CS 1331 Homework 3 - Heroes

Introduction

This homework will cover basic class creation and instantiation.

Problem Description

The hero society needs your help! In order to better plan and prepare heroes for combat, they need a way to simulate battles. You will be responsible for implementing multiple classes in a coherent manner and will need to familiarize yourself with constructor chaining.

You will need to create `Hero.java`, `Sidekick.java`, `Villain.java`, and `Battle.java`.

Solution Description

The Hero Class

This class has the following instance variables:

- `private String name`: name of hero
- `private String power`: name of superpower
- `private Sidekick sidekick`: the hero's trusty sidekick
- `private int strength`: number representing their strength rating
- `private int defense`: number representing their defense rating
- `private int health`: number representing their remaining health

This class has the following methods:

- `public void attack(Villain v)`: prints "Take that, villain!" to the console, making sure that it will be printed on its own line then decreases the villain's health by the hero's strength minus the villain's defense. If the villain's defense is higher than the hero's strength, the villain's health should not change.
- `public void train()`: prints "I am ready for battle!" to the console, making sure that it will be printed on its own line and increases strength by 5 and defense by 1
- `public String toString()`: returns a string with following format:
  - A hero named `name` with `power` powers and `strength` strength, `defense` defense, and `health` health with `String representation of sidekick`
- Properly implemented getter and setter methods for each instance variable
  - Note for `setHealth()`: the health variable should never be less than zero. If their health hits zero, print "The hero has been defeated!" to the system, making sure that it will be printed on its own line.

The constructors for this class should be able to accept none (no-args constructor), two, or all six arguments (instance variable values).

- Pass in no argument: defaults name to "Mighty Fist", power to "Super-strength", sidekick to a new Sidekick object, strength and defense to 10 and health to 50
- Pass in two arguments(`name` and `power`): defaults sidekick to a new Sidekick object, strength and defense to 10 and health to 50
- Pass in six arguments(`name`, `power`, `sidekick`, `strength`, `defense`, and `health`): no defaults needed for this one.

> **NOTE:** Use constructor chaining for this. Order matters!

The Sidekick Class

This class has the following instance variables:

- `private String name`: name of sidekick
- `private int support`: amount of health the sidekick can heal

This class should have the following methods:

- `public void heal(Hero h)`: prints "I'm here for you." to the console, making sure that it will be printed on its own line and increases the hero's health by the value of `support`
- `public void cheer(Hero h)`: increases support by 10 and prints "We believe in you, `h's name`!" to the console, making sure that it will be printed on its own line.
- `public void cheer(Villain v)`: decreases v's defense by 1 and prints "You don't stand a chance, `v's name`!" to the console, making sure that it will be printed on its own line.
- `public String toString()`: returns a string with following format:
  - A sidekick named `name` that can heal `support` health.

The constructors for this class should be able to accept no (no-args constructor) or both arguments (instance variable values).

- Pass in no argument: defaults name to "Underdog",and support to 20
- Pass in six arguments(`name`, `support`): no defaults needed for this one.

> **NOTE:** Use constructor chaining for this. Order matters!

The Villain Class

This class has the following instance variables:

- `private String name`: name of villain
- `private int strength`: number representing their strength rating
- `private int defense`: number representing their defense rating
- `private int health`: number representing their remaining health

This class has the following methods:

- `public void attack(Hero h)`: prints "Perish, hero!" to the console, making sure that it will be printed on its own line then decreases the hero's health by the villain's strength minus the hero's defense. If the hero's defense is higher than the villain's strength, the hero's health should not change.
- `public void evilLaugh()`: increases strength by 5 and prints "MWAHAHA!" to the system, making sure that it will be printed on its own line.
- `public String toString()`: returns a string with following format:
  - A villain named `name` with `strength` strength, `defense` defense, and `health` health.
- Properly implemented getter and setter methods for each instance variable
  - Note for `setHealth()`: the health variable should never be less than zero. If their health hits zero, print "The villain has been defeated!" to the system, making sure that it will be printed on its own line.

The constructors for this class should be able to accept no (no-args constructor) or all four arguments (instance variable values).

- Pass in no argument: defaults name to "Dr. Evil", strength to 15, and defense to 5 and health to 50
- Pass in four arguments(`name`, `strength`, `defense`, and `health`): no defaults needed for this one.

> **NOTE:** Use constructor chaining for this. Order matters!

The Battle Class

This class has no instance variables and no explicit constructors.

