PRAXIS 3: Understand that all non-`static` methods can be overridden by default

By default, all non-`private`, non-`static` methods of classes can be overridden by subclasses. The programmer of the class must take specific action in order to prevent subclasses from overriding, and thus changing, the behavior of a method. Subclasses can override any non-`static` methods unless the methods are declared `final`.

The `final` keyword is overloaded in Java. It can be used on instance and static class variables (see PRAXIS 2), on classes, or on methods to indicate that they cannot be overridden. For example:

```
class Base
{
  public void foo()
  {}
  public final void bar()
  {}
}

class Derived extends Base
{
  public void foo()
  {
    //Overriding Base.foo()
  }
  public void bar()
  {
    //Attempting to override Base.bar()
  }
}
```

Compiling this code results in the following error message:

```
Derived.java:15: The method void bar() declared in class Derived
cannot override the final method of the same signature declared in
class Base.  Final methods cannot be overridden.
  public void bar()
              ^
1 error
```