CS 349 Task 1: Design by Contract

Description

This task demonstrates a basic application of the philosophy of design by contract that we are covering in class. It is an individual task, which may be revisited in teams, depending on the results.

Requirements

Your objective is to write the following three Java classes such that the specifications and constraints are satisfied. Beyond the stated aspects, you are allowed to implement them any way you want, including adding additional files.

The three required classes are called `EntityDog`, `EntityGhost`, and `EntityTree`.

Each includes the following public contract as appropriate:

1. A constructor that takes a `String` as an arbitrary identifier; e.g., "Casper".
2. A method `getID()` that returns the identifier in (1).
3. A method `move()` that prints to standard output "*id* moves by *mode*", where *id* is the identifier from (1) and *mode* is "walking", "floating", and "swaying" for the three classes, respectively.
4. A method `initiateAttack()` that takes an entity to attack, prints to standard output "*id1* initiated *type* attack against *id2*", where *id1* is the identifier from (1), *id2* is the identifier of the object being attacked, and *type* is "biting" and "supernatural" for the first two classes, respectively, and calls with itself the `receiveAttack()` method of the object being attacked. Note that according to the constraints below, a tree cannot initiate an attack, so this behavior is undefined; use your best judgment according to the discussion of the constraints below.
5. A method `receiveAttack()` that takes an entity that is attacking it and prints to standard output "*id1* received attack by *id2*", where *id1* is the identifier from (1) and *id2* is the identifier of the object attacking. Again, use your best judgment.

Your design needs to reflect the following constraints, which correspond to our mantra that your solution must do what it is supposed to do and not do what it is not supposed to do:

The prescriptive constraints on each type are:

- An `EntityDog` can attack only an `EntityDog` or an `EntityTree`.
- An `EntityGhost` can attack only an `EntityTree` or an `EntityDog`.
- An `EntityTree` cannot attack anything.

Your solution needs to allow what is permitted and prevent what is not. This requirement is open-ended to give you the opportunity to solve it however you feel is appropriate. You will be graded on this behavior, not on your specific approach (unless it is absurd).

Deliverables

Submit your source files. In addition, include a text file called `experience.txt` which states (1) whether you have had CS 350, and if so, with which instructor, and (2) the same for CS 488/490.

It is not necessary to comment your code.