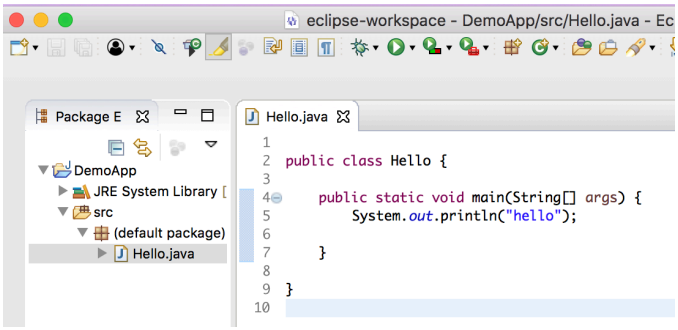
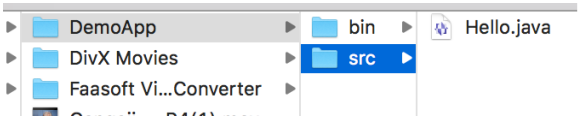
Packages

A package is a collection of related classes (and perhaps other sub-packages). There are three kinds of package in Java:

The default package

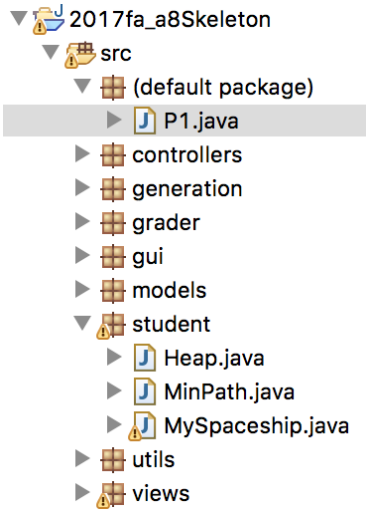
To the left below is an image of the Eclipse window. In the Package Explorer pane, project *DemoApp* contains directory *src* (for *source*), which contains the *default package*, which contains one class, *Hello*. To the right below is an image of the hard drive, showing that directory *DemoApp* contains directory *src*, and *src* contains file *Hello.java*. Thus, the files in the default directory appear directly in directory *src*; there is no directory for the default package.



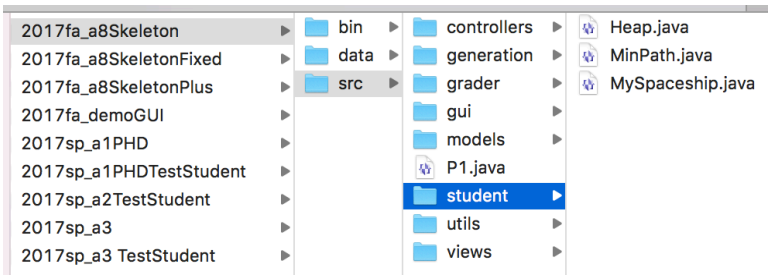


Local named packages

A local named package is stored on your hard drive in a directory with the same name as the package. It may contain subpackages. The image to the right below shows project *2017fa_a8Skeleton* in the Package Explorer pane. It has a default package, which contains one file, *P1.java*, along with 8 other packages. The first is named *controllers*; the last, *views*. Package *student* contains the files of three classes.



Below is the corresponding image of the hard drive. Note that *P1.java* is in directory *src*. You see also the files in package *student*.



Packages that come with Java

Java comes with well over 150 packages. Some contain classes for IO (input/output). Class *java.lang* contains classes for manipulating strings (sequences of characters); class *Math*, which contains many mathematical functions; and much more. Several packages in the building of GUIs (Graphical User Interfaces).

The first video on the tutorial "API & strings" shows you how to get to the webpages for the Java API documentation and read a package that is of interest to you. Please watch that video.

(continued on next page)

Packages

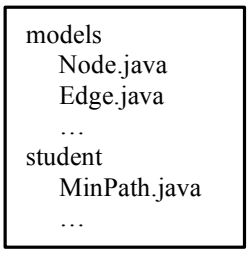
The package statement

A class that is in a package —but not the default package— must have a package statement at its beginning. For example, here is the beginning of file *MinPath.java* shown in package *student* near the bottom of the previous page:

```
package student;
/** This class contains the shortest-path algorithm and some other methods. */
public class MinPath { … }
```

The import statement

Suppose method *m* in class *MinPath* needs to reference class *Edge* in package *models* (see the diagram to the right), for example to define a variable *e* of class *Edge*. The outline of class *MinPath* below shows that it can be done using *models.Edge*. That is, use the package name, a period, and the class name:




```
package student;
/** This class contains the shortest-path algorithm and other methods. */
public class MinPath {
    public void m(…) {
        models.Edge e= …;          // Use class Edge in package models
        …
    }
}
```

If there are many references to class *Edge* in class *MinPath*, this can get rather cumbersome. *Importing* the class using an import statement after the package statement but before the class, as shown below, removes the need for "*models.*":

```
package student;
import models.Edge;
/** This class contains the shortest-path algorithm and other methods. */
public class MinPath {
    public void m(…) {
        Edge e= …;                 // Use class Edge in package models.
        ..
    }
}
```

Finally, if it is necessary to import several classes from a package, one can import *all* the classes using "*", as shown here:

```
import models.*;    // This imports all classes in package models
```

Classes in the Java API can also be imported using the import statement. For example, below, the import statement imports class *JFrame*, which is in package *swing*, which is in package *javax*:

```
import javax.swing.JFrame;
```