Java Packages

Suppose you have 4 Java source files Util.java, T1.java, T2.java and test.java. They are all sitting in the same directory and none of the source files have any "package" statements in them. So the classes belong to the *default package*.

Here is a directory listing of the current source and class files

```
$ ls -1
T1.class
T1.java
T2.class
T2.java
Util.class
Util.java
test.class
test.java
$ java test
Wed Sep 12 22:01:17 EDT 2012
T1
from T1
Wed Sep 12 22:01:17 EDT 2012
T2
$
```

Each source file is shown below:

```
public class Util {
   public static void printMessage(String s){
        System.out.println(s);
   }
   public static void printDate(){
        Date d= new Date();
        System.out.println(d);
   }
}
-----------------------------------------------------
public class T1 {
   public void f(){
        Util.printDate();
        Util.printMessage("T1");
        myApp2.shared.Util.printMessage("from T1");
   }
   public static void main(String[] args){
        T1 t1=new T1();
        t1.f();
   }
}
-----------------------------------------------------
public class T2 {
   public void f(){
        Util.printDate();
        Util.printMessage("T2");
    }
    public static void main(String[] args){
        T2 t2=new T2();
        t2.f();
    }
}
-----------------------------------------------------
public class test{
    public static void main(String[] args){
        T1 t1=new T1();
        t1.f();
        T2 t2=new T2();
        t2.f();
    }
}
```

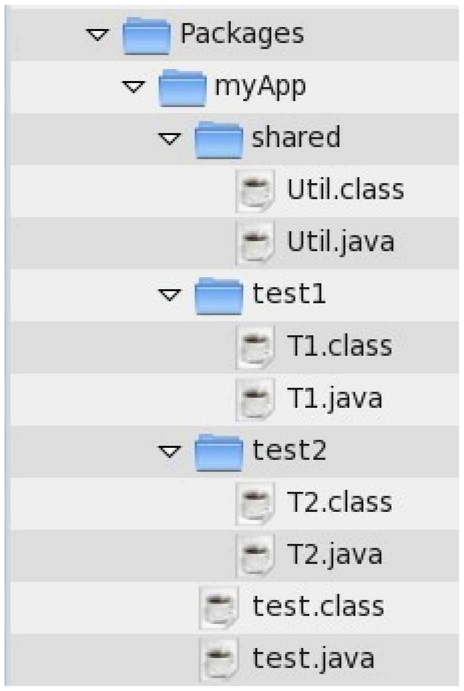
Suppose you decide to move the classes into separate **packages** as shown below. You are creating a package named myApp, and “subpackages” myApp.shared, myApp.test1 and myApp.test2 to place the different source files, as shown in the table below.

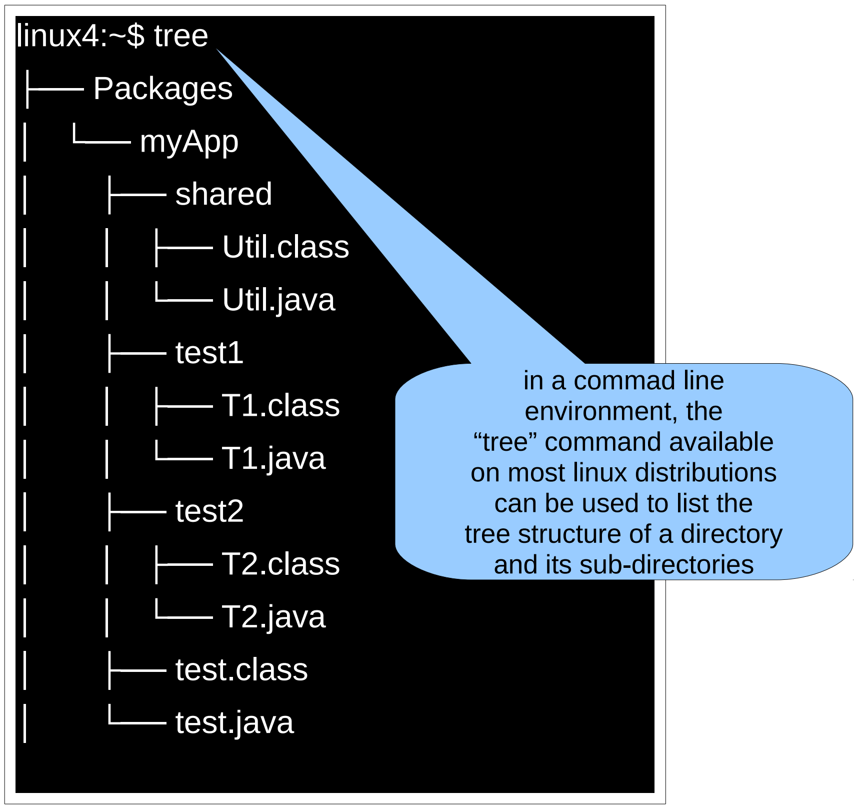
How will you modify the source files, and where will you place each of them? How will you compile and run the test class?

