CS 2340 Objects and Design - Scala

Scala Programs

Christopher Simpkins
chris.simpkins@gatech.edu

Scala Programs and Libraries

Scala code can be delivered as a program or a library.

- Two kinds of scala programs: scripts and applications
  - A Scala script is a file containing Scala code, the last line of which is en executable expression or statement
  - A Scala application is an object, defined with Scala's `object` keyword, that has a method with the signature `def main(args: Array[String])` (name of parameter doesn't have to be args, but has to be of type `Array[String]`)
- A library is a `.jar` file containing a tree of Scala classes
  - Technically, a `.jar` file contains a tree of JVM (Java Virtual Machine) bytecode classes that could be compiled from any source language, such as Java, Scala, Jython, Groovy, Clojure, etc.

To compile and run Scala code you need JRE 1.5 or higher (1.6 recommended).

Running a Scala Script

Given a file `foo.scala` with the following content:

```
class Writer(val repetitions: Int = 1) {

  def say(text: String) {
    sayRepeatedly(text, repetitions)
  }
}

def sayRepeatedly(text: String, times: Int) {
  println(text*times)
}

val repetitions = if (args.length > 0) args(0).toInt else 1
val writer = new Writer(repetitions)
writer.say("foo.scala run as a Scala script.")
```

we can run it like this:

```
[chris@nijinsky ~/examples]
$ scala foo.scala
foo.scala run as a Scala script.
```

Scala Scripts as Shell Scripts

Add following lines to top of Scala script to turn into a Unix shell script:

```
#!/bin/sh
exec scala "$0" "$@"
!#
```

and run it like this:

```
[chris@nijinsky ~/examples]
$ chmod +x foo.scala
[chris@nijinsky ~/examples]
$ ./foo.scala
foo.scala run as a Scala script.
```

On Windows you can acheive the same effect by giving your file a .bat extension and putting this at the top:

```
::#!
@echo off
call scala %0 %*
goto :eof
::!#
```

(So I'm told – I haven't tested this myself since I don't use Windows.)

Compiling Scala Code

Let's try compiling our `foo.scala file`:

```
[chris@nijinsky ~/examples]
$ scalac foo.scala
foo.scala:8: error: expected class or object definition
def sayRepeatedly(text: String, times: Int) {
^
foo.scala:12: error: expected class or object definition
val repetitions = if (args.length > 0) args(0).toInt else 1
^
foo.scala:13: error: expected class or object definition
val writer = new Writer(repetitions)
^
foo.scala:14: error: expected class or object definition
writer.say("foo.scala run as a Scala script.")
^
four errors found
```

We have to remove the executable expressions and statements and move the function sayRepeatedly inside a class or object. Compiling is for class and object definitions.

Compiling Scala Applications

Here's `foo2.scala`, which defines a Scala *application* that does the same thing as our script.

```
object Example {
  def main(args: Array[String]) {
    val repetitions = if (args.length > 0) args(0).toInt else 1
    val writer = new Writer(repetitions)
    writer.say("Example object run as a Scala application.")
  }
}
class Writer(val repetitions: Int = 1) {
  def sayRepeatedly(text: String, times: Int) {
    println(text*times)
  }
  def say(text: String) {
    sayRepeatedly(text, repetitions)
  }
}
```

Compiling and Running Scala Applications

Given our

```
[chris@nijinsky ~/examples]
$scalac foo2.scala
[chris@nijinsky ~/examples]
$ ls
Example$.class Example.class Writer$.class Writer.class
foo.scala foo2.scala
```

Now we can run this Scala application like this:

```
[chris@nijinsky ~/examples]
$ scala -cp . Example
Example object run as a Scala application.
```

The Classpath

Here, we compile classes to another directory with `-d` option to `scalac` so we don't clutter our working directory.

```
[chris@nijinsky ~/examples]
$ mkdir classes
[chris@nijinsky ~/examples]
$ scalac -d classes foo2.scala
[chris@nijinsky ~/examples]
$ ls classes/
Example$.class Example.class Writer$.class Writer.class
```

Specify classpath for an application with the `-cp` option to `scala`.

```
[chris@nijinsky ~/examples]
$ scala -cp ./classes Example
Example object run as a Scala application.
```

There's a global classpath specified by `$CLASSPATH` and a local classpath specified by the `-cp` argument to `scala`. The total classpath of a Scala application is the global + local classpaths.

Libraries

In typical Scala projects, you'll keep source directories and compiler output (.class files) separate. You'll also have external libraries.

- Put source code in a `src` directory
- Put compiler output in a `classes` directory
- Put external libraries in a `lib` directory

Let's reorganize our Scala application.

- Put our Writer class in `src/mypackage/Writer.scala`
- Put Example object in `src/mypackage/Example.scala` directory