Decorator Pattern

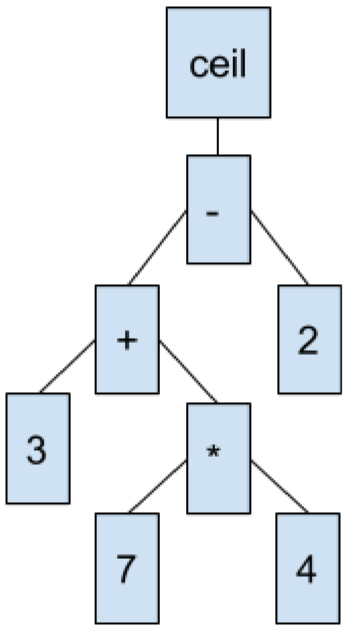
Author: Brian Crites

You <u>must</u> work in a group of two for this lab

In this lab you will be extending your Composite & Strategy Pattern lab with a decorator, which will be used to modify expressions. For example, we may want to create the following equation:

```
ceil(3 + 7 * 4 - 2)
```

would be represented with the following tree, with the top most node being a decorator



You can see how the ceiling function simply modifies the return of the statement, and that the return value at the top would be the ceiling of the equation below. You will be in charge of building three separate decorator classes, and testing each one at different levels of your system (including when decorators have children which are other decorators). Note that it is easiest to fully test a system when it holds double values rather than integer values. You must create the following classes as decorators:

- Ceiling (`Ceil`)
- Floor (`Floor`)
- Absolute Value (`Abs`)

To implement the functionality of each of these classes, you may employ the cmath library (`math.h`). This makes the implementation of the functionality trivial, so it is important that you are fully testing all variations of decorator composition (specifically, when an Absolute Value points to a Ceiling/Floor and vice-versa), since this is one of the more important concepts when discussing decorator patterns.

Submission

To receive credit for this lab you must show an example program to your TA that demonstrates the full functionality of this pattern along with the composite and strategy patterns, and must explain to your TA the structure of both your classes and how they interact.

Once you have demo'd to your TA, submit a tarball of your code to the lab submission on iLearn.