Destination Packages	
Package Name	**Class Name**
myApp.shared	Util
myApp.test1	T1
myApp.test2	T2
myApp	test

Util.java moved to myApp/shared/
T1.java moved to myApp/test1/
T2.java moved to myApp/test2/
test.java moved to myApp/

The new file locations are shown in the tree diagrams below. What we have done is to create a myApp package, along with myApp.shared, myApp.test1 and myApp.test2 packages.

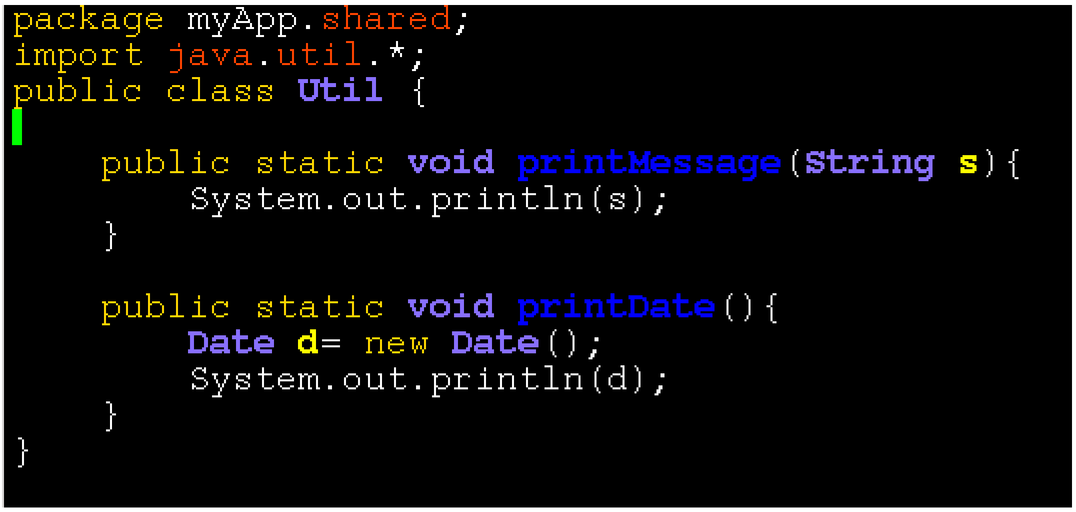




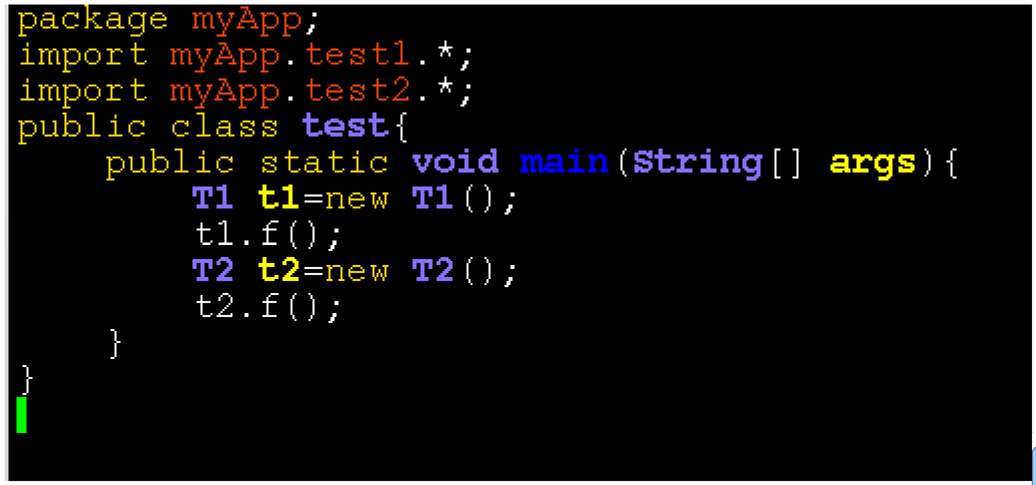
```
linux4:~$ tree
├── Packages
│   └── myApp
│       ├── shared
│       │   ├── Util.class
│       │   └── Util.java
│       ├── test1
│       │   ├── T1.class
│       │   └── T1.java
│       ├── test2
│       │   ├── T2.class
│       │   └── T2.java
│       ├── test.class
│       └── test.java
```

