**Entregable #1.1 [ejercicios] [semana 12 Feb 2018]**

* **Objetivos específicos** que persigue la realización del trabajo.
  + - Practicar con conceptos básicos de Scala sobre el entorno Kojo
    - Familiarizarse con el entorno eclipse para el lenguaje Scala
* **Competencias**: CC14

**Apellidos y nombre**

|  |
| --- |
| Fabricio Altamirano |
| Anderson Gutiérrez Ospina |

1. **Familiarización con Scala sobre Kojo**

Abre el entorno "kojo". Si pulsas en la pestaña "Help", verás una entrada etiquetada "Scala Tutorial". Selecciona dicha entrada.

Para familiarizarte con los conceptos básicos del lenguaje, leerás las 5 primeras secciones del Tutorial e irás ejecutando los ejemplos que aparecen. Estarás trabajando con el intérprete de Scala, por lo que el código que introduzcas en la ventana de Script Editor del entorno, se ejecutará y generará la salida correspondiente en la ventana de Output Pane.

Cuando hayas leído la sección "Getting started", escribe en la ventana del editor estas sentencias (especificando tu altura) y ejecútalas:

val altura = 1.64

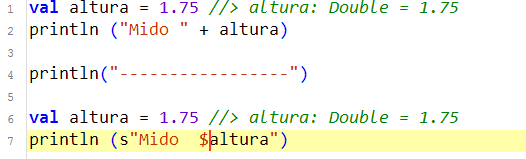
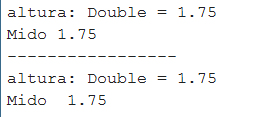
println ("Mido " + altura)

A continuación, sustituye la sentencia println anterior por la siguiente, y fíjate bien en la sintaxis de esta última:

println(s"Mido $altura")

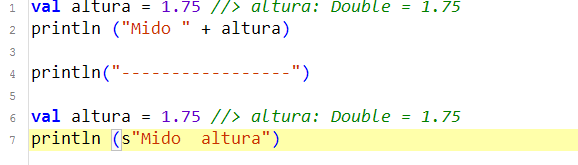
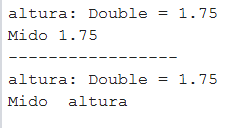
1.1.1 ¿Producen ambas sentencias println la misma salida?

Si, se trata de otra forma de concatenar cadenas



1.1.2 ¿Qué pasa si quitas el "$" que precede a la variable "altura"? ¿Sigue imprimiendo lo mismo?

No, porque no reconoce la sintaxis para imprimir el valor de la variable

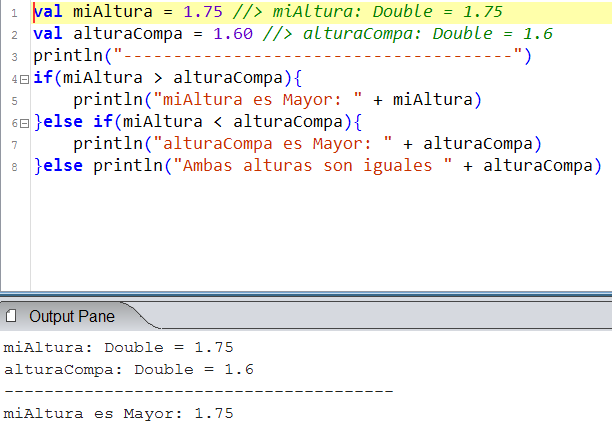


Después de leer la sección "Flow control: If, Else and While", escribe en la ventana del editor un trozo de código que haga lo siguiente:

- Define 2 variables: "miAltura" y "alturaCompa", que contengan tu altura y la de tu compañer@ de laboratorio, respectivamente

- Escribe una sentencia "if" que escriba por pantalla un mensaje indicando cuál de los dos es más alto, o si medís lo mismo.

1.1.3 Una vez probado que dicho código funciona, transcribe aquí las líneas que has escrito:



**val** miAltura = 1.75

**val** alturaCompa = 1.60

println("-------------------------------")

**if** (miAltura > alturaCompa) {

println("miAltura es Mayor: " + miAltura)

} **else** **if** (miAltura < alturaCompa) {

println("alturaCompa es Mayor: " +

alturaCompa)

} **else** println("Ambas alturas son iguales "

+ alturaCompa)

**2. Familiarización con Scala sobre Eclipse**

Vamos a familiarizarnos con la introducción de código Scala en el entorno eclipse a través de un programa muy sencillo:

object helloScala {

def main(args: Array[String]) {

println ("Bienvenidos a Scala")

}

}

Ahora ya no estaremos trabajando directamente con el intérprete de Scala, como hacíamos en Kojo, aunque podemos acceder al intérprete desde Eclipse si lo necesitamos. En general, lo que haremos ahora es incluir todo el código del programa para poder ejecutarlo.

