Southern New Hampshire University

CS 230 Java Packages

Overview

You probably have sub-folders under your Documents folder on your own computer for organizing your work. Packages are an organizational tool for developers. Their primary purpose is to organize large projects and simplify navigation within a project.

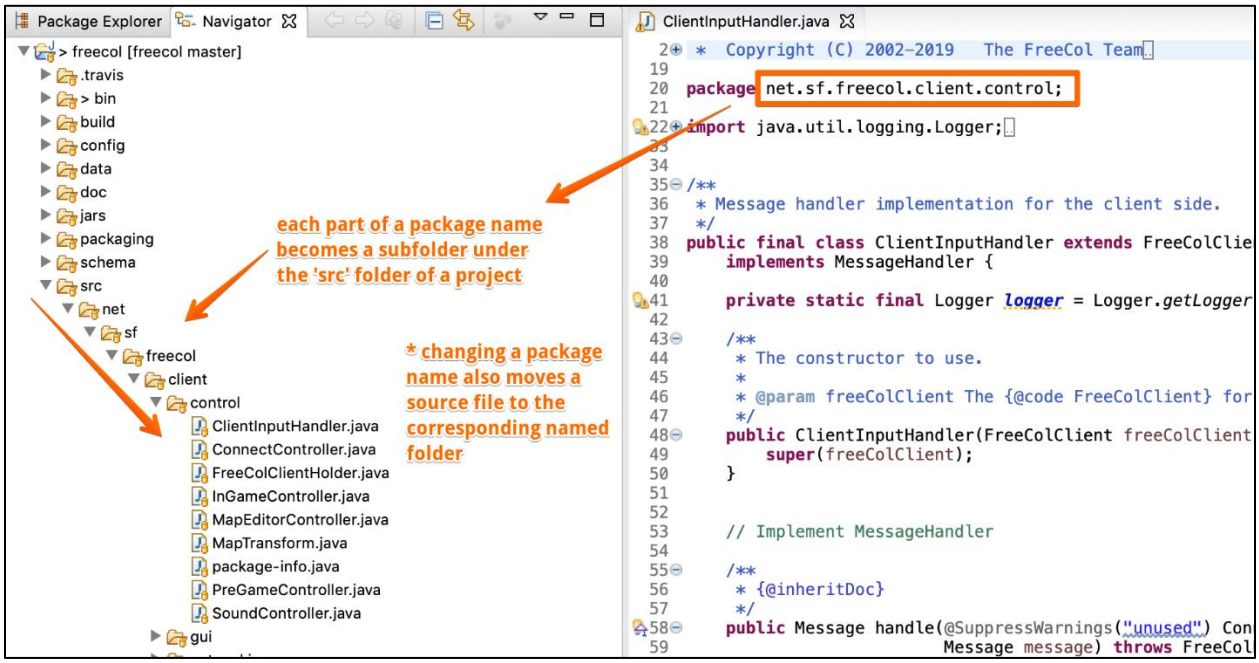
Java packages allow developers to group related class, interface, and other files together by giving them a name. This is similar to how a family surname is often shared by all the members of the family and identifies them as a **unit**.

Similar to the way a physical package sent to you from an online e-commerce vendor is a container for the items inside, a Java package can be thought of as a **container** for the Java classes within.

Naming Conventions

The long-established convention is to begin a package name with the Internet domain name in reverse order. Oracle would use **com.oracle** and NPR would begin with **org.npr**. One notable exception is the Java API framework that is part of the SDK; these packages begin with **java**.

The components of a package name correspond to the subfolders under the project's source directory. This screenshot shows the Navigator view within Eclipse.



The ClientInputHandler class has a package name of **net.sf.freecol.client.control**. As shown, it is located in the **src/net/sf/freecol/client/control** subfolder. Here is the breakdown of the package name:

`net.sf.freecol.client.control`

1. **net.sf** is the domain name in reverse order. Here's a case where the conventional rule is being bent a little. This project is part of SourceForge and the full domain name is "sourceforge.net," however the author chose to abbreviate it as **net.sf** instead of spelling it out as **net.sourceforge**. **Please note:** These types of shortcuts are NOT best practice and NOT recommended.
2. **freecol** is the project name; FreeCol is a free Java game based on the older game Colonization.
3. **client** is a sub-package of the game containing the client code (FreeCol uses a client/server architecture). There is another sub-package called "server" containing the server-related code.
4. **control** is a sub-package of the client code containing the controller classes.

In the case of FreeCol, there are 763 individual Java source files - a very large list to work with. Breaking the list down into smaller, more manageable groups of related files is what Java packages are designed for. Eclipse's Package Explorer further simplifies the navigation by showing the files under the full package name, making them easier to access without opening four or five subfolders to get to the source files.

Notice all the sub-packages under the **gui** package. It is clear that the **animation** package will contain animation-related code.

This is another important consideration – choose a meaningful name for your packages. A common technique is to organize your user-defined packages similar to the built-in packages found in the Java API documentation.

Namespace

Packages also serve to define a namespace for the classes, interfaces, and other types contained within them. For example, if you have created five classes, each in their own **JAVA** source file, and all of them have the same package name (package **com.example**), then these five classes are part of the **com.example** namespace.

A namespace is used to uniquely identify one or more names from other similar names of different objects, groups, or the namespace. A namespace makes it possible to distinguish objects with similar names but different origins.

Practically, namespaces provide a scope for the names contained within, similar to how variables defined at a class level are visible and known to the entire class. Meanwhile, variables defined within a method are visible and known to only that method, and classes and types within a package are known only to the classes and types within that package.

To be able to "see" and use the members of a package, you must use the **import** keyword to gain visibility to that package. You have likely done this before while programming in Java. You place the following statement at the top of the class you are developing:

```
import java.util.Scanner;
```

To reiterate, the best practice is to **explicitly list each class with a full package name on import statements** like the example above.

Reference

Techopedia. (n.d.) What is a namespace? - Definition from techopedia. Retrieved from

https://www.techopedia.com/definition/1341/namespace.