Groudon, Arceus, Solgaleo
6 Xerneas
7 Probably either Moonblast or Geomancy with power herb
8 Kyogre
9 Kyogre, Rayquaza
```

-Structure of the Program-

The program has a total of 4 classes: Calculator, mainGUI, PokemonSet, and Type. mainGUI uses PokemonSet to store text files of Pokemon teams. Calculator uses Type class to create methods for calculating Pokemon type multipliers, which will be used by mainGUI. The Program was split into classes for better organization.

-Data Structure Used-

Arrays and Arrays of Objects

Arrays are utilized in any part of the program in order to organize both primitive data types and objects in groups, which also allows them to be looped through more easily. In the program, I used arrays to separate the words from a string in order to find a keyword in the text from the game.

Sample use of an array of primitive type to split sentence:

```
//second process, first splitting the words into array by space
String text = inputTF.getText();
String[] words = text.split("\\s+");
String keyword = "started";
//System.out.println(Arrays.toString(words));

//checking the first word
if(keyword.equals(words[1]))
{
```

Sample use of an array of object

```
ArrayList<String> multiplier4 = new ArrayList<String>();
Type[] types = new Type[18];
double[] normalMultiplier = {1,1,1,1,1,1,2,1,1,1,1,1,1,0,1,1,1,1};
types[0] = new Type("Normal",normalMultiplier);
```

ArrayList

In this program, ArrayList is used to contain and organize all the types of Pokemon calculated from the calculator class. It's simpler and more straightforward to just use ArrayList, which have a dynamic size, to store these types as opposed to using an array, which have a fixed size.

Text Files

Text files were used as a means to save the recorded Pokemon teams and notes. An example of saved notes can be seen above.

JSON Files

The information about every Pokemon (base stats and types) are saved in a JSON file downloaded from the internet and is accessed by the mainGUI and PokemonSet class in order to gain information about each Pokemon. It was done by initializing a JSON object as the JSON file, then get the value out of that JSON object.

Sample code using JSON object to access information from the JSON file:

```
//Creating a JSON variable
JSONObject pokemonName = (JSONObject) pokedex;

//System.out.println(pokemonName.get("name"));

JSONObject pokemonName = (JSONObject) pokedex.get("name");
//System.out.println(pokemonName.get("name").equals(name));

if(pokemonName.get("name").equals(name)){
```

-Main Unique Algorithm-

Type Calculator Algorithm

The algorithm of the type calculator methods works by using Types object class, which have attributes of a String of type name and an array of 18 doubles, which contains the multipliers of that specific type as it interacts with another type. Then, the method creates an array of all 18 types, with each element as a specific type with name and type modifier array initialized. Each method will take in varying amounts of String parameter which are names of the type in Pokemon.

The general algorithm of every type calculator method is as follow:

