Assignment 1: Minion Tracker

- See website for due date
- Submit deliverables to CourSys: https://courses.cs.ca/
- Late penalty is 10% per calendar day (each 0 to 24 hour period past due).
  - Maximum 2 days late (20%)
- This assignment is to be done **individually**. Do not show another student your code, do not copy code from another person/online. Please post all questions on Piazza.
  You may use general ideas you find online and from others, but your solution must be your own.
- See the marking guide for details on how each part will be marked.
- Default development environment is IntelliJ; however your may use another tool.
  Your submissions must be either a IntelliJ or Maven project. We support only IntelliJ.

1. Minion Tracker

The course website has a capture of some sample output showing how the entire application operates.

1.1 Requirements

- You must have (at least) the following three classes:
  - A class for holding minion information: name (`String`), height (`double`), and number of evil deeds completed (`int`).
    - A name may be more than one word (like "Evie the Evil").
    - Minion class must correctly implement `toString()`, as discussed in lecture.
  - A class for a text menu.
    - Have a field to store the menu's title (`String`)
    - Have a field to store the menu's options (as an array of `Strings`).
      - The menu option strings should not include numbers.
        So instead of having it store `{"1. Do thing one", "2. Do thing two"};`
        have it store `{"Do thing one", "Do thing two"}` and your program generate the numbers automatically.
    - Have a method to display (print) the menu to the screen.
      - Your program must **automatically** place a rectangle of *'s around the menu's title, sizing the rectangle to the length of the title. This must be computed, not hard-coded!
      - Automatically number the options starting at 1.
  - A class for the main application.
    - Contains a `main()` method which uses the menu and minion classes to implement the application.
    - Create an `ArrayList` of minions to hold the set of minions the user enters.
    - Be careful not to have much duplicate code in your application! Use functions.
- For this assignment, it is fine if all your classes are in one package.

1.2 Text Interface Requirements

- When you prompt the user to choose a menu option, if the user enters an invalid number you must re-ask the user to enter a valid value.
  - You may assume user always enters correct *type* of data: when asked for an `int`, it is OK if the program crashes when the user enters a non-`int` such as 'A'.
  - *Hint (optional):*
    Have a method in your menu class which handles this. It will already know how many options there are in the menu!

- **Main Menu Option: List minions**
  - List the name, height, and number of evil deeds for each minion.
  - Number the minions from 1.
- **Main Menu Option: Add a new minion**
  - Ask user for name (may be multiple words), and height (a double) of the minion.
  - Create a new minion with 0 evil deeds done.
- **Main Menu Option: Remove minion**
  - List the minions currently in the system.
  - Allow user to select a minion (by number), or 0 to cancel.
  - Entering an invalid number (like -3) handled by application. Entering invalid data *type* ("hello") need *not* be handled.
- **Main Menu Option: Attribute an evil deed to a minion**
  - Similar to "remove minion", user selects a minion to work with (or cancel).
  - Increment the number of evil deeds by the selected minion.
- **Main Menu Option: Debug dump of minion details**
  - For each minion in the minion-list, call `toString()` on each minion and print the result to the screen.
- **Main Menu Option: Exit**
  - Exit the application.
- Your text UI need not match the sample exactly, but it should be of equal quality.

1.3 Coding Requirements

- Your code must conform to the programming style guide for this course; see course website.
- All classes must have a class-level JavaDoc comment describing the purpose of the class.

1.4 Suggestions

- Think about the design before you start coding.
  - List the classes you expect to create.
  - For each class, decide what its responsibilities will be.
  - Think through some of the required features. How will each of your classes work to implement this feature? Can you think of design alternatives?

2. Deliverables

Submit a ZIP file of your project to CourSys: https://courses.cs.ca/
See course website for directions on creating and testing your ZIP file for submission.

All submissions will automatically be compared for unexplainable similarities.