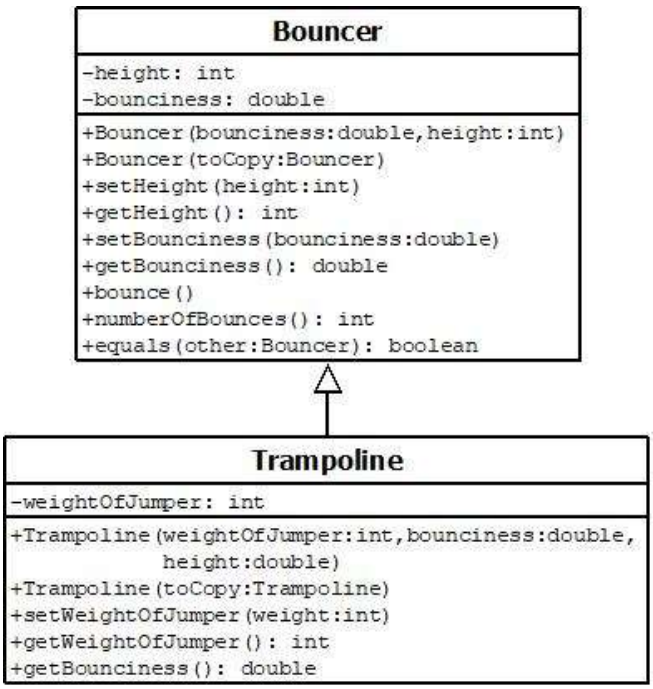
CPSC 219 – Coding Challenge 3 – Practice 2

This is a practice coding challenge. You may ask other students and your TA any questions you wish. Note that you are expected to complete the actual coding challenge independently. Having a solution for this coding challenge will not help you come up with your own solution for the actual coding challenge. Only if you can solve this practice independently can you be confident that you can complete the actual coding challenge successfully. See previous coding challenges for rules for coding challenges and best practices for success.

Remember to always create a 'skeleton' first and compile and test this using the provided JUnit test. For this coding challenge, you will be asked to create two or three classes. Make sure you create a skeleton of all classes first and place them in a zip file for submissions to WebCAT.

Skeleton of Bouncer and Trampoline



Additional Requirements for Bouncer

height should be greater than 0. If 0 or less is provided, set it to a default of 1.
bounciness is a percentage value (between 0 and 1) and should not be 0% (meaning no bounce) nor 100% meaning no loss of height on a bounce. If an invalid percentage is provided, set the bounciness to a default of 50%.
bounce() should update the height to reflect a single bounce. The object will be bounciness% of it's original height after a single bounce. (eg, if the height is 100 and bounciness is .9, then height is 90 after a single bounce.) The height should always be rounded down to the nearest integer.
numberOfBounces() returns the number of bounces before the object remains remains still on the ground (height is zero). The height should not change.
*equals()*returns true if the argument has the same height and bounciness. Note that the operator == will return true if two objects refer to the same object.

Additional Requirements for Trampoline

weightOfJumper is between 50 (inclusive) and 300 (inclusive) pounds. If weight is invalid, default to 140.

getBounciness is overridden in this child class to adjust the bounciness based on the jumpers weight as follows: If the jumper's weight is less than 100, set it to 75% of the original bounciness, if greater than 150, set it to 110% of the original bounciness, if greater than 200 set it to 120% of the original bounciness. Leave the bounciness unchanged otherwise. (Note: you will need to call getBounciness in the parent to do this computation.)

Notes: Do not duplicate instance variables from parent class in child class. Instead invoke appropriate super constructor and methods in parent class.