```
public static ArrayList<String> typeCalc1(String first, String second){
    ArrayList<String> multiplier4 = new ArrayList<String>();
    Type[]types = new Type[18];
    double[]normalMultiplier = {1,1,1,1,1,1,2,1,1,1,1,1,1,0,1,1,1,1};
    types[0] = new Type("Normal",normalMultiplier);
    double[]fireMultiplier = {1,0.5,2,0.5,1,0.5,1,1,2,1,1,0.5,2,1,1,1,0.5,0.5};
    types[1] = new Type("Fire",fireMultiplier);
    double[]waterMultiplier = {1,0.5,0.5,2,0.5,1,1,1,1,1,1,1,1,1,1,0.5,1};
    types[2] = new Type("Water",waterMultiplier);
    double[]grassMultiplier = {1,2,0.5,0.5,0.5,2,1,2,0.5,2,1,2,1,1,1,1,1};
    types[3] = new Type("grass",grassMultiplier);
    double[]electricMultiplier = {1,1,1,1,0.5,1,1,1,2,0.5,1,1,1,1,1,0.5,1};
    types[4] = new Type("Electric",electricMultiplier);
    double[]iceMultiplier = {1,2,1,1,1,0.5,2,1,1,1,1,1,2,1,1,1,2,1};
    types[5] = new Type("Ice",iceMultiplier);
    double[]fightingMultiplier = {1,1,1,1,1,1,1,1,2,2,0.5,0.5,1,1,0.5,1,2};
    types[6] = new Type("Fighting",fightingMultiplier);
    double[]poisonMultiplier = {1,1,1,0.5,1,1,0.5,0.5,2,1,2,0.5,1,1,1,1,0.5};
    types[7] = new Type("Poison",poisonMultiplier);
    double[]groundMultiplier = {1,1,2,2,0,2,1,0.5,1,1,1,1,0.5,1,1,1,1,1};
    types[8] = new Type("Ground",groundMultiplier);
    double[]flyingMultiplier = {1,1,1,0.5,2,2,0.5,1,0,1,1,0.5,2,1,1,1,1,1};
    types[9] = new Type("Flying",flyingMultiplier);
    double[]psychicMultiplier = {1,1,1,1,1,1,0.5,1,1,1,0.5,2,1,2,1,2,1,1};
    types[10] = new Type("Psychic",psychicMultiplier);
    double[]bugMultiplier = {1,2,1,0.5,1,1,0.5,1,0.5,2,1,1,2,1,1,1,1,1};
    types[11] = new Type("Bug",bugMultiplier);
    double[]rockMultiplier = {0.5,0.5,2,2,1,1,2,0.5,2,0.5,1,1,1,1,1,1,2,1};
    types[12] = new Type("Rock",rockMultiplier);
    double[]ghostMultiplier = {0,1,1,1,1,1,0,0.5,1,1,1,0.5,1,2,1,2,1,1};
    types[13] = new Type("Ghost",ghostMultiplier);
    double[]dragonMultiplier = {1,0.5,0.5,0.5,0.5,2,1,1,1,1,1,1,1,2,1,1,2};
    types[14] = new Type("Dragon",dragonMultiplier);
    double[]darkMultiplier = {1,1,1,1,1,1,2,1,1,0,2,1,0.5,1,0.5,1,2};
    types[15] = new Type("Dark",darkMultiplier);
    double[]steelMultiplier = {0.5,2,1,0.5,1,0.5,2,0,2,0.5,0.5,0.5,0.5,1,0.5,1,0.5,0.5};
    types[16] = new Type("Steel",steelMultiplier);
    double[]fairyMultiplier = {1,1,1,1,1,1,0.5,2,1,1,1,0.5,1,1,0,0.5,2,1};
    types[17] = new Type("fairy",fairyMultiplier);
    if(!(second.equals("xdl"))){
        boolean type1Found = true;
        int type1Index = 0;
        int i = 0;
        while(type1Found){
            if(first.equalsIgnoreCase(types[i].getTypeName())){
                type1Index = i;
                type1Found = false;
            }
            i++;
        }
        boolean type2Found = true;
        int type2Index = 0;
        i = 0;
        while(type2Found){
            if(second.equalsIgnoreCase(types[i].getTypeName())){
                type2Index = i;
                type2Found = false;
            }
            i++;
        }
        double[] type1 = types[type1Index].getTypeMultiplier();
        double[] type2 = types[type2Index].getTypeMultiplier();
        for(int o=0;o<types.length;o++){
            if(type1[o]*type2[o]==1){
                multiplier4.add(types[o].getTypeName());
            }
        }
    }
    return multiplier4;
}
```

Setting the Array of types and the type multipliers

Finding the index of each type

Getting typeMultiplier Array

Loop through and add up all the types

Team and Notes Saving Algorithm

The algorithm involves getting String values from text fields and write them into a file using FileWriter variable. The file was written in a specific order that, when the team was searched up again, the file will be read and the value will be taken and shown in the correct order in a table.

```
JSONObject pokemonName = (JSONObject) pokedex;
String pokeName = (String) pokemonName.get("name");
//System.out.println(pokemonName.get("name"));

//Get employee object within list
JSONObject pokemonName = (JSONObject) pokedex.get("name");
//System.out.println(pokemonName.get("name").equals(name));

if(pokeName.equalsIgnoreCase(name)){

    try {
        FileWriter myWriter = new FileWriter("./src/Filedata/"+ name + ".txt");

        //System.out.println(pokemonName.get("name"));
        //System.out.println(pokemonName.get("types"));
        //System.out.println(pokemonName.get("base_stats"));
        myWriter.write(name);
        myWriter.write("\n"+ Arrays.toString(moves));
        myWriter.write("\n"+ item);
        myWriter.write("\n"+ ability);
        myWriter.write("\n"+ pokemonName.get("types"));
        myWriter.write("\n"+ pokemonName.get("base_stats"));
        myWriter.write("\n"+ pokedex.get("type"));

