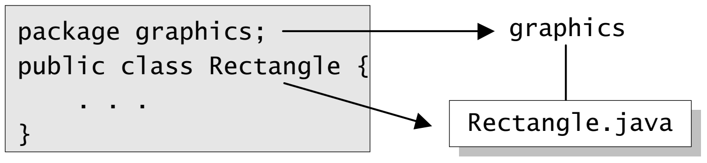


Figure 71 The source code for the `Rectangle` class is in the file `Rectangle.java`, which is located in a folder named `graphics`.

`Rectangle.java`, and the file would be in a directory named `graphics`. The `graphics` directory might be anywhere on the file system. Figure 71 shows how this works.

The qualified name of the package member and the path name to the file are parallel, assuming the UNIX file name separator slash (`/`):

class name	`graphics.Rectangle`
pathname to file	`graphics/Rectangle.java`

As you may recall, by convention a company uses its reversed Internet domain name in its package names. The fictional company whose Internet domain name is `taranis.com` would precede all its package names with `com.taranis`. Each component of the package name corresponds to a subdirectory. So if Taranis had a `graphics` package that contained a `Rectangle.java` source file, it would be contained in a series of subdirectories, as shown in Figure 72.

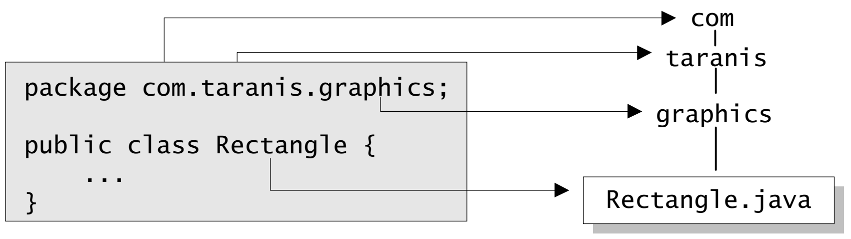
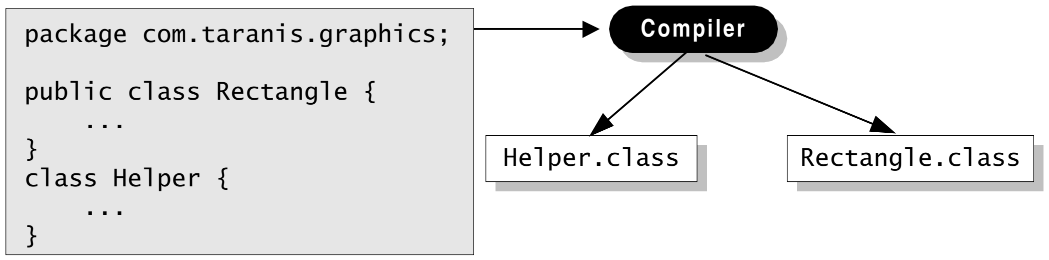


Figure 72 By convention, companies use their Internet domain names in reverse in their package names.

When you compile a source file, the compiler creates a different output file for each class and interface defined in it. The base name of the output file is the name of the class or the interface, and its extension is `.class`, as shown in Figur e73.



```
package com.taranis.graphics;

public class Rectangle {
    ...
}
class Helper {
    ...
}
```

Figure 73 The compiler creates a separate `.class` file for every class.

Like a `.java` file, a `.class` file should also be in a series of directories that reflect the package name. However, it does not have to be in the same directory as its source. You could arrange your source and class directories separately, as in Figure 74.

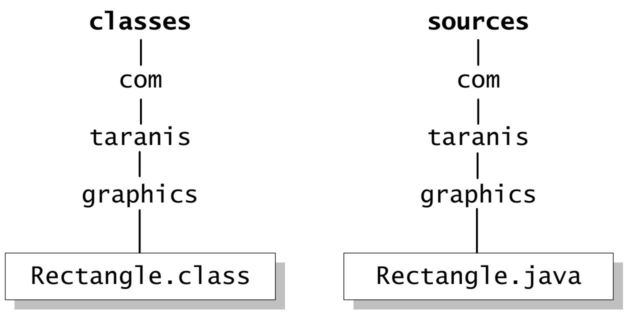


Figure 74 An example of how to arrange your source code and class files separately.

By doing this, you can give the classes directory to other programmers without revealing your sources.

Why all the bother about directories and file names? You need to manage your source and class files in this manner so that the compiler and the interpreter can find all the classes and interfaces your program uses. When the compiler encounters a new class as it's compiling your program, it must be able to find the class so as to resolve names, do type checking, and so on. Similarly, when the interpreter encounters a new class as it's running your program, it must be able to find the class to invoke its methods, and so on. Both the compiler and the interpreter search for classes in each directory or ZIP file listed in your class path.

Definition: A *class path* is an ordered list of directories or ZIP files in which to search for class files.