

**Technical Report**

**WIX1002**

**Fundamentals of Programming Assignment**

**Semester 1 2018/2019**

Task A: Pokemon World Simulator

Our company, a regional software developer has received a project called “Pokemon World Simulator” this month for a local hospital. The project is important, with certain level of technical difficulties and challenges. We have been instructed by the CEO to lead a team of 4-5 members, to handle the project and given 6 weeks to complete the project as well as prepare a demo for the board of directors

TASK

We are assigned to write a program to create our own Pokemon World. We are required to invent our own virtual world with certain rules, creatures, cities and even civilizations.

TASK REQUIREMENT

In this Pokemon World Simulator, we are required to make a pokemon 1 vs 1 combat system. Before running the simulator, some figures must be full-filled to enable a similar virtual Pokemon World as the real Pokemon Game:

1. Pokemon selection

User ​must be given choice to choose 3 pokemons they wanted meanwhile bot’s 3 pokemons are chosen ramdomly. Each pokemons is categorized based on the elements fire, leaf and water respectively. The pokemons are given 4 different skills that can be used during 1 vs 1 combat which the user chooses any skill they wish to use.

1. Attack rules

The attack order is speed-based which is calculated based on Pokemon Speed. Pokemon that reaches accumulated value of 100 will attack​​ first and the accumulator will recalculate again but not reset. It is totally possible for a pokemon to attack two times in a row.

For better understandings, refer to the example below:

Pokemon A speed: 40

Pokemon B speed: 50

Speed accumulator for Pokemon A: 0

Speed accumulator for Pokemon B: 0

First loop:

Speed accumulator for Pokemon A: 0 + 40 = 40

Speed accumulator for Pokemon B: 0 + 50 = 50

No pokemon will attack

Second loop:

Speed accumulator for Pokemon A: 40 + 40 = 80

Speed accumulator for Pokemon B: 50 + 50 = 100 - 100 = 0

Pokemon B will attack and the Speed accumulator for Pokemon B minus 100 because of attacking

Third loop:

Speed accumulator for Pokemon A: 80 + 40 = 120 - 100 = 20

Speed accumulator for Pokemon B: 0 + 50 = 50

Pokemon A will attack and the Speed accumulator for Pokemon A minus 100 because of attacking.

1. Skill accuracy

There are chances of missing attacks due to accuracy. For example, a skill with skill accuracy of 70% will have 30% of missing attacks. For sample input of skill accuracy can refer to the sample input section.

The program repeats until user prompts to exit.



1. Damage calculation

Damage = (((Pokemon attack \* Skill’s power / Opponent’s pokemon defense) / 20) + 2) \* Damage Multiplier

The Damage Multiplier is the skill type advantage or disadvantage when attack other pokemon. For example, if fire type move hits a grass pokemon, it will double the damage, if fire type move hits a water pokemon, the damage will be halved. If the move type is same as pokemon type or either move type or pokemon type is normal type, the damage will remain the same.

APPROACH

We designed a virtual simulator that enable user to experience the Pokemon World. It started with several pokemons that further detailed with names, skills and elements. We had provided a total of 12 different types of pokemon to be run in the tester output. The user is given priority to choose 3 pokemons first and the bot’s 3 pokemons are randomly generated.

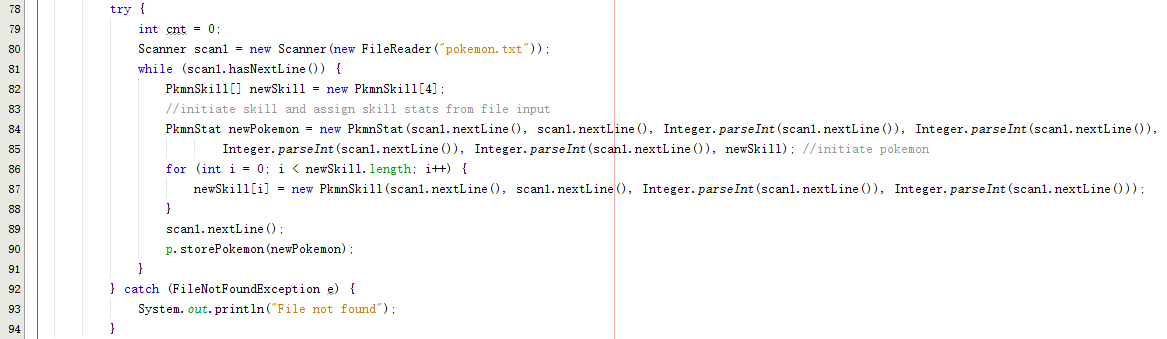
During the combat, we had set that the user is allowed to select 1 of the skill out from 4 skills. The damages are calculated by using the Damage formula stated previously and added on with damage multiplier. The damage multiplier is set based on types of element during the combat among pokemons, which fire element overwhelmed grass element, grass element overwhelmed water element, and water element overwhelmed fire element. This setting is important which no any element is outstanding and the equality of game experience is maintained.

Besides attacking, we also provide the pokemons to dodge during the combat. We had set up the skill accuracy on all pokemons when they having chance to attack. This measure makes some difficulties in the game which not just skills and elements are important, but as well as the luck of the user. When the skill accuracy is higher, damage is dealt, meanwhile, when the skill accuracy is lower, the damage dealt is missed which results no deduction in the hit points of the pokemons.

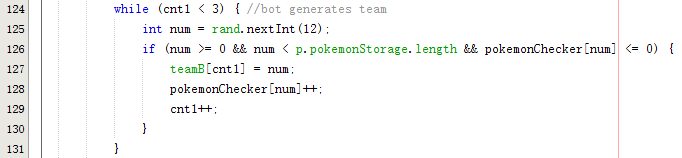
In order to provide more gaming experience to the user, we had also gone through the Pokemon Game by our self. We are exposed to even more features that can be added in our simulator to improve our interactions between our virtual Pokemon World Simulator and the user.

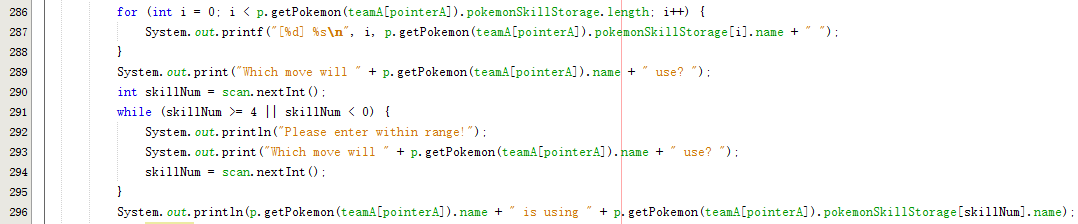
SOLUTION DESCRIPTION

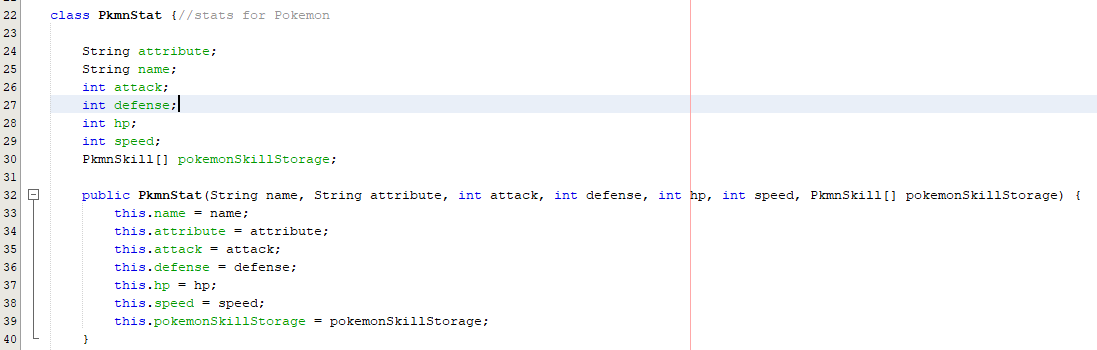
In our source code, we have read the pokemon text file given in order to get the pokemon and their attribute.



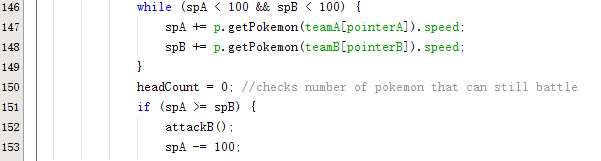
We had created 2 classes which are pokemon stats class and the main class where the codes will run. For our bot to choose pokemon, we generate 3 not duplicated random number within the range of the array of pokemon.

A list of different pokemons are important as during the 1 vs 1 combat, the user would not get bored in our simulator. The more the types of pokemon available, the more interesting our simulator will be. Besides than the pokemon types, we also inserted the method to choose the skills the user prefers the most.



