Homework 09 - Wizard's Duel

Authors: Allan Nguyen, Tushna Eduljee, Emma Barron, Ryan Miles, Manny Sonubi, Shishir Bhat

Problem Description

You're a muggle programmer at Hogwarts and you are a big fan of the Duelling Club. As a fan, you enjoy betting and predicting the outcome of the duels. Dumbledore asks you to create a duel simulator to predict the winner of a matchup and to bet appropriately! Using some of your own coding "magic", you decide to create a class hierarchy that represents the wizards around you.

Solution Description

For this assignment, create files with the following names:

- `Wizard.java`
- `Gryffindor.java`
- `Slytherin.java`
- `Spell.java`

Please make sure to use descriptive variables and javadocs as we will be manually grading most of this assignment. You don't have to make getters and setters for all variables, just where you think is needed!

You must use all of the best practices that we have learned in class.

Wizard Class:

Description:

- A wizard has a name, health, max health, attack power, favorite color, and spell book.
- We can create a wizard if we know their name, favorite color, max health, attack power, and spell book.
  - If we don't know their max health, let it be 25.
  - All wizards have a spell book, and the default spell book has only one spell in it:
    - Wingardium Leviosa: 0 attack power
  - Wizards' health should be at max when they are created!

Functionality:

- Wizards can cast spells.
  - A wizard will randomly choose one spell from their spell book.
- When you print a wizard, we get an output:
  - `"[name] wishes to join the [color] house at Hogwarts! They have [current health] health points and [attack power of wizard] attack power."`
- Two wizards should be able to duel each other. Because they are having a Wizard's duel, this method should be called on the Wizard class, not something an individual wizard does.
  - The duel continues until one of the wizards' health falls to 0.
    - Wizards will deal damage to their opponent.
      - The amount of damage is attack power of wizard + attack power of spell
    - Each round before the attack phase, There is a 20% chance that the wizard with less health will gain 3 health.
      - Print `"[name] drinks an invigoration potion and restores 3 health! [name] now has [health] health."`

        - Wizard's health should not be able to go above max health.
        - If the two wizards have equal health, the potion is not drunken.
      - After taking the potion, the wizard with less health should attack first, then the other wizard attacks back.
        - If a potion is drunk, check wizard health again. The wizard with the least amount of health should always attack first. This means the wizard who takes the potion may be different from the wizard who attacks first.
        - If the two wizards have equal health, you can choose the wizard that attacks first.
      - When each wizard attacks,
        - Print `"[name] casts [spell name] and deals [damage] damage. [other name] now has [other wizard health] health."`
      - Health cannot fall below 0 at any point.
        - If a wizard's health falls to 0, print `"[name] falls to the ground. [other name] wins the duel!"`
      - After the duel, restore both wizards' health to their max health.
- Wizards can interact with another wizard.
  - prints `"[name]: Hey [other name], let's be friends!"`
  - Duel will commence between two wizards.
- Correctly decides if two wizards are equal based on: name, health, attack power, color, and spell book.

Gryffindor Class:

Description:

- Gryffindors are also wizards, so they should be able to have/do everything a normal wizard can do!
- We can create a Gryffindor wizard if we know their name.
- Gryffindors focus on resilience and therefore have more health.
  - Max health is a random number between 25 to 30 (inclusive).
  - Base attacking power is a random number between 4 to 6 (inclusive).
- Gryffindors love the color scarlet.
- Gryffindors have a special spell book given by Dumbledore and can cast the following spells:
  - Expecto Patronum: 3 attack power
  - Expelliarmus: 2 attack power
  - Ridikkulus: 1 attack power
  - Wingardium Leviosa: 0 attack power

Functionality:

- Interacts with another wizard:
  - prints `"[name]: Hey [other name], let's be friends!"`
  - Duel will commence between two wizards.

Slytherin Class:

Description:

- Slytherins are also wizards, so they should be able to have/do everything a normal wizard can do!
- We can create a Slytherin wizard if we know their name.
- Slytherins focus on self preservation and therefore have stronger attacking power.
  - Max health is a random number between 22 to 27 (inclusive).
  - Base attacking power is a random number between 5 to 7 (inclusive).
- Slytherins love the color green.
- Slytherins have a special spell book with some dark arts and can cast the following spells:
  - Expulso: 3 attack power
  - Levicorpus: 2 attack power

    - Oppugno: 1 attack power
    - Flipendo: 1 attack power
    - Wingardium Leviosa: 0 attack power

Functionality:

- Interacts with another wizard:
    - prints `"[name]: Hey [other name], let's be friends!"`
    - Duel will commence between two wizards.

Spell Class:

Description:

- Each spell should have a name and a damage associated with it.
- When you print a spell it will output:
    - `[spell name]: [damage] attack power`
- Compares two spells' name and damage and returns a boolean depending on if the spells are the same.

