8.10 Garbage Collection

Every object uses system resources, such as memory. We need a disciplined way to give resources back to the system when they're no longer needed; otherwise, "resource leaks" might occur that would prevent resources from being reused by your program or possibly by other programs. The JVM performs automatic **garbage collection** to reclaim the *memory* occupied by objects that are no longer used. When there are *no more references* to an object, the object is *eligible* to be collected. Collection typically occurs when the JVM executes its **garbage collector**, which may not happen for a while, or even at all before a program terminates. So, memory leaks that are common in other languages like C and C++ (because memory is *not* automatically reclaimed in those languages) are *less* likely in Java, but some can still happen in subtle ways. Resource leaks other than memory leaks can also occur. For example, an app may open a file on disk to modify its contents—if the app does not close the file, it must terminate before any other app can use the file.

A Note about Class `Object`'s `finalize` Method

Every class in Java has the methods of class `Object` (package `java.lang`), one of which is method `finalize`. (You'll learn more about class `Object` in Chapter 9.) You should *never* use method `finalize`, because it can cause many problems and there's uncertainty as to whether it will *ever* get called before a program terminates.

The original intent of `finalize` was to allow the garbage collector to perform **termination housekeeping** on an object just before reclaiming the object's memory. Now, it's considered better practice for any class that uses system resources—such as files on disk—to provide a method that programmers can call to release resources when they're no longer needed in a program. `AutoClosable` objects reduce the likelihood of resource leaks when you use them with the `try`-with-resources statement. As its name implies, an `AutoClosable` object is closed automatically, once a `try`-with-resources statement finishes using the object. We discuss this in more detail in Section 11.12.

Software Engineering Observation 8.8

Many Java API classes (e.g., class `Scanner` *and classes that read files from or write files to disk) provide* `close` *or* `dispose` *methods that programmers can call to release resources when they're no longer needed in a program.*