in a commad line environment, the "tree" command available on most linux distributions can be used to list the tree structure of a directory and its sub-directories

Each source file need to have, as their first non-comment line, a *package* statement saying which package that source file belongs to. The modifications for the source files are shown below. The root of the package is ~/Packages. *test.java* is in ~/Packages/myApp directory and has *package myApp;* as its first line. T1.java sits in ~/Packages/myApp/test1/ and has *package myApp/test1;* as its first line.
T2.java sits in ~/Packages/myApp/test2/ and has *package myApp/test2;* as its first line.
Util.java sits in ~/Packages/myApp/shared/ and has
package myApp/shared; as its first line.

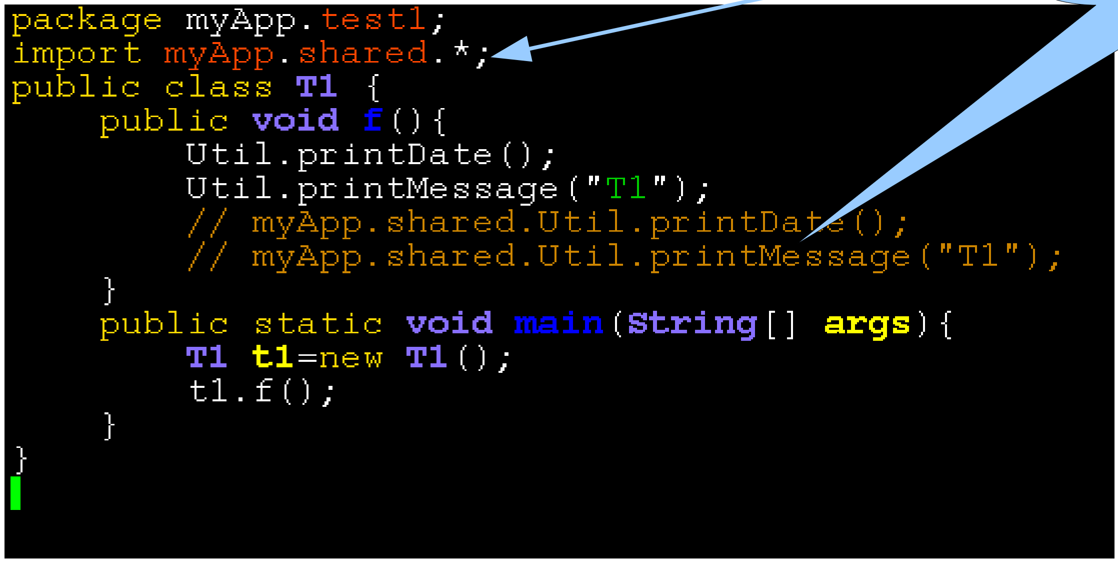


```java
package myApp.shared;
import java.util.*;
public class Util {

    public static void printMessage(String s){
        System.out.println(s);
    }

    public static void printDate(){
        Date d= new Date();
        System.out.println(d);
    }
}
```