This class has a main method that should do the following:

- Instantiate a Sidekick whose name is "Bucky Barnes" and whose support is 25
- Instantiate a Hero whose name is "Captain America", power is "Shield", sidekick is the Sidekick instantiated previously, strength is 15, defense is 10, and health is 50.
- Instantiate a default Villain
- Print the Hero
- Print the Villain
- Call the villain's `evilLaugh` method
- Call the hero's `train` method
- Call the villain's `attack` method on the hero
- Call the sidekick's `cheer` method on the villain
- Call the hero's `attack` method on the villain
- Call the sidekick's `cheer` method on the hero
- Call the villain's `attack` method on the hero, then call the hero's `attack` method on the villain (repeat this step a total of 3 times)
- Call the sidekick's `heal` method on the hero

- Print the Hero
- Print the Villain

If implemented correctly, the output should match the following:

```
A hero named Captain America with Shield powers and 15 strength, 10 defense, and 50
health with A sidekick named Bucky Barnes that can heal 25 health.
A villain named Dr. Evil with 15 strength, 5 defense, and 50 health.
MWAHAHA!
I am ready for battle!
Perish, hero!
You don't stand a chance, Dr. Evil!
Take that, villain!
We believe in you, Captain America!
Perish, hero!
Take that, villain!
Perish, hero!
Take that, villain!
Perish, hero!
Take that, villain!
The villain has been defeated!
I'm here for you.
A hero named Captain America with Shield powers and 20 strength, 11 defense, and 49
health with A sidekick named Bucky Barnes that can heal 35 health.
A villain named Dr. Evil with 20 strength, 4 defense, and 0 health.
```

Clarifications and Tips

You should not import any libraries or packages that trivialize the assignment. If you are unsure of whether something is allowed, ask on Piazza. In general, if something does a large part of the assignment for you, it is probably not allowed.

- Order matters for constructor chaining.

We recommend you test your code by doing the following:

- Create your own version of the Battle class and use it to test edge cases that could break your code

Javadocs

You will need to write Javadoc comments and watch for checkstyle errors with your submission.

- Every class should have a class level Javadoc that includes `@author <GT Username>` and `@version <version number>`.
- Every public method should have a Javadoc explaining what the method does and includes any of the following tags if applicable:
  - `@param <parameter name> <brief description of parameter>`
  - `@return <brief description of what is returned>`

See the CS 1331 Style Guide on Canvas for details.

Checkstyle

For each of your homework assignments we will run checkstyle and deduct one point for every checkstyle error, with the points deducted being capped.

For this homework the **checkstyle cap is 30**, meaning you can lose up to 30 points on this assignment due to style errors. This limit will increase with each homework.

- If you encounter trouble running checkstyle, check Piazza for a solution and/or ask a TA as soon as you can!
- You can run checkstyle on your code by using the jar file found on Canvas that includes xml configuration file specifying our checks. To check the style of your code run `java -jar checkstyle-6.2.2.jar *.java`.
- To check your Javadocs run `java -jar checkstyle-6.2.2.jar -j *.java`.
- Note that the command for checking code and the command for checking Javadocs are different. You will have to run both commands to fully test for style errors.
- Javadoc errors are the same as checkstyle errors, as in each one is worth a single point and they are counted towards the checkstyle cap.
- **You will be responsible for running checkstyle on *ALL* of your code.**
- Depending on your editor, you might be able to change some settings to make it easier to write style-compliant code. See the Customization Tips on Canvas for more information.

Collaboration

When completing homeworks for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homework
- Parts of the homework specification you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may not discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

OKAY: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"

BY NO MEANS OKAY: "Hey... the homework is due in like 20 minutes... Can I see your code? I *promise* won't copy it directly!"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public

repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Submission

- Submit your `Hero.java`, `Sidekick.java`, `Villain.java`, and `Battle.java` files as attachments to the `Homework 3` assignment on Canvas. You can submit as many times as you want, so feel free to submit as you make substantial progress on the homework. We only grade your **last** submission, meaning we will ignore any previous submissions.
- If you submit multiple times Canvas will append a number to your Java file (`Hero.java` becomes `Hero-1.java`). **Do not worry about this, we will fix the file name before compiling and running your code.**
- Non-compiling code will be given a score of 0. For this reason, we recommend submitting early and then confirming that you submitted ALL of the necessary files by re-downloading your file(s) and compiling/running them.

Good Luck! ¯\(°□°)/¯