类和对象

一、实验介绍

1.1 实验内容

有了前面的Scala基础，从本课程开始由浅到易逐步介绍Scala编程的各个方面。本课程可能不会面面俱到，但仍然希望学习此课程的同学有些编程基础，尤其是有些面向对象的编程基础，如Java，C++，C#等更好。

除支持函数化编程外，Scala也是一个纯面向对象的编程语言。本课程将介绍Scala的类和对象。

1.2 实验知识点

- 类和对象的定义
- 对象详解

1.3 实验环境

- Scala 2.11.7
- Xfce 终端

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合零基础或具有 Java 编程基础的用户。

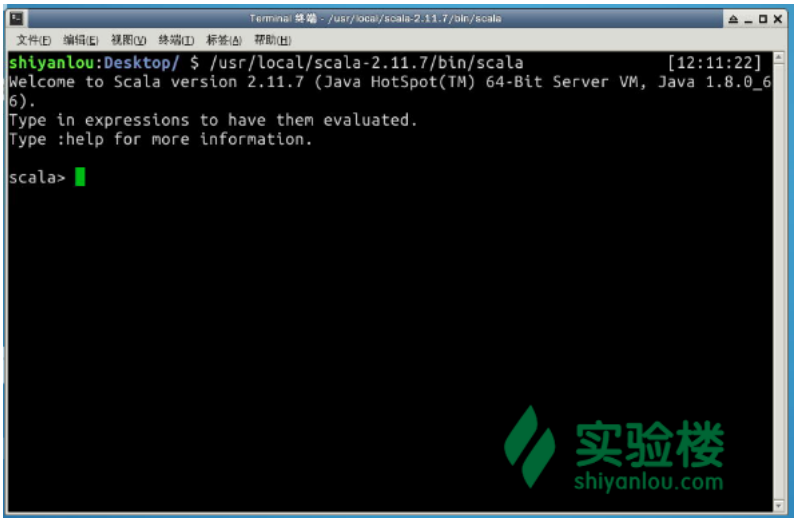
二、开发准备

为了使用交互式 Scala 解释器，你可以在打开的终端中输入命令：

```
cd /usr/local/scala-2.11.7/bin/

scala
```

当出现 `scala>` 开始的命令行提示符时，就说明你已经成功进入解释器了。如下图所示。



三、实验步骤

3.1 类和对象的定义

首先介绍 Scala 的类定义，我们以一个简单的例子开始，创建一个计算整数累计校验和的类 `ChecksumAccumulator` 。

```
class ChecksumAccumulator{
   private var sum=0
   def add(b:Byte) :Unit = sum +=b
   def checksum() : Int = ~ (sum & 0xFF) +1
}
```

可以看到 Scala 类定义和 Java 非常类似，也是以 `class` 开始，和 Java 不同的，Scala 的缺省修饰符为 `public` ，也就是如果不带有访问范围的修饰符 `public` 、 `protected`、 `private` 等，Scala 将默认定义为 `public` 。类的方法以 `def` 定义开始，要注意的Scala的方法的参数都是 `val` 类型，而不是 `var` 类型，因此在函数体内不可以修改参数的值，比如：如果你修改 `add` 方法如下：

```
def add(b:Byte) :Unit ={
      b=1
      sum+=b
  }
```

此时编译器会报错：

```
/root/scala/demo.scala:5: error: reassignment to val
b=1
^
one error found
```

类的方法分两种，一种是有返回值的，一种是不含返回值的，没有返回值的主要是利用代码的“副作用”，比如修改类的成员变量的值或者读写文件等。Scala 内部其实将这种函数的返回值定为 `Unit`（类同 Java 的 void 类型），对于这种类型的方法，可以省略掉 `=` 号，因此如果你希望函数返回某个值，但忘了方法定义中的 `=` ，Scala 会忽略方法的返回值，而返回 `Unit` 。

再强调一下，Scala 代码无需使用 ； 结尾，也不需要使用 `return` 返回值，函数的最后一行的值就作为函数的返回值。

但如果你需要在一行中书写多个语句，此时需要使用 ； 隔开，不过不建议这么做。你也可以把一条语句分成几行书写，Scala 编译器大部分情况下会推算出语句的结尾，不过这样也不是一个好的编码习惯。

