10.12 `private` Constructors

In Section 3.4, we mentioned that constructors are normally declared `public`. Sometimes it's useful to declare one or more of a class's constructors as `private`.

Preventing Object Instantiation

You can prevent client code from creating objects of a class by making the class's constructors `private`. For example, consider class `Math`, which contains only `public static` constants and `public static` methods. There's no need to create a `Math` object to use the class's constants and methods, so its constructor is `private`.

Sharing Initialization Code in Constructors

One common use of a `private` constructor is sharing initialization code among a class's other constructors. You can use delegating constructors (introduced in Fig. 8.5) to call the

`private` constructor that contains the shared initialization code.

Factory Methods

Another common use of `private` constructors is to force client code to use so-called “factory methods” to create objects. A **factory method** is a `public static` method that creates and initializes an object of a specified type (possibly of the same class), then returns a reference to it. A key benefit of this architecture is that the method’s return type can be an interface or a superclass (either `abstract` or concrete). We discuss factory methods in more detail in the online Design Patterns appendix.2

2. Gamma, Erich et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.