TEALS Program

1 Lesson 3.07 — Pokémon Battle Programming Project

1.1 Overview

1.1.1 Objectives — *Students will be able to…*

- **Write** a program that requests user input and returns data.

1.1.2 Assessments — *Students will…*

- **Write** a program that calculates damage done to Pokémon in a battle.

1.1.3 Homework — *Students will…*

- **Summarize** their class notes since the last exam
    - If they are missing notes, get them from another student or supplement them from the textbook

1.2 Materials & Prep

- **Projector and computer**
- **Whiteboard and markers**
- **Classroom copies** of WS 3.7 LP Battle
- **Video** of sample battle (http://youtu.be/k7k5lee9xxw?t=48s)
- **Advanced damage calculator** (https://pokemonshowdown.com/damagecalc)

The 8-minute video demonstrates a typical battle sequence from one of the more recent Pokémon versions. If your class does not play the video game, you could show battle footage of the anime series, the card game, or the coin game. As the instructor, you should familiarize yourself with the sequence of a Pokémon battle, so you can help students with procedural decomposition and grade different student solutions.

1.3 Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction to lab & viewing of battle	10min
Student programming practice	40min

1.4 Procedure

The programming project today has students programming a "starter" Pokémon battle sequence. This is a somewhat open-ended assignment, since students can submit a basic program that runs 1 or 2 interactions, or a complete battle sequence, depending on their level of understanding.

Student programs for this assignment will be a lobotomized version of a Pokémon battle since students have not yet learned conditional statements. This is a deliberate move: students can focus on building segments of code that accept basic user input, use the math class to generate random numbers to determine battle outcomes (or roll-of-the-dice or spin for the card-game and coin game versions), and return game text. Capitalize on student frustration by (or motivate students with the prospect of) hinting at a more interactive program after the next few lessons on Ch. 4 and Ch. 5.

1.4.1 Bell-work and Attendance `[5 minutes]`

1.4.2 Introduction to Lab and Viewing of Battle `[10 minutes]`

At the beginning of class, introduce the lab and watch the sample battle video.

1.4.3 Student Programming Practice `[40 minutes]`

1. Have students complete this programming project individually. Before you break out the class for lab time, read the question out loud to the class, taking time to pause between each of the requirements outlined in the lab assignment.

2. Ask students what their very first steps should be.

    - They should outline their approach in pseudocode or with a structure diagram. Remind them that this documentation should be submitted in order to get full credit for their lab, and refer them to the Algorithm for Solving Problems sheet.

    - Remind students to tackle one part of the problem at a time. Remind students that it is OK if they leave pseudocode in while they solve a different part of the problem, and partial credit should be given to correct pseudocode.

3. To encourage grit, have students review the steps they should take before raising their hand for a question:
    - Refer to notes, textbooks, and posters/displayed work around the room.
    - Work on a different part of the problem if they get stuck, then return to it later.
    - Ask another student for a hint, tip, or for error-spotting.

4. In an email, on the projector, or as a handout WS 3.7, give student the following questions to work on individually (or, if scaffolding requires it, in pairs).

1.5.0.1 Emphasize with students...

1.5.0.2 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.

1.5.0.3 PROGRAMMING PROJECT

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);
System.out.print("Hello, what is your name? ");
String name = console.nextLine();

System.out.print("What is your age? ");
int age = console.nextInt();
```

1.5.0.3.1 Exercise 1

Write a method called battleStart() that introduces the battle, prompts the user to choose their first Pokémon 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 Pokémon do you choose? Arcanine

You chose Arcanine!

It's a Pokémon battle between Arcanine and Zebstrika! Go!

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

1.5.0.3.2 Exercise 2

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:

Modifier = Same Type Attack Bonus (STAB) * Random Damage = Modifier * ((2*Level+10)/250 + (Attack/Defense)*Base + 2)

Hint: The Pokémon 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.

1.5.0.3.3 Exercise 3

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:

You are not required to align the columns of the tables in any fancy way, but if you do, use escape sequences to align data. For your drawing, you may use code you've grabbed from the internet, or recycle an image you created earlier in the year.

Sample output:

Name Alakazam

1.6 Level 40

HP 96 ATTACK 52 DEFENSE 51 SP. ATK 121 SP. DEF 81

1.7 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.

1.7.0.0.1 Conclusion

In your 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.

Proper Java syntax and thorough comments are required.

1.7.0.1 Emphasize with students...

1.7.0.2 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 about the things that they love.

1.7.0.3 Final Project

Imagine you had to create a program similar to the one above, but instead of using Pokemons it used insects, dinosaurs or basketball teams. What initial stats would each dinosaur have? What attacks would insects have? How would you program the ability to have one basketball team challenge another basketball team?

The final project in this course will provide you with the opportunity to create a program, similar to Pokemon, that uses different characters or "challengers". As you read more about the Pokemon program, think carefully about how you would implement the same ideas in a different context.

1.8 Accommodation and Differentiation

If you have students finish the lab quickly, invite them to check out the advanced damage calculator online. They can add input fields to their own damage calculator, thus improving their Pokémon simulation.

Alternatively, if students seem interested in increasing the interactivity of their battle sequence, you can allow them to read ahead in the book. Students may read up on Bulbapedia (the wiki for Pokémon) that Thunderbolt has a 10% chance of paralyzing its' target. Invite students to think how to add that factor into their battle simulation as they read through Ch. 4 and Ch. 5 materials.

If students are struggling with creating the graphic in Exercise 3, help students by writing helper lines of code on the board, or creating a pseudocode outline as a whole group.

1.9 Forum discussion

Lesson 3.07 Pokémon Battle Programming Project (TEALS Discourse account required)