3.2 对象

Scala 比 Java 更加面向对象，这是因为 Scala 不允许类保护静态元素(静态变量或静态方法）。在 Scala 中提供类似功能的是成为“ `Singleton`（单例对象）”的对象。在 Scala 中定义 `Singleton` 对象的方法除了使用 `object` ，而非 `class` 关键字外和类定义非常类似。下面例子创建一个 `ChecksumAccumulator` 对象：

```
object ChecksumAccumulator {
   private val cache = Map [String, Int] ()
   def calculate(s:String) : Int =
      if(cache.contains(s))
         cache(s)
      else {
         val acc=new ChecksumAccumulator
          for( c <- s)
             acc.add(c.toByte)
          val cs=acc.checksum()
         cache += ( s -> cs)
         cs
       }
}
```

这个对象和上一个创建的类 ChecksumAccumulator 同名，这在 Scala 中把这个对象成为其同名的类的“伴侣”对象（Companion object)。 如果你需要定义的类的 companion 对象，Scala 要求你把这两个定义放在同一个文件中。类和其 companion 对象可以互相访问对方的私有成员。

如果你是 Java 成员，可以把 Singleton 对象看成以前 Java 定义静态成员的地方。你可以使用类似 Java 静态方法的方式调用 Singleton 对象的方法，比如下面为这个例子完整的代码：

```
import scala.collection.mutable.Map
class ChecksumAccumulator{
   private var sum=0
   def add(b:Byte) :Unit = sum +=b
   def checksum() : Int = ~ (sum & 0xFF) +1
}

object ChecksumAccumulator {
   private val cache = Map [String, Int] ()
   def calculate(s:String) : Int =
      if(cache.contains(s))
         cache(s)
      else {
         val acc=new ChecksumAccumulator
         for( c <- s)
            acc.add(c.toByte)
         val cs=acc.checksum()
         cache += ( s -> cs)
         cs
       }
}

println ( ChecksumAccumulator.calculate("Welcome to Scala Chinese community"))
```

Scala 的 singleton 对象不仅限于作为静态对象的容器，它在 Scala 中也是头等公民，但仅仅定义 Singleton 对象本身不会创建一个新的类型，你不可以使用 new 再创建一个新的 Singleton 对象（这也是 Singleton 名字的由来），此外和类定义不同的是， singleton 对象不可以带参数（类定义参数将在后面的内容中介绍）。

回过头来看看我们的第一个例子 “Hello World” 。

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello, world!")
  }
}
```

这是一个最简单的 Scala 程序， HelloWorld 是一个 Singleton 对象，它包含一个 main 方法（可以支持命令行参数）。和 Java 类似， Scala 中任何 Singleton 对象，如果包含 main 方法，都可以作为应用的入口点。

在这里要说明一点的是，在 Scala 中不要求 public 类定义和其文件名同名，不过使用和 public 类和文件同名还是有它的优点的，你可以根据个人喜好决定是否遵循 Java 文件命名风格。

最后提一下 Scala 的 Trait 功能，Scala 的 Trait 和 Java 的 Interface 相比，可以有方法的实现（这点有点像抽象类，但如果是抽象类，就不会允许继承多个抽象类）。Scala 的 Trait 支持类和 Singleton 对象和多个 Trait 混合（使用来自这些 Trait 中的方法，而不是不违反单一继承的原则）。

Scala 为 Singleton 对象的 main 定义了一个名为 App 的 trait 类型，因此上面的例子可以简化为：

```
object HelloWorld extends App{
   println("Hello, world!")
}
```

这段代码就不能作为脚本运行，Scala 的脚本要求代码最后以表达式结束。因此运行这段代码，需要先编译这段代码：

```
scalac Helloworld.scala
```

编译好之后，运行该应用。

```
scala HelloWord
```

注意： Scala 提供了一个快速编译代码的辅助命令 `fsc` (fast scala compliler) ，使用这个命令，只在第一次使用 `fsc` 时启动JVM，之后 `fsc` 在后台运行，这样就避免每次使用 `scalac` 时都要载入相关库文件，从而提高编译速度。

四、实验总结

在本节实验中，我们学习了 Scala 中的类和对象的相关知识。在下一节，我们将学习 Scala 中的基本数据类型及其操作。拥有了一部分基础知识后，我们还将回顾一些类和对象的要点。

上一节 (/courses/490/labs/1684/document)

下一节 (/courses/490/labs/1686/document)

课程教师

引路蜂

共发布过**6**门课程

CSDN 专家博主，擅长Java ME, Blackberry ,LWUIT , iPhone, Android, Windows Mobile, Mono , Windows Phone 7等平台开发，主页 http://www.imobilebbs.com/

查看老师的所有课程 > (/teacher/164063)

进阶课程

Scala 专题教程 - Case Class和模式匹配 (/courses/514)

Scala 专题教程 - 隐式变换和隐式参数 (/courses/515)

Scala 专题教程 - 抽象成员 (/courses/516)

Scala 专题教程 - Extractor (/courses/526)