Chapter 3 Battle Programming Project

**Content**

**Elements for interface design that is efficient and intuitive for the user**

As you begin to receive more input from the user, and generate more output for the user to read, it’s important to think carefully about how the user will interact with your program. Sometimes instructions can be unclear for the user and the programmer doesn’t realize it.

Be sure to ask specific, detailed questions and be sure to format and display output in a way that the user can easily understand. One of the best ways to determine whether or not your program is user-friendly is to have a classmate, friend or relative use your program. As you watch them use it, take notes about how they interact with the program and about whether or not they are confused about any input our output.

Complete this programming project using your notes, the text book, and any online or in-class sources you like. Your work must be your own; you may ask a friend to look over your work, or discuss procedural decomposition with you, but you must write all code on your own. To receive full credit on this lab, you must submit a structure diagram or pseudocode-plan for each question.

Recall how to use Scanner to get user input:

Scanner console = new Scanner([System.in](http://system.in));

System.out.print("Hello, what is your name? ");

String name = console.nextLine();

System.out.print("What is your age? ");

int age = console.nextInt();

Exercise 1 - Introduction

*Write a method called* battleStart() *that introduces the battle, prompts the user to choose their first Pokemon to battle, and outputs the pairing.* battleStart()*should also return the name of the Pokemon chosen. Your output should look something like this:*

Another trainer is issuing a challenge!

Zebstrika appeared.

Which Pokemon do you choose? *Arcanine*

You chose Arcanine!

It’s a Pokemon battle between Arcanine and Zebstrika! Go!

*Call* battleStart() *from your* main() *method and store the name of the Pokemon in a variable.*

Exercise 2 - Battle

*Write a method called* damage() *that takes a Pokemon’s name as a parameter and returns the about of HP after damage has been done.* damage() *should prompt the user for their base stats in order to calculate damage. Use the following equations for calculating damage:*

*Damage =*

*Modifier = Same Type Attack Bonus (STAB) \* Random*

*Hint: The Pokemon game always selects a random number between 0.85 and 1.0.*

*Your output should look like this:*

Zebstrika used Thunderbolt!

Trainer, what are your Arcanine’s stats?

Level:

Attack:

Defense:

Base:

STAB:

HP:

Arcanine sustained 10 points damage.

HP, after damage, are now 70.

*Call* damage() *from your* main() *method with the Pokemon’s name from Exercise 1 and store the return value (HP) in a variable.*

Exercise 3 - Summary

*Write a method called* statsTable() *that accepts the user’s Pokemon name, stats and learned moves as parameters, and outputs something similar to this image:*

**

*Sample output:*

Name Alakazam

Level 40

------------------------------

HP 96

ATTACK 52

DEFENSE 51

SP. ATK 121

SP. DEF 81

SPEED 107

-------------------------------

Moves Learned: Thunder Wave, Hidden Power, Psycho Cut, Recover

*Call* statsTable() *from your* main() *method with the Pokemon’s name from Exercise 1 and the HP from Exercise 2 and any other values you’d like for the other parameters.*

Conclusion

In y*our completed project should include the following methods:*

* + battleStart()
  + damage()
  + statsTable()

*These methods should all be called in* main()*so that the player can experience the entire battle in one sitting. Also be sure to close your* Scanner *when finished.*

*Proper Java syntax and thorough comments are required.*

**Curricular Competencies – Applied Design**

**Understanding Context**

**Testing**

One of the best ways to effectively design and develop for end-users is to include them in the design and development processes. Computer programmers do this by interviewing the end-users at the beginning of the process. This allows the programmer to understand what the user is looking for and it helps create an initial plan for design ideas.

When it’s time for the program to be tested, it is often good practice to have the end-user test the software. This allows them to comment and provide feedback on different parts of the program before it is totally complete. The programmer should take notes related to the end-user’s questions and concerns about the program, and also the things that they love.