Part 2: Triggers (WordTriggers)

(20/20 points)

Given a set of news stories, your program will generate **alerts** for a subset of those stories. Stories with alerts will be displayed to the user, and the other stories will be silently discarded. We will represent alerting rules as **triggers**. A trigger is a rule that is evaluated over a single news story and may fire to generate an alert. For example, a simple trigger could fire for every news story whose title contained the word "Microsoft". Another trigger may be set up to fire for all news stories where the summary contained the word "Boston". Finally, a more specific trigger could be set up to fire only when a news story contained both the words "Microsoft" and "Boston" in the summary.

In order to simplify our code, we will use object polymorphism. We will define a trigger interface and then implement a number of different classes that implement that trigger interface in different ways.

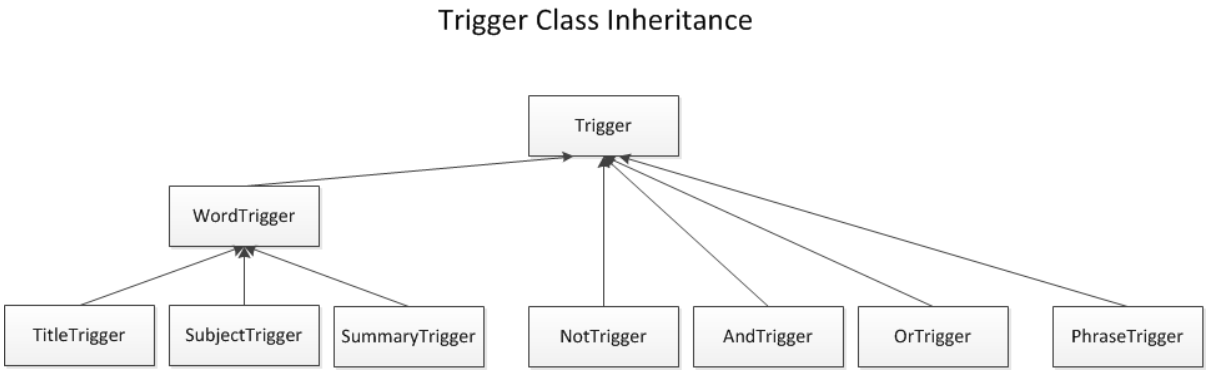
TRIGGER INTERFACE

Each trigger class you define should implement the following interface, either directly or transitively. It must implement the `evaluate` method that takes a news item (`NewsStory` object) as an input and returns `True` if an alert should be generated for that item. We will not directly use the implementation of the `Trigger` class, which is why it throws an exception should anyone attempt to use it

The class below implements the `Trigger` interface (you will not modify this). Any subclass that inherits from it will have an evaluate method. By default, they will use the evaluate method in `Trigger`, the superclass, unless they define their own `evaluate` function, which would then be used instead. If some subclass neglects to define its own `evaluate()` method, calls to it will go to `Trigger.evaluate()`, which fails (albeit cleanly) with the `NotImplementedError` exception:

```
class Trigger(object):
    def evaluate(self, story):
        """
        Returns True if an alert should be generated
        for the given news item, or False otherwise.
        """
        raise NotImplementedError
```

We will define a number of classes that inherit from `Trigger`. In the figure below, `Trigger` is a superclass, which all other classes inherit from. The arrow from `WordTrigger` to `Trigger` means that `WordTrigger` inherits from `Trigger` - a `WordTrigger` ***is a*** ***`Trigger`*** . Note that other classes inherit from `WordTrigger`.



[Click on the above image for a full-size view]

WHOLE WORD TRIGGERS

Having a trigger that always fires isn't interesting; let's write some that are. A user may want to be alerted about news items that contain specific words. For instance, a simple trigger could fire for every news item whose *title* contains the word "Microsoft". In the following problems, you will create a `WordTrigger` *abstract class*and implement three classes that inherit from this class.

The trigger should fire when the whole word is present. For example, a trigger for "soft" should fire on:

- Koala bears are soft and cuddly.
- I prefer pillows that are soft.
- Soft drinks are great.
- Soft's the new pink!
- "Soft!" he exclaimed as he threw the football.

But should **not** fire on:

- Microsoft recently released the Windows 8 Consumer Preview.
- Downey makes my clothes the softest they can be!

This is a little tricky, especially the case with the apostrophe. For the purpose of your parsing, pretend that a space or any character in `string.punctuation` is a word separator. If you've never seen `string.punctuation` before, go to your interpreter and type:

```
>>> import string
>>> print string.punctuation
```

Play around with this a bit to get comfortable with what it is.
The `split` and `replace` methods of strings will almost certainly be helpful as you tackle this part.

You may also find the string methods `lower` and/or `upper` useful for this problem.

PROBLEM 2

Implement a word trigger abstract class, `WordTrigger`. It should take in a string `word` as an argument to the class's constructor.

`WordTrigger` should be a subclass of `Trigger`. It has one new method, `isWordIn`, which takes in one string argument `text`. It returns `True` if the whole word `word` is present in text, `False` otherwise, as described in the above examples. This method should not be case-sensitive. Implement this method.

Hint

Because this is an abstract class, we will not be directly instantiating any `WordTrigger`s. `WordTrigger` should inherit its `evaluate` method from `Trigger`. We do this because now we can create subclasses of WordTrigger that use its `isWordIn` method. In this way, it is much like the Trigger interface, except now actual code from this `WordTrigger` class is used in its subclasses.

PROBLEM 3

You are now ready to implement `WordTrigger`'s three subclasses: `TitleTrigger`, `SubjectTrigger`, and `SummaryTrigger`.

Implement a word trigger class, `TitleTrigger`, that fires when a news item's **title** contains a given word. The word should be an argument to the class's constructor. This trigger should not be case-sensitive (it should treat "Intel" and "intel" as being equal).

For example, an instance of this type of trigger could be used to generate an alert whenever the word "Intel" occurred in the title of a news item. Another instance could generate an alert whenever the word "Microsoft" occurred in the title of an item.

Think carefully about what methods should be defined in `TitleTrigger` and what methods should be inherited from the superclass. This class can be implemented in as few as 3 lines code!

Hint

Once you've implemented `TitleTrigger`, the `TitleTrigger` unit tests in our test suite should pass. Run `ps7_test.py` to check.

Canopy specific instructions: Every time you modify code in `ps7.py` go to Run -> Restart Kernel (or hit the CTRL with the dot on your keyboard) before running `ps7_test.py`. **You have to do this every time you modify the file** `ps7.py` **and want to run the file** `ps7_test.py`, otherwise changes to the former will not be incorporated in the latter.

PROBLEM 4

Implement a word trigger class, `SubjectTrigger`, that fires when a news item's **subject** contains a given word. The word should be an argument to the class's constructor. This trigger should not be case-sensitive.

Once you've implemented `SubjectTrigger`, the `SubjectTrigger` unit tests in our test suite should pass.

PROBLEM 5

Implement a word trigger class, `SummaryTrigger`, that fires when a news item's **summary** contains a given word. The word should be an argument to the class's constructor. This trigger should not be case-sensitive.

Once you've implemented `SummaryTrigger`, the `SummaryTrigger` unit tests in our test suite should pass.