Item 2: Enforce the singleton property with a private constructor

A singleton is simply a class that is instantiated exactly once [Gamma95, p. 127]. Singletons typically represent some system component that is intrinsically unique, such as a video display or file system.

There are two approaches to implementing singletons. Both are based on keeping the constructor private and providing a public static member to allow clients access to the sole instance of the class. In one approach, the public static member is a final field:

```java
// Singleton with final field
public class Elvis {
    public static final Elvis INSTANCE = new Elvis();

    private Elvis() {
        ...
    }

    ... // Remainder omitted
}
```

The private constructor is called only once, to initialize the public static final field Elvis.INSTANCE. The lack of public or protected constructors guarantees a "monoelvistic" universe: Exactly one Elvis instance will exist once the Elvis class is initialized—no more, no less. Nothing that a client does can change this.

In a second approach, a public static factory method is provided instead of the public static final field:

```java
// Singleton with static factory
public class Elvis {
    private static final Elvis INSTANCE = new Elvis();

    private Elvis() {
        ...
    }

    public static Elvis getInstance() {
        return INSTANCE;
    }

    ... // Remainder omitted
}
```