Para ello, arranca el entorno Eclipse y sigue los siguientes pasos:

* Crea un proyecto con tu nombre a través de los siguientes pasos:

File -> New -> Scala Project

Project name: nombreEstudiante

* Situado sobre el proyecto recién creado, crea un nuevo objeto:

File -> New -> Scala Object

Name: helloScala

* En la ventana de código del entorno, aparecerá el esqueleto del nuevo objeto. Complétalo para obtener el programa mostrado arriba. Una vez completo, salva su contenido.

A continuación, ejecútalo (Run -> Run as -> scala application). Comprueba que en la consola de salida muestra la salida esperada.

En realidad, para que se ejecute el procedimiento “println” habría que importar una biblioteca (o librería) disponible en el lenguaje, pero esto se hace automáticamente con algunas bibliotecas. Scala importa implícitamente los paquetes “java.lang”, “scala” y “Predef”. Algunos ejemplos de objetos importados son:

Predef.println

java.lang.String

scala.Int

scala.Char

scala.Double

Gracias a la importación implícita de los paquetes citados, podemos usar estos objetos sin necesidad de indicar de dónde provienen. Así, el tipo entero se identificará con “Int” en vez de “scala.Int”, sin indicar explícitamente qué paquete lo contiene.

En la sección “Funciones” del Tutorial de Scala que has revisado con el entorno Kojo, había una función max para calcular el máximo de dos números. Trae el código de dicha función a Eclipse y crea un programa que, usando dicha función, obtenga el máximo de los dos argumentos que se pasen en la llamada y los imprima por pantalla.

1.1.4 Una vez probado y funcionando, transcribe a continuación el código del programa que has ejecutado para llamar a la función max.

**object** helloScala {

**def** max(x: Int, y: Int): Int = {

**if** (x > y) x

**else** y

}

**def** main(args: Array[*String*]) {

println ("Bienvenidos a Scala")

**val** num = max(5,10)

println(s"El maximo entre 5 y 10 es ==> $num")

}

}

A continuación, se proporciona la definición de la clase Point del apartado “Objects and Classes” de Kojo. Además, se proporciona un programa que usa dicha clase.

case class Point(x:Int,y:Int){

def +(newpt:Point)=Point(x+newpt.x,y+newpt.y)

def -(newpt:Point)=Point(x-newpt.x,y-newpt.y)

override def toString="Point("+x+","+y+")"

}

object PPoint {

def main(args: Array[String]) {

val p1=Point(3,4)

val p2=Point(7,2)

val p3=Point(-2,2)

val p4=p1+p2-p3

println(p4)

}

}

1.1.5 Indica qué hace el programa y por qué se imprime correctamente el contenido de p4 sin ser un String.

* + - * Suma todos los valores de (x) y todos los valores de (y), desde el objeto PPoint que llama a la clase Point.
      * El contenido de P4 se imprime porque la función println tiene sobrecarga de operadores (sobrecarga de métodos)

Para responder a las siguientes preguntas, lee la sección titulada “Case clases and pattern matching” del libro “*Programming in Scala – Third Edition*”*.* Dicho libro puede accederse desde el enlace de la sección de Prácticas de la página Moodle de la asignatura o directamente en http://proquest.safaribooksonline.com/book/programming/scala/9780981531687

1.1.6 Indica cómo es posible que el programa anterior no dé errores de compilación siendo que no están definidos en la clase las variables x e y, y sus métodos de acceso.

* Porque se trata de una **case class** que que se utiliza para modelar datos inmutables, y las variables x e y ya están implícitamente definidos como val. Los métodos son funciones que se ejecutan dentro del contexto de un objeto y que tienen acceso a todos los campos del objeto.

1.1.7 Explica también cómo es posible que para crear una instancia de la clase Punto no se invoque al constructor new.

* Esto se debe a que las case class tienen un método por defecto que se ocupa de la creación del objeto