18. Inheritance :: *Shape* Class Design

We are going to create a *Shape* class which is the superclass of *Oval*, *Rectangle*, *Square*, etc. The next question is: what should be the data members of a *Shape*? And once we answer that: what should be the method members of *Shape*?

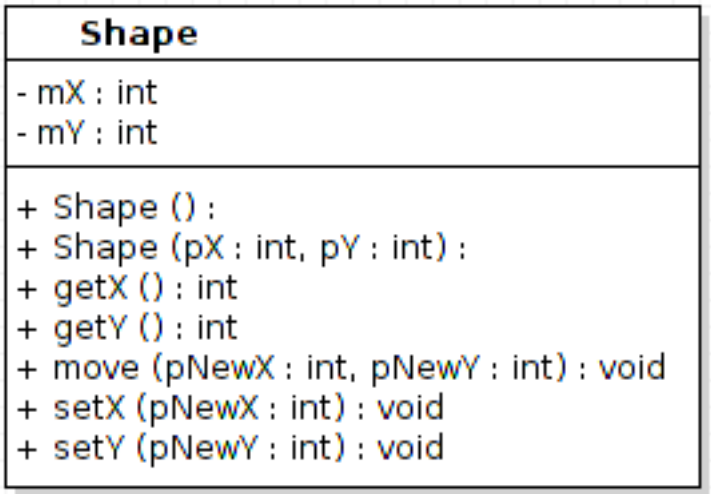
A *Shape* is a **generalization** of *Oval*, *Rectangle*, and *Square*. All *Oval*s are *Shape*s. All *Rectangle*s are *Shape*s. And all *Square*s are *Shape*s. An *Oval* is a **specific type** of *Shape*. A *Rectangle* is a specific type of *Shape* (different than *Oval*s, but similar in some respects).

The data members and method members of the *Shape* class should be things that all *Shape*s have in common. Why? Because subclasses of *Shape* will inherit those data members. It would not make sense to declare an *mWidth* data member in *Shape* because a *Point* (subclass of *Shape*) is technically dimensionless: it has no width, no height, and no area.

However, all *Shape*s are located on the graphical window at an (x, y) coordinate, so it **would make a lot of sense** to put the declarations for *mX*, *mY*, and the accessor/mutator methods in *Shape*. With inheritance, all subclasses of *Shape* (*Oval*, *Rectangle*, etc) will inherit *mX* and *mY*, and since the accessor/mutator methods are **public** all subclass objects will be able to call those methods—which is exactly what we require.

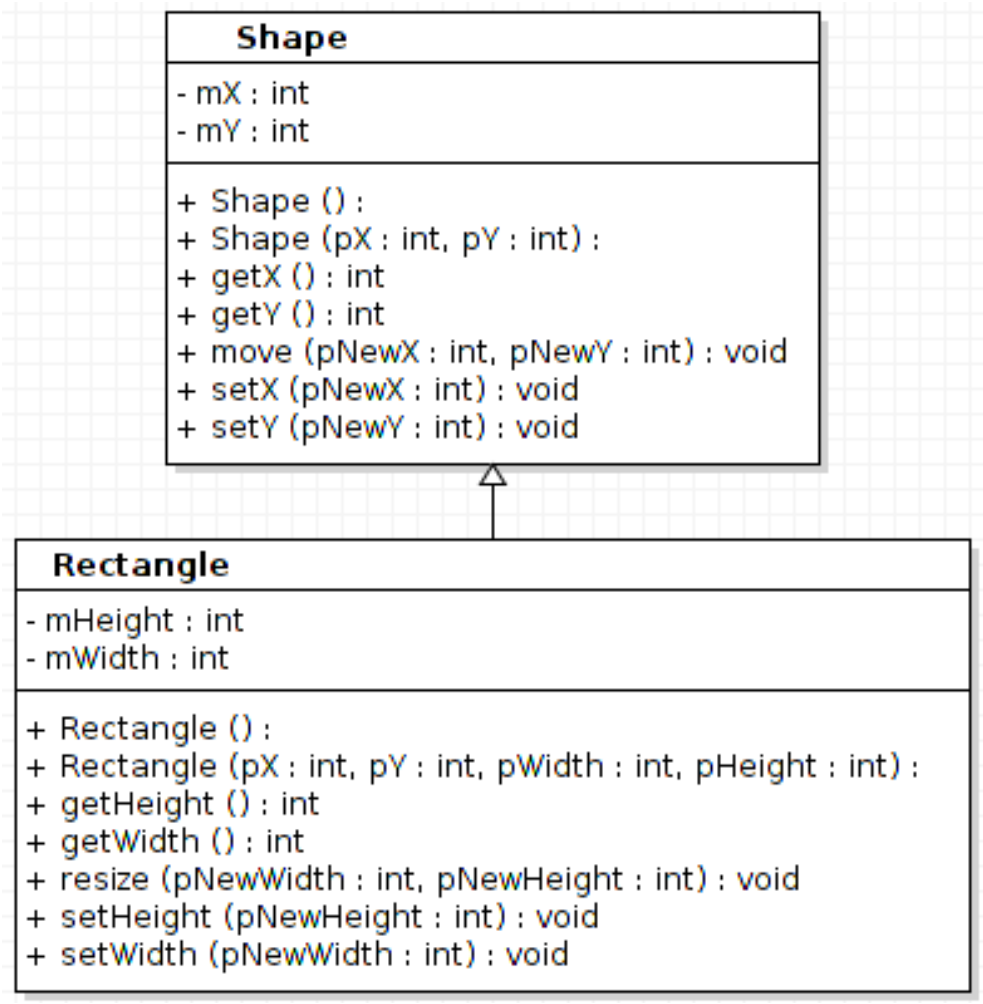
18. Inheritance :: *Shape* Class UML Diagram

Here is the *Shape* class diagram which encapsulates the x and y coordinates of *all* *Shape*s (including subclasses of *Shape*) and provides accessor/mutator methods for reading and writing the mX and mY instance variables.



18. Inheritance :: Revised *Rectangle* Class UML Diagram

Here is the revised *Rectangle* class diagram which becomes a subclass of *Shape*:



Note that the *Square* class diagram does not change because we had already eliminated the mX and mY instance variables from *Square*.