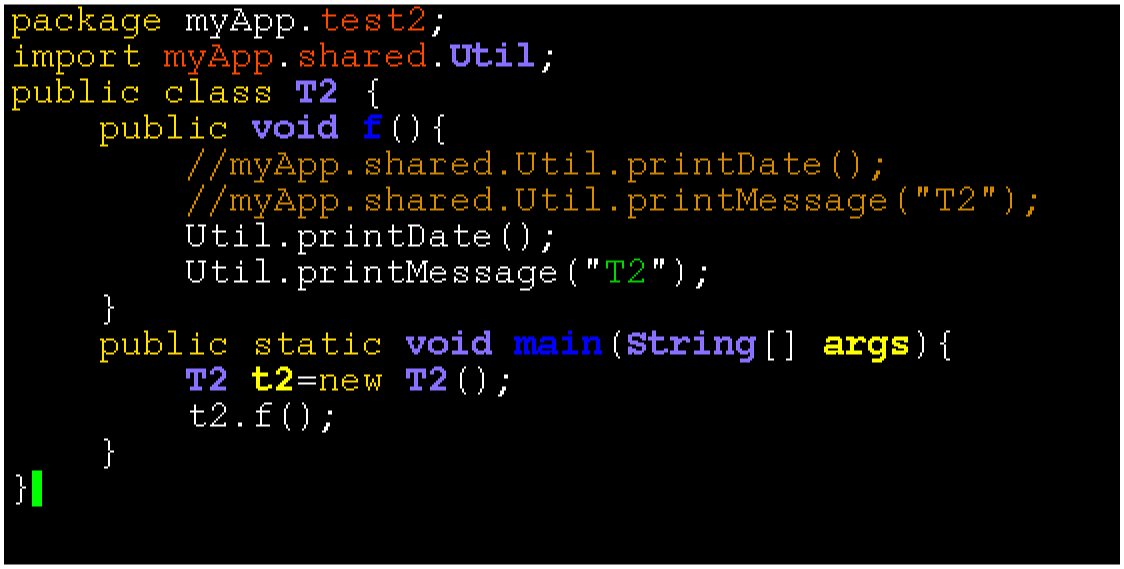


```java
package myApp;
import myApp.test1.*;
import myApp.test2.*;
public class test{
    public static void main(String[] args){
        T1 t1=new T1();
        t1.f();
        T2 t2=new T2();
        t2.f();
    }
}
```

If you don't have the import statement,
these will work



```java
package myApp.test1;
import myApp.shared.*;
public class T1 {
    public void f(){
        Util.printDate();
        Util.printMessage("T1");
        // myApp.shared.Util.printDate();
        // myApp.shared.Util.printMessage("T1");
    }
    public static void main(String[] args){
        T1 t1=new T1();
        t1.f();
    }
}
```



```
package myApp.test2;
import myApp.shared.Util;
public class T2 {
    public void f(){
        //myApp.shared.Util.printDate();
        //myApp.shared.Util.printMessage("T2");
        Util.printDate();
        Util.printMessage("T2");
    }
    public static void main(String[] args){
        T2 t2=new T2();
        t2.f();
    }
}
```

To compile the java source files in myApp package, you need to execute "javac" from myApp's parent directory:

```
linux4:~$ javac myApp/*.java
```

The shortcut '*.java' compiles all java source files in myApp/ along with any other source files necessary in other sub-directories. Instead of using this shortcut you could compile each source file separately. The following screenshot shows both processes. To run the test.class compiled program,

```
linux4:~$ java myApp/test
```

Alternatively, you can do

```
linux4:~$ java myApp.test
```

It is also important to note that a statement such as

import myApp.;*

does **not** import classes from myApp/shared/. You explicitly need to say:

import myApp.shared.;*

```
linux4:~$ find myApp
myApp
myApp/test.java
myApp/test.class
myApp/test1
myApp/test1/T1.java
myApp/test1/T1.class
myApp/shared
myApp/shared/Util.class
myApp/shared/Util.java
myApp/test2
myApp/test2/T2.class
myApp/test2/T2.java

linux4:~$ java myApp.test
Thu Sep 13 05:45:12 EDT 2012
T1
from T1
Thu Sep 13 05:45:13 EDT 2012
T2
linux4:~$ java myApp/test
Thu Sep 13 05:45:20 EDT 2012
T1
from T1
Thu Sep 13 05:45:20 EDT 2012
T2
linux4:~$
```

Also, note that

```
linux4:~/myApp$javac *.java
```

will not work:

```
linux4:~/myApp$ javac test.java
test.java:2: error: package myApp.test1 does not exist
import myApp.test1.*;
^
test.java:3: error: package myApp.test2 does not exist
import myApp.test2.*;
^
.......
```