CISC 2000—Computer Science II
Fall, 2013

Programming Project #2: Extending the GUI
Date Due: Wednesday 9 October 2013

Define a new class `Square`, as an extension to the `Shape` class.
Here's what the class definition should contain:

- It should have two constructors.
  - The parameter list of the first constructor consists of a `Point` (the upper left-hand corner) and an `int` (the length of the `Square`'s side).
  - The parameter list of the second constructor consists of two `Point`s, which are the upper left-hand and lower right-hand corners of the `Square`. This constructor will need to do some error handling (see below).
- It should have two member functions:
  - `draw_lines()`, which should draw the lines of the `Square`.
  - `side()`, which should return the side-length of the `Square`.
- It only needs to have one data member, namely, an `int` that holds the side-length of the `Square`.

I'm not telling you what access specifiers to use for the member functions and data members; part of this assignment is to figure this out for yourself, using the criteria we've discussed in class.

Your solution will consist of three files:

- A header file `Square.h`.
- An implementation file `Square.cc`.
- A file `proj2.cc`, which will test out your implementation of `Square`. Your code should test at least as much functionality as my sample version does (see below).

A few considerations:

1. You might find it useful to model your implementation of `Square` on the author's implementation of `Rectangle`. Look at the relevant parts of `Graph.h` and `Graph.cc`, which may be found in the directory

   `~agw/class/cs2/share/Graphics`

   on the Departmental Linux machines. You might find it handy to do a cut-and-paste from these files into the files `Square.h` and `Square.cc`.

2. You should do this project in a directory named "`~/private/cs2/proj2`". As with the previous project, you'll need to create this directory before using it. You should always place yourself within said directory before doing any work on the project.

3. I have made an executable version of this program , called `proj2-agw` for you to try out. This is available in the directory

   ```
   ~agw/class/cs2/share/proj2
   ```

   on the Departmental Linux machines. This means that once you log in and `cd` into said directory, you execute them by simply typing in its name (`proj2`). As always, you should try out my sample version before you start working on the assignment. This is *especially* important for Project 2, because I expect your version of `proj2.cc` to do *at least* as much as mine does.

   You might find it useful to copy same into your working directory.

4. The directory "`~agw/class/cs2/share/proj2`" also contains a `Makefile`, which you should copy to your working directory (`~/private/cs2/proj2`). Once you've done this, you can use the `make` command (from within this working directory) as follows:

   (a) The command `make` (by itself) will compile and link the source file `proj2.cc`, producing and executable program named `proj2`. (If you really feel like typing more, the command `make proj2` will also do this. ☺) Note that you must have files named `proj2.cc`, `Square.h`, and `Square.cc` in the working directory for this to work.

   (b) If you want to precede more incrementally, you can do the following:

   - The command `make Square.o` will compile `Square.cc` into non-executable object code. You must have files named `Square.cc` and `Square.h` for this to work.
   - The command `make proj2.o` will compile `proj2.cc` into non-executable object code. You must have files named `proj2.cc`, `Square.cc` and `Square.h` for this to work.

   (c) `make clean` will clean out the directory.

5. Make sure that you adhere to our documentation standards, i.e., block comments at the beginning of each file, documentation of member functions and data members. Dr. Stroustrup doesn't adhere to these standards, but (then again) he's not enrolled in this class ☺.

6. Note that some pairs of `Point`s do not yield a square, but a rectangle whose width and height are not the same. For example, this would happen if you were to do

   ```
   Point p1(100, 200);
   Point p2(300, 800);
   Square s(p1, p2);
   ```

   Your two-point constructor should call `error()` should such a situation arise.

7. My sample solution mentions that the sidelength of the square is 200 pixels. This output is the test of whether the `side()` member function works properly. Creating a `Text` object via

   ```
   Text t1(p, "The side-length of the blue-outlined square is 200 pixels");
   ```

   does not test this capability. To do this properly, you'll need to call the apropriate member function within an output operation that involves an `ostringstream`.

8. You might be tempted to design `Rectangle` as a derived class of `Square`. This is a bad idea. You should be able to explain why this is a bad idea. (Such a question, or something similar to same, might well appear on an exam.)

9. When you're ready to turn in your listing for this project, issue the shell commands

   ```
   a2ps -o - proj2.cc Square.h Square.cc | ps2pdf - proj2.pdf
   ```

   Now take a moment to see what the listing looks like, by issuing the shell command

   ```
   xpdf proj2.pdf &
   ```

   If you're happy with what you see, mail me `proj2.pdf` by issuing the shell command

   ```
   mail -s "Project 2" -r  bovik@cs.cmu.edu agw < proj2.pdf
   ```

   Note that unless you happen to be Dr. Harry Q. Bovik, the address `bovik@cs.cmu.edu`[1] is simply a placeholder. You should replace it by the email address to which my confirmation message should be sent, i.e., the email address you most commonly use.

Have fun!

[1] Aren't you even *slightly* interested in the story of Harry Q. Bovik by now?