10.8 A Deeper Explanation of Issues with Calling Methods from Constructors

We've stated that you should not call overridable methods from constructors. To understand why, recall that when you construct a subclass object, the *subclass* constructor first calls a constructor in its direct *superclass*. At this point, any subclass instance-variable initialization code in the subclass constructor's body has not yet executed. If the *superclass* constructor then calls a method that the subclass overrides, the *subclass's* version executes. This could lead to subtle, difficult-to-detect errors if the subclass method uses instance variables that have not yet been initialized properly, because the subclass constructor hasn't finished executing.

Let's assume that a constructor and a *set* method perform the same validation for a particular instance variable. How should you handle the common code?

- If the validation code is brief, you can duplicate it in the constructor and the *set* method. This is a simple way to eliminate the problem we're considering here.
- For lengthier validation, you can define a `static` validation method—typically a `private static` helper method—then call it from the constructor and from the *set* method. It's acceptable to call a `static` method from a constructor, because `static` methods are not overridable.

It’s also acceptable for a constructor to call a `final` instance method, provided that the method does not directly or indirectly call any overridable instance methods.