3. Inheritance :: Motivation

We have highlighted in **bold red** text the similarities between the *Rectangle* and *Square* classes. You should note that there are **a lot** of similarities. In fact, there are so many that to write the code for the *Square* class we: (1) copied the file *Rectangle.java* to a file *Square.java*; (2) edited *Square.java* and performed a global-search-and-replace for *Rectangle* changing all occurrences to *Square*; (3) performed a global-search-and-replace for *mWidth* changing all occurrences to *mSideLen*; (4) performed a global-search-and-replace for *getWidth*() and *setWidth*() replacing all occurrences to *getSideLen*() and *setSideLen*(); and (5) deleted the remainder of the code which dealt with the *mHeight* instance variable.

In programming, *anytime* you find yourself copying-and-pasting code from one place to another and making a few changes to it so it has essentially the same behavior as the code you copied, there is generally **a better way** to do what you are doing.

In this situation, the better way becomes apparent when we first realize that a *Square* **is a** *Rectangle* and specifically, a *Square* is simply a *Rectangle* where *mWidth* and *mHeight* are always equal. So rather than creating an entirely new *Square* class, why don't we take advantage of the *Rectangle* class somehow—it has already been written, tested, and debugged after all—and reuse the *Rectangle* class to create the *Square* class.

As we saw in the object oriented design section, a **generalization** or **inheritance** relationship exists between two classes when one class (the **subclass**) is based on another class (the **superclass**). A *Rectangle* is a generalization of a *Square*: all *Square*s are *Rectangle*s.

3. Inheritance :: Motivation (continued)

Conversely, we can say that a *Square* is a specialization of a *Rectangle*: a *Square* is a specific type of *Rectangle* (one in which the width and height are the same). Whenever a generalization/specialization relationship exists between two classes in an OO design, that relationship can be modeled and implemented using **inheritance**. Another way to say this, is that whenever we have one class (the **subclass**) which **"is a"** another class (the **superclass**), then we have a a generalization/specialization relationship.