CIS 330: Project #3C
Assigned: May 9th, 2018
Due May 16th, 2018
(which means submitted by 6am on May 16th, 2018)
Worth 7% of your grade

Please read this entire prompt!

Assignment: Change your 3B project to be object-oriented.

== New code available on the website ==

=== main3C.C ===

Start with my main3C.C. It shows what the interfaces should be for the modules. Do not modify my main3C.C, aside from adding "#includes" and print statements (if you want).

=== Makefile ===

I added another Makefile. Note that I put my Filter class and all of its derived types in a file called "filter.C" / "filter.h". I didn't think it was worth splitting them into separate files.

== Code you should be re-using ==

=== Image ===

Re-use your Image class from 3B. Note that I found it necessary to add a "ResetSize" method that changes the size of the Image after it is already constructed. (The filters construct the Image output before they know the size it should be ... so I couldn't use the parameterized constructor variant to set the Image size).

Changes to Image:

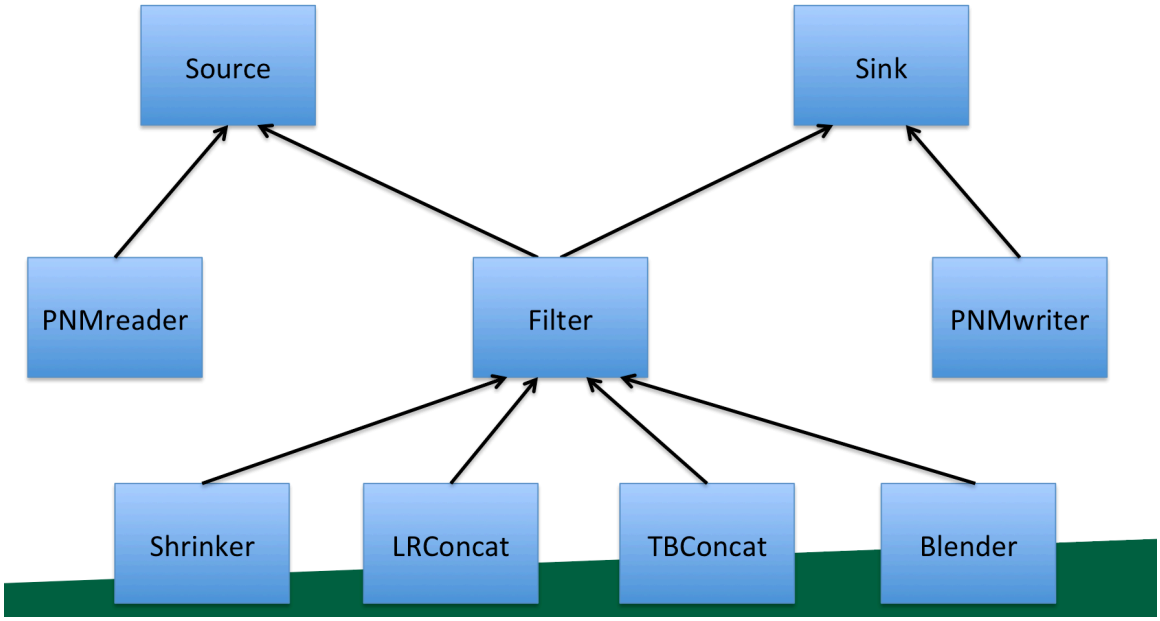
- Add public/private
- Data members must be private
- You will need getters and setters to modify the data members.

=== Functions, Reader, Writer ==

All of this code will be re-purposed into the objects you create

== New objects ==

You should create the following inheritance hierarchy:



Source:
This object should have a data member that is an Image.
It should have a pure virtual function called Execute.
It should have a method called GetOutput that returns a pointer to its Image data member.

Sink:
This object should contain two Image pointers. (It needs two, since some filters take two inputs.) The Image pointers can be set with SetInput and SetInput2. This object should not make copies of its Image inputs ... it should simply point at Images that come from Sources.

PNMreader:
This object needs to read the file with its Execute. Since Execute takes no arguments, the object needs to know about the filename ahead of time. So it is passed in via the constructor. The constructor will need to allocate memory for a string and copy over the string. This means it should also have a destructor, which frees the memory. Although this seems like an awkward way to do it, it will make our lives easier as we continue this project.

Filter: you might be surprised how little it takes to implement a filter.

Shrinker, LRCombine, TBCombine, Blender: these will each apply their operation in the Execute method. Blender has the same issue as PNMReader where it needs to take an argument in its constructor.

My tips:

- consult main3C.C for further clarification on the required methods. (You need to conform to my main3C.C ... don't change main3C.C)
- test often! Whenever you have something that you can compile and run, then do it. Writing code without the feedback of compiling and running it is a bad strategy. This is not just true for this project ... it is true for every project you will ever work on. This means you **should** modify main3C.C … you should be removing filters so that you can test pieces as you go. Of course, change it back to the original form for submission.
- Print statements are hugely useful for debugging.

== What to turn in ==

tar cvf proj3C.tar image.C sink.C source.C PNMreader.C PNMwriter.C filter.C image.h sink.h source.h PNMreader.h PNMwriter.h filter.h

Finally: I want use you to use a traditional file structure.
If a method has a single line of code, then you may inline it in the header:

```
class Image
{
    int GetWidth() { return width; };  // this is OK … one line of code
};
```

If there is more than one line of code, then the method should be defined in the corresponding .C file. You will lose points if you do not follow the traditional file structure