        JSONObject pokemonBase = (JSONObject)pokedex.get("base");

        myWriter.write("\n"+pokemonBase.get("HP"));
        myWriter.write("\n"+ pokemonBase.get("Attack"));
        myWriter.write("\n"+ pokemonBase.get("Defense"));
        myWriter.write("\n"+ pokemonBase.get("Sp. Attack"));
        myWriter.write("\n"+ pokemonBase.get("Sp. Defense"));
        myWriter.write("\n"+ pokemonBase.get("Speed"));
        myWriter.close();

        JOptionPane.showMessageDialog(null, "Add Pokemon success!");
        pokemonTF.setText("");
        move1TF.setText("");
        move2TF.setText("");
        move3TF.setText("");
        move4TF.setText("");
        itemTF.setText("");
        abilityTF.setText("");
        errorTF.setText("");
        setOk(false);
    }

}
```

Create a JSON object to access the JSON file

If found the Pokemon in the JSON file, save its data in text files

Reset the text fields

Move Prediction Algorithm

The algorithm is a pile of if/else statements with conditions of checking the stats, types, and moves of both opponent's and user's Pokemon on the field. The Pokemon on the field are detected through reading the input text from the game. The stats of the Pokemons are taken from a JSON file downloaded from the internet and found through searching for the Pokemon name in the file.

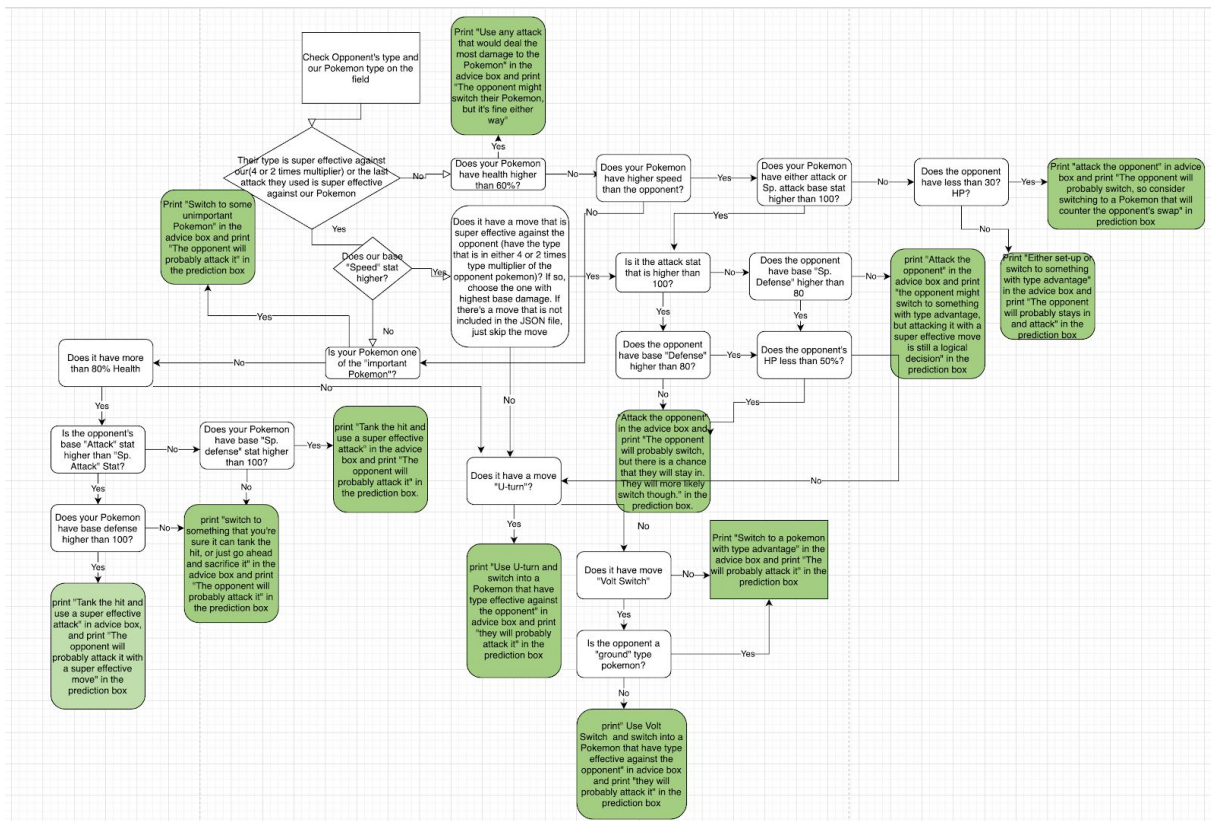
Sample code of reading JSON file and searching for content with a specific keyword (Pokemon name):

