Special Topic 10.2

Final Methods and Classes

In Special Topic 10.1 on page 437 you saw how you can force other programmers to create subclasses of abstract classes and override abstract methods. Occasionally, you may want to do the opposite and *prevent* other programmers from creating subclasses or from overriding certain methods. In these situations, you use the `final` reserved word. For example, the `String` class in the standard Java library has been declared as

```
public final class String { . . . }
```

That means that nobody can extend the `String` class.

The `String` class is meant to be *immutable*—string objects can't be modified by any of their methods. Since the Java language does not enforce this, the class designers did. Nobody can create subclasses of `String`; therefore, you know that all `String` references can be copied without the risk of mutation.

You can also declare individual methods as final:

```
public class SecureAccount extends BankAccount
{
   . . .
   public final boolean checkPassword(String password)
   {
      . . .
   }
}
```

This way, nobody can override the `checkPassword` method with another method that simply returns `true`.