

---

# Prácticas Scala

## 1. Variables y constantes

- Acceder a REPL, la línea de comandos de escala.
  - Crear 2 val, llamadas nombre y apellidos y rellenarlas con ese valor
  - Visualizarlas por pantalla, por separada y concatenadas
  - Crear un “var” llamada “edad” y rellenarla con algún valor
  - Visualizarla concatenada con el nombre y apellidos
  - Intentar cambiar el nombre o el apellido
  - Intentar cambiar la edad y visualizarla
- Hay una librería denominada `util.Properties` que contiene una gran cantidad de funciones muy útiles para Scala
  - Una de ellas es “`javaVersion`”, que devuelve la versión de Java con la que trabajamos.
  - Poner el valor en una “val” denominada `versión_java` y visualizarla por pantalla
- Cuando estamos en REPL, podemos ir viendo opciones si tabulamos.
  - Por ejemplo si ponemos “`util.Properties.`” y tabulamos nos da todas la opciones disponibles

The screenshot shows a Scala REPL window with two panes. The left pane shows the command `scala> util.Properties.` followed by a list of methods: `getClass()`, `synchronized()`, `versionMsg`, `javaSpecName`, `javaVmInfo`, `propIsSetTo()`, `main()`, `javaSpecVendor`, `javaVmVendor`, `jdkHome`, `javaClassPath`, `envOrElse()`, `propOrElse()`, `→()`, `scalaHome`, `envOrElse()`, `==(`, `versionString`, `sourceReader`, `+()`, `setProp()`, `eq()`, `isWin`, and `scala> util.Properties.`. The right pane shows the same command followed by a list of methods: `ne()`, `sourceEncoding`, `isInstanceOf()`, `isMac`, `javaSpecVersion`, `!=(`, `asInstanceOf()`, `propOrElse()`, `scalaCmd`, `wait()`, `developmentVersion`, `formatted()`, `javaVersion`, `javaVmVersion`, `notifyAll()`, `encodingString`, `lineSeparator`, `notify()`, `propOrElse()`, `tmpDir`, `scalaPropOrElse()`, `propOrElse()`, and `userHome`. The command `scala> util.Properties.` is highlighted in the left pane.

- 
- Hay una serie de funciones de java que comienzan por “util,Properties.javaXXXXX”.

```
scala> util.Properties.java
javaClassPath      javaSpecVendor    javaVersion        javaVmVendor
javaHome           javaSpecVersion   javaVmInfo         javaVmVersion
javaSpecName       javaVendor        javaVmName
```

- Con esta información averiguar y visualizar los siguientes datos
  - Java Vendor
  - Java VM Name
- Intentar crear una variable con la siguiente expresión.

var resultado=10/0

- Debe arrojar un error
- Con una cláusula lazy retrasar el error hasta usarla
- Con el comando :save guardar el resultado en un fichero denominado ejer1.scala
- Comprobar el contenido en el sistema Operativo
- Salir del REPL
- Ejecutar el script que hemos creado en el paso anterior
-

---

## 2. Variables y constantes (SOLUCIONES)

- Acceder a REPL, la línea de comandos de scala.
  - Crear 2 val, llamadas nombre y apellidos y rellenarlas con ese valor

```
scala> val nombre="Pedro"

val nombre: String = Pedro

scala> val apellidos="Rodriguez"

val apellidos: String = Rodriguez
```

- Visualizarlas por pantalla, por separada y concatenadas

```
scala> print(nombre)

Pedro

scala> print(apellidos)

Rodriguez

scala> print(apellidos+" "+apellidos)

Rodriguez Rodriguez
```

- Crear un “var” llamada “edad” y rellenarla con algún valor

```
scala> var edad=30

var edad: Int = 30
```

- Visualizarla concatenada con el nombre y apellidos

```
print(apellidos+" "+apellidos+" "+edad)

Rodriguez Rodriguez 30
```

- Intentar cambiar el nombre o el apellido

```
scala> nombre="Rosa"

      ^

error: reassignment to val
```

- 
- Intentar cambiar la edad y visualizarla

```
scala> edad=40

// mutated edad

scala> print(edad)

40
```

- Hay una librería denominada `util.Properties` que contiene una gran cantidad de funciones muy útiles para Scala
  - Una de ellas es `javaVersion`, que devuelve la versión de Java con la que trabajamos.
  - Poner el valor en una `val` denominada `versión_java` y visualizarla por pantalla

```
scala> val version_java="La versión de Java es
"+util.Properties.javaVersion

val version_java: String = La versión de Java es 1.8.0_272

scala> println(version_java)

La versión de Java es 1.8.0_272
```

- Cuando estamos en REPL, podemos ir viendo opciones si tabulamos.
  - Por ejemplo si ponemos `util.Properties.` y tabulamos nos da todas la opciones disponibles

```

usu1@curso:~/practicas x          usu1@curso:~/idea x
getClass() (universal) ne( (universal) ScalaCompilerVersion
synchronized( (universal) sourceEncoding hashCode() (univer
versionMsg isInstanceOf (universal) propIsSet( (univer
javaSpecName isMac envOrElse( (univer
javaVmInfo javaSpecVersion ensuring( (univer
propIsSetTo( !=( (universal) versionNumberString
main( asInstanceOf (universal) ## (univer
javaSpecVendor propOrElse( isLinux
javaVmVendor scalaCmd releaseVersion
jdkHome scalacCmd javaVmName
javaClassPath wait() (universal) javaVendor
envOrElse( developmentVersion osName
propOrElse( formatted( (universal) userName
+ ( (deprecated) javaVersion clearProp(
scalaHome javaVmVersion copyrightString
envOrElse( notifyAll() (universal) javaHome
==( (universal) encodingString scalaPropOrElse(
versionString lineSeparator equals( (univer
sourceReader notify() (universal) isJavaAtLeast(
+ ( (deprecated) propOrElse( userDir
setProp( tmpDir toString() (univer
eq( (universal) scalaPropOrElse( propOrElse(
isWin propOrElse( scalaPropOrElse(
scala> util.Properties.[] userHome

```

- Hay una serie de funciones de java que comienzan por "util,Properties.javaXXXXX".

```

scala> util.Properties.java
javaClassPath javaSpecVendor javaVersion javaVmVendor
javaHome javaSpecVersion javaVmInfo javaVmVersion
javaSpecName javaVendor javaVmName

```

- Con esta información averiguar y visualizar los siguientes datos

```

Java Vendor

Java VM Name

scala> println("El vendor es "+util.Properties.javaVendor)

El vendor es Red Hat, Inc.

scala> println("El vendor es "+util.Properties.javaVmName)

El vendor es OpenJDK 64-Bit Server VM

```

- Intentar crear una val con la siguiente expresión.

```
resultado=10/0
```

- Debe arrojar un error

```

scala> val resultado=10/0

java.lang.ArithmeticException: / by zero

```

---

... 32 elided

- Con una cláusula lazy retrasar el error hasta usarla

```
scala> lazy val resultado=10/0

lazy val resultado: Int // unevaluated

scala> resultado

java.lang.ArithmeticException: / by zero

    at resultado$lzycompute(<console>:1)
    at resultado(<console>:1)

... 32 elided
```

- Con el comando :save guardar el resultado en un fichero denominado ejer1.scala

```
scala> :save ejer1.scala

scala> :quit
```

- Comprobar el contenido en el sistema Operativo
- Salir del REPL
- Ejecutar el script que hemos creado en el paso anterior
-