```
JSONParser jsonParser = new JSONParser();
try (FileReader reader = new FileReader("pokemon.json"))
{
    //Read JSON file
    Object obj = jsonParser.parse(reader);

    JSONArray pokemon = (JSONArray) obj;

    pokemon.forEach( item -> {
        JSONObject pokemonName = (JSONObject) item;
        String name = (String) pokemonName.get("name");
    });
}
```

The move prediction follows this flow chart:



-GUI Works-

Text Fields

Text Fields are used in 2 ways in this program: to display output texts(notes and advice) and to take in the input text.

Predicted Opponent's Next Move

Your Advised Next Move and Reminder

Opponent's Pokemon Possible Moves

Pokemon 1

Your Pokemon's Weakness

Opponent's Pokemon's Weakness

Input Texts

Paste Input

Buttons

Buttons are implemented to give the user an ability to make a process occurs.

Radio Buttons

Radio buttons are implemented to give the user an ability to choose between 2 choices on how the Notes tab functions.

Process Input

Start Game

End Game

Search by Pokemon Name

Search by Note Name

Opponent's Pokemon Stats

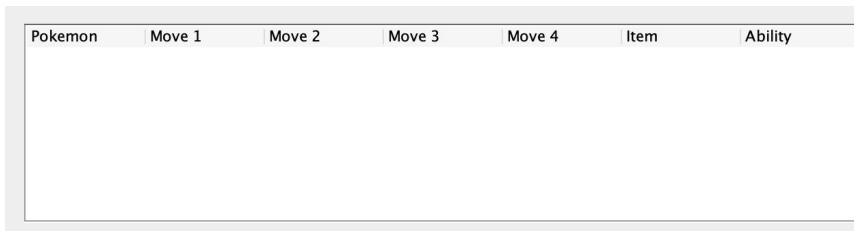
Opponent's Pokemon Stats

Combo Boxes

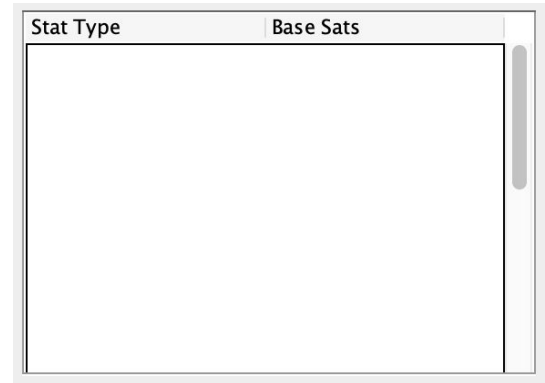
Combo boxes are used to allow the user to choose between multiple options, such as choosing Pokemon to view stats.

Tables

Tables were used to display information about a Pokemon team and base stats of the opponent's Pokemon.



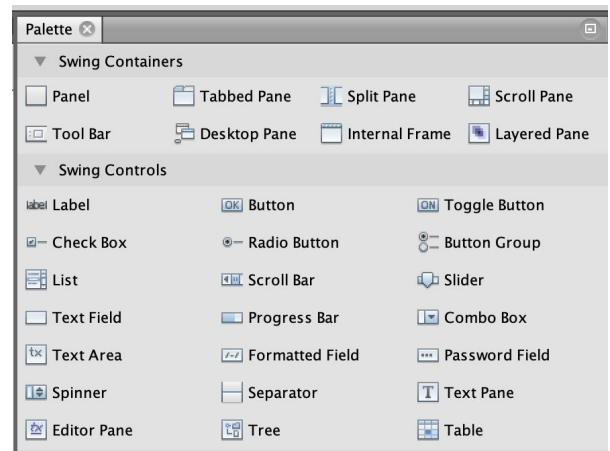
Pokemon	Move 1	Move 2	Move 3	Move 4	Item	Ability
---------	--------	--------	--------	--------	------	---------



Stat Type	Base Stats
-----------	------------

-Software Tool Used-

The software tool used is the Netbeans IDE 8.2. It's one of the more popular Integrated Development Environment(IDE) for Java. It allows the programmer to code in Java, organizes the OOP hierarchy and files, and easily create GUI for the program, as seen in the pre-packaged GUI components shown on the right.



Word Count: 848