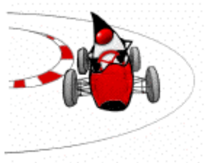
Java Programming 2 – Lecture #11 – Jeremy.Singer@glasgow.ac.uk



Generic Classes

As we saw last week, the Java Collections Framework uses *type parameters* to allow the data structures to be specialized for particular element types, e.g. `ArrayList<String>`. It is also possible to define your own generic classes[1] with type parameters. In the example below, we will define a `Pair` class with a single type parameter *T*. Note how *T* is specified in angle brackets at the start of the class declaration. Then this type parameter can be used as a reference type within the scope of the class body.

```
public class Pair<T> {
  private T first;
  private T second;

  public T getFirst() {
    return this.first;
  }

  public void setFirst(T first) {
    this.first = first;
  }
}
```

Note that type parameters may be *constrained* in terms of the object-oriented inheritance hierarchy[2]. For instance, suppose the `Pair` generic class above should only be allowed to store `java.lang.Number`[3] types, we would specify this as:

```
public class Pair<T extends Number> {…}
```

Generics are a compile-time feature of Java. They are useful for compile-time type checking. However generics are *erased* before runtime. This has several important implications:

- `static` members are shared across all specialized versions of a generic class
- there is no way of distinguishing between types of generic classes using `instanceof` or Java reflection facilities at runtime

[1] See http://docs.oracle.com/javase/tutorial/java/generics/types.html for full details
[2] See http://docs.oracle.com/javase/tutorial/java/generics/bounded.html for details
[3] `Number` is a superclass of the library numeric types, see http://docs.oracle.com/javase/7/docs/api/java/lang/Number.html

Packages in Java

Java packages[4] are a unit of modularity. A package is used to group together a set of related resources (generally Java classes). Use the `package` keyword at the top of a Java source code file to specify the package to which a class belongs. Generally, a class in package Foo should be stored in directory Foo on the filesystem. If no package is specified, then the class belongs to the default package, which is the current working directory.

A fully-qualified classname includes its package, e.g. `java.lang.String` or `java.util.ArrayList`. However a class may be referred to without its package name if the `import` statement is used. This statement opens the namespace of the imported package to the current scope.

```
package a;
public class A { … }
```

```
package b;
public class B {  A.a … }
```

```
package c;
import a.A;
import b.B;
public class C { A … B … }
```

Classes in the current package do not require their fully-qualified names. Also the `java.lang` package is imported implicitly.

Package Naming Conventions

In theory, every Java class defined by any software developer should have a globally unique name. To accomplish this, there is a standard convention[5] for naming packages. Developers use their associated internet domain name in reverse form, followed by a locally unique suffix. So for instance, for JP2 example programs, I might use the package `uk.ac.glasgow.dc.jp2`

Questions

To which packages do the following classes belong?

1. `Scanner`
2. `FileNotFoundException`
3. `List`
4. `Boolean`

[4] See http://docs.oracle.com/javase/tutorial/java/package/ for a good tutorial on packages
[5] See http://en.wikipedia.org/wiki/Java_package#Package_naming_conventions