CSCI 2120

Homework Two – Class Pokemon: Javadoc and JUnit

Commenting and testing your code is extremely important – your code is only as good as it is provably correct, and usable. For this assignment, you will be given a class that represents a Pokemon – you'll be asked to finish commenting this code with detailed and extensive Javadoc comments, specify each methods pre- and post-conditions in the Javadoc, and finally write a JUnit tester that proves that all aspects of the Pokemon class behave as advertised.

Submission:

Create a new directory within your repository called HW2. Modify the given Pokemon.java and write a JUnit tester called PokemonTester.java. As you write the program, add, commit, and push your files to the remote repository on the GitLab server. When you have finished the assignment, make sure you've uploaded the most up to date version of your files (check the website to see).

Assignment Statement:

This assignment will consist of two parts:

1. A class that defines an object for a Pokemon and **this class will be fully documented using doc comments** - see https://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html for an example of the how your class should be documented.
2. A JUnit tester to show that your code performs exactly as one would expect - I will expect extensive testing and that the JUnit tester also has documentation explaining the point of each test.

Writing class Pokemon's Javadoc:

In the Pokemon.java file you'll notice that the first few methods and instance variables already have extensive Javadoc documentation – expand this level of documentation to include the rest of the methods of Pokemon.

Writing the JUnit Tester

In a separate file, PokemonTester.java, implement a JUnitTester for your Pokemon class using JUnit 5. You should provide at least one @Test method for each of the public methods in Pokemon. Each test method should verify the correctness of the method it is testing using multiple Pokemon instances. Compile and run the test to verify that all your tests pass. Be sure to test for a wide variety of input combinations to ensure adequate testing. Take a look at the JUnit documentation online to discover new interesting testing techniques.

Tip: when testing your program to make sure your calculations are correct, use a pencil and paper, or a calculator to verify your calculations.

Due Date

You need to complete this assignment and have it pushed to the remote repository on the Git server by the date/time specified on Moodle.

There should be a README.txt file stating exactly how to compile, run, and test your code. Bonuses, if they have been assigned and you chose to do them, should be separate (in subdirectories labeled bonus1, etc.) and fully complete implementations beyond the required one (in other words, start working from a fully implemented original copy in the subdirectory to do the bonus).