

8.14 Package Access

If no access modifier (`public`, `protected` or `private`—we discuss `protected` in [Chapter 9](#)) is specified for a method or variable when it's declared in a class, the method or variable is considered to have **package access**. In a program that consists of one class declaration, this has no specific effect. However, if a program uses *multiple* classes from the *same* package (i.e., a group of related classes), these classes can access each other's package-access members directly through references to objects of the appropriate classes, or in the case of `static` members through the class name.

Package access is rarely used.

[Figure 8.15](#) demonstrates package access. The app contains two classes in one source-code file—the `PackageDataTest` class (lines 5–19) containing `main` and the `PackageData` class (lines 22–30). Classes in the same source file are part of the same package. Consequently, class `PackageDataTest` is allowed to modify the package-access data of `Package-Data` objects. When you compile this program, the compiler produces two separate `.class` files—`PackageDataTest.class` and `PackageData.class`. The compiler places the two `.class` files in the same directory. You can also place class `PackageData` (lines 22–30) in a separate source-code file.

```
1 // Fig. 8.15: PackageDataTest.java
2 // Package-access members of a class are accessible
3 // in the same package.
4
5 public class PackageDataTest {
6     public static void main(String[] args) {
7         PackageData packageData = new PackageData(
8             8
9             // output String representation of package
10            System.out.printf("After instantiation:%n"
11
12            // change package access data in packageData
13            packageData.number = 77;
14            packageData.string = "Goodbye";
15
16            // output String representation of package
17            System.out.printf("%nAfter changing values"
18            }
19        }
20
21    // class with package access instance variables
22    class PackageData {
23        int number = 0; // package-access instance variable
24        String string = "Hello"; // package-access instance variable
25
26        // return PackageData object String representation
27        public String toString() {
28            return String.format("number: %d; string: "
29            }
30        }
```



```
After instantiation:
number: 0; string: Hello
After changing values:
number: 77; string: Goodbye
```

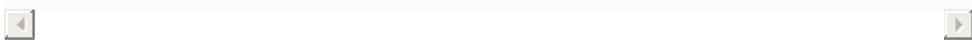


Fig. 8.15

Package-access members of a class are accessible by other classes in the same package.

In the `PackageData` class declaration, lines 23–24 declare the instance variables `number` and `string` with no access modifiers—therefore, these are package-access instance variables. Class `PackageDataTest`'s `main` method creates an instance of the `PackageData` class (line 7) to demonstrate the ability to modify the `PackageData` instance variables directly (as shown in lines 13–14). The results of the modification can be seen in the output window.