Note: Each method requires proper JavaDocs (see below for more information)

Testing your code:

Feel free to create your own tests in the `main` method! Here is an example printed output from a duel:

- Harry Potter is a Gryffindor with 30 health, 6 attack power.
- Draco Malfoy is a Slytherin with 27 health, 7 attack power.

```
Draco Malfoy casts Expulso and deals 10 damage. Harry Potter now has 20 health.
Harry Potter casts Expelliarmus and deals 8 damage. Draco Malfoy now has 19 health.
Draco Malfoy drinks an invigoration potion and restores 3 health! Draco Malfoy now has 22 health.
Harry Potter casts Expecto Patronum and deals 9 damage. Draco Malfoy now has 13 health.
Draco Malfoy casts Levicorpus and deals 9 damage. Harry Potter now has 11 health.
Harry Potter casts Expecto Patronum and deals 9 damage. Draco Malfoy now has 4 health.
Draco Malfoy casts Oppugno and deals 8 damage. Harry Potter now has 3 health.
Harry Potter casts Ridikkulus and deals 7 damage. Draco Malfoy now has 0 health.
Draco Malfoy falls to the ground. Harry Potter wins the duel!
```

Rubric

- [45] `Wizard.java`
    - [5] Correct constructors
        * [5] Constructor chaining and default values
    - [5] Correct method to cast a spell
    - [5] Correct output when printing a wizard
    - [5] Correct method to interact with wizards
    - [5] Correctly checks if two wizards are the same
    - [20] Working duel method
        * [5] Duel continues until health falls to 0 at any point
        * [5] The wizard with less health attacks first. The other wizard attacks next. Turn order may change after a round
        * [5] Wizard with less health has 20% chance to gain 3 health
        * [2] Damage calculation is correct and includes the attack power of spells
        * [3] Wizards restore their health to their original max health after dueling

- [20] `Gryffindor.java`
  - [10] Correct constructor
    - [5] Correct random range for health and attack power
    - [5] Constructor chaining
  - [10] Correct method for interaction
    - [5] Duels
    - [5] Correct output
- [20] `Slytherin.java`
  - [10] Correct constructor
    - [5] Correct random range for health and attack power
    - [5] Constructor chaining
  - [10] Correct method for interaction
    - [5] Duels
    - [5] Correct output
- [15] `Spell.java`
  - [5] Correct constructor
  - [5] Correctly checks if two spells are the same
  - [5] Correct output when printing a spell

Allowed Imports

- You may import java.util.Random.

Javadocs

For this assignment, you will be commenting your code with Javadocs. Javadocs are a clean and useful way to document your code's functionality. For more information on what Javadocs are and why they are awesome, the online overview for them is extremely detailed and helpful.

You can generate the javadocs for your code using the command below, which will put all the files into a folder called javadoc:

```
$ javadoc *.java -d javadoc
```

The relevant tags that you need to include are `@author`, `@version`, `@param`, and `@return`. Here is an example of a properly Javadoc'd class:

```java
import java.util.Scanner;

/**
 * This class represents a Dog object.
* @author George P. Burdell
* @version 1.0
*/

public class Dog {

    /**
     * Creates an awesome dog (NOT a dawg!)
     */

    public Dog() {
    ...
    }

    /**
     * This method takes in two ints and returns their sum
     * @param a first number
     * @param b second number
     * @return sum of a and b
     */

    public int add(int a, int b) {
    ...
    }
}
```

A more thorough tutorial for Javadocs can be found here. Take note of a few things:

1. Javadocs are begun with `/**` and ended with `*/`.
2. Every class you write must be Javadoc'd and the `@author` and `@verion` tag included. The comments for a class should start with a brief description of the role of the class in your program.
3. Every non-private method you write must be Javadoc'd and the `@param` tag included for every method parameter. The format for an `@param` tag is `@param <name of parameter as written in method header> <description of parameter>`. If the method has a non-void return type, include the `@return` tag which should have a simple description of what the method returns, semantically.

Checkstyle can check for Javadocs using the -a flag, as described in the next section.

Checkstyle

For this assignment, we will be enforcing style guidelines with Checkstyle, a program we use to check Java style. Checkstyle can be downloaded here. To run Checkstyle, put the `jar` file in the same folder as your homework files and run

```
java -jar checkstyle-6.2.2.jar -a *.java
```

The Checkstyle cap for this assignment is **75 points**. This means that up to 75 points can be lost from Checkstyle errors.

Collaboration Statement

To ensure that you acknowledge collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment AT THE TOP of at least one java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Recall that comments are special lines in Java that begin with `//`.

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Wizard.java`
- `Gryffindor.java`
- `Slytherin.java`
- `Spell.java`

Make sure you see the message stating "HW09 submitted successfully". From this point, Gradescope will run a basic autograder on your submission as discussed in the next section.

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission: be sure to **submit every file each time you resubmit**.

Gradescope Autograder

For each submission, you will be able to see the results of a few basic test cases on your code. Each test typically corresponds to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.

The Gradescope tests serve two main purposes:

1. Prevent upload mistakes (e.g. forgetting checkstyle, non-compiling code)
2. Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or checkstyle your code; you can do that locally on your machine.

Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.