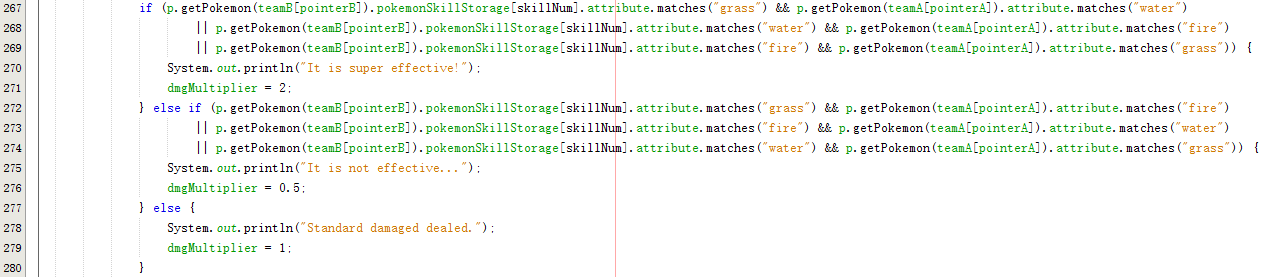
This method is significant because the skills would not generate randomly and at the same time giving user to have more interaction space with the simulator. In pokemon stats class, we added the method where basic stats of each pokemons are given and at the same time there would not exist any overwhelming stats that can cause any imbalance combat platform.

For the pokemon battle, we created the attack rule that similar to the example given previously which is based on speed. The pokemon are only given the chance to attack first whenever the speed accumulator reached 100. The value is then minus by 100 so that it has to wait for other time, when speed accumulator reach 100 again, to have the next attack in the simulator.

 Moreover, we also included some tricky aspect which it enables pokemon to dodge the damage dealt during the combat. We generate 100 number from 1 to 100 and when the number generated is higher than skill accuracy, damage dealt will turn to zero and miss totally.



The battle method also included the damage dealt calculation based on the type of element used. As mentioned before, element that overwhelmed other will cause double damage, same element causes no extra damage and otherwise, cause the damage dealt become half. This conditional damage calculation enhances the gaming experience further as user are needed to think twice on the elements used during combat.



Flowchart

Start

User choose 3 pokemons

Bot generate 3 pokemons

User choose skill

Bot generate skill

accumulatorA+=skillA

accumulatorB+=skillB

When accumulatorA >=100

When accumulatorB>=

100

no

no

yes yes

randNum>accuracy

randNum>accuracy

yes yes

no

no no

Press Y to chane pokemon

Press Y to chane pokemon

accumulatorA-=100

hpB-damagedelt\*damagemultiplier

accumulatorB-=100

hpA-damagedelt\*damagemultiplier

no no

hpB==0

yes

yes

hpA==0

A win

B win

TECHNICAL DIFFICULTIES

1. Collaboration difficulties

At find it difficult to work together and share code. We cannot code at the same time as it will be redundant hence we resort to coding to coding parts by parts with is not a very good practice as they can hardly link together. We get to solve this problem by resorting to a code repository platform known as GitHub. This allows us to collaborate with each easily. Other than that, we make sure we maintain readable code to make sure everyone of use understands every function of the code by using appropriate variables and inserting comments for every function in the main method. There were learning curves on GIT commands among our teammates but managed to pick it up in the end.

1. Coding for graphic user interface

As the first year students without programming based, it become a huge task for us to create a GUI which enables interaction between user and the simulator. We are needed to attend to online tutorials on how to create the GUI and took almost a week of time just to produce a simple GUI. Several series of tutorials re needed to go through before coming out the result that we had created.

EXTRA FEATURES

1. Graphic user interface

We had added a graphic User Interface that enables user to have interaction with our simulator. The GUI we had created is a two dimensional (2D) only which is

much more simple.

1. Movable character

In our GUI, we had also designed a character that can move within the 2D interface. We further adding some features which whenever our avatar met with other avatars, it trigger the battle mode which is 1 vs 1 combat.

1. Audio and sound

To prevent the user from getting bored, we included audio and sound in the GUI which make the interface more interesting to play. We target to create a chilling atmosphere in our simulator.

LIMITATION

1. The types of pokemon is limited

This is stated as limitation as we had few time in creating this source code. We did not manage to study all of the types available. If the variety of the type of pokemon can be increased then it would have a better gaming experience to the user as when more pokemon types, more pokemon skills.

1. Exist of one element in one pokemon

As it is still a starting simulator, we could only manage to set each pokemons with only one elements. The combination of two or more elements in pokemon would make the coding more complex and may alter the output of the source code.

1. Unable in continuous combat

In our simulator, it could be only have once combat which involving 3 pokemons of the user battle with the other 3 pokemons from the bot. once the battle finished and the winner shown up, the simulator will straight away close the file and user cannot continue to have next battle.