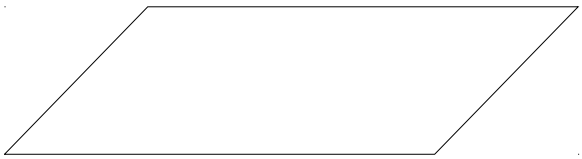
Program: Shapes

Using the template provided on the class web site, your task in this programming assignment is to extend the shapes example in a recent lesson (i.e., the example with Shape, Rectangle, Square, and Triangle classes) as follows:

- Add a new **Parallelogram** class that inherits from the Shape class;
- Redefine/rename the existing instance variables in the Shape class, **length** and **width**, to **width** and **height** in order to increase clarity with respect to displaying the shapes with asterisks;
- Set the width and height of a shape to 1 by default;
- Add accessors and mutators for all instance variables;
- Add range checking to ensure that the values of width and height are always greater than 0; and
- Replace the existing **draw** function with the magic **__str__** function so that shapes can be directly **print**ed.

Let's briefly review what a parallelogram is: a shape with two pairs of parallel sides. The pairs are of equal length; therefore, a parallelogram is basically a skewed rectangle/square. Here's an example of one:



Since the various existing shape classes display the shapes using asterisks, parallelograms will have to be similarly displayed. Here's how a parallelogram with a width of 6 and a height of 3 should be displayed in your program (note that the parallelogram below is centered for clarity):

```
      * * * * * *
    * * * * * *
  * * * * * *
```

Notice how the left and right sides are always displayed at an angle similar to a forward slash (/). This is a constraint in this assignment. Also note that you must **not** modify the main part of the program:

```
r1 = Rectangle(12, 4)
print r1
s1 = Square(6)
print s1
t1 = Triangle(7)
print t1
p1 = Parallelogram(10, 3)
print p1
r2 = Rectangle(0, 0)
print r2
p1.width = 2
p1.width = -1
```

```
p1.height = 2
print p1
```

Therefore, your output should look **exactly** like the following (you should confirm this by stepping through the main part of the program listed above):

```
jgourd@pi:~$ python Shapes.py
* * * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * *

* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *

* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*

     * * * * * * * * * * *
   * * * * * * * * * * *
* * * * * * * * * * *

*

   * *
* *
```

Note how the rectangle, *r2*, is instantiated with a width and height of 0 (which is invalid). As a result, the rectangle is instead instantiated with a width and height of 1 (the defaults). Also note how the parallelogram, *p1*, has its width changed to -1 (which is invalid). Therefore, this change is ignored. Ultimately, you should use the main part of the program and the output above to help you update/design the classes properly.

To help clarify, here are some specifics and/or constraints:

(1) Make sure to address all of the specifics outlined at the beginning of this document;
(2) When instantiating a new shape, the default values of the instance variables width and height should be 1;
(3) Range checking for the width and height should ensure that any invalid value specified (i.e., 0 or less) is ignored;

(4) Use the decorator method discussed in class for accessors and mutators to properly *wrap* the instance variables;
(5) Properly (and efficiently) implement inheritance (i.e., place shared traits in superclasses);
(6) Note that **there is no need** to make the Shape class abstract;
(7) You must include a meaningful header, use good coding style, use meaningful variable names, and comment your source code where appropriate;
(8) Your output should be **exactly** like the sample run shown above; and
(9) You must submit your source code as a single .py file.

To assist you in this assignment, here's an updated class diagram: