Módulo Profesional: Big Data Aplicado

Scala

Object Oriented Basics

```
//constructor
class Persona (nombre: String, val edad: Int) {
//cuerpo
val x = 2 //son campos
println (1 + 3) //expresiones
//método --//overloading
def saluda(nombre: String): Unit = println(s"${this.nombre} dice: Hola, $nombre")
def saluda(): Unit = println (s"Hola, Yo soy $nombre")
//def saluda(): Int = 43 --> En este caso el compilado sí se confude

//multiple constructores
def this(nombre: String) = this(nombre,0)
def this() = this ("Juan Carlos")
}
```

Constructor

```
//constructor
class Persona (nombre: String, val edad: Int) {
```

Se declara una clase Persona con dos parámetros:
- Nombre de tipo String: parámetro privado sólo accesible desde la clase
- Edad de tipo entero: parámetro público accesible desde fuera de la clase por “val”.

Cuerpo

```
//cuerpo
val x = 2 //son campos
println (1 + 3) //expresiones
```

val x = 2 → se define una variable inmutable con el valor 2 como propiedad de la clase.
Println (1 + 3) → se define una expresión que imprime el valor 4.

Cuerpo : Sobrecarga

```
//método --//overloading
def saluda(nombre: String): Unit = println(s"${this.nombre} dice: Hola, $nombre")
def saluda(): Unit = println (s"Hola, Yo soy $nombre")
def saluda(): Int = 43 --> En este caso el compilado sí se confude
```

Las dos primeras funciones se denominan igual pero con diferencias:
Def saluda (nombre:String)...-→ declara un argumento y devuelve una interpolación de cadenas para incluir variables de texto. ${this.nombre} accede al atributo privado nombre de la clase.
Def saluda () que no requiere argumentos y utiliza el nombre del objeto.
De saluda () que devuelve un entero el compilador no sabe distinguir entre la segunda función y la tercera función.

Cuerpo : Múltiples constructores

```
//multiple constructores
def this(nombre: String) = this(nombre,0)
def this() = this ("Juan Carlos")
```

def this (nombre: String) = this(nombre, 0) → permite crear el objeto sólo con el nombre, asignado la edad por defecto como 0.
def this() = this (·"Juan Carlos") → permite crear un objeto sin argumentos, asignando Juan Carlos como nombre y 0 como edad

Instancia de la clase Persona

```
val persona = new Persona ("Carlos", 25)
println(persona.edad)
println(persona.x)
persona.saluda("Daniel")
persona.saluda()
```

Val persona = new Persona(“Carlos” ,25) → objeto de la clase Persona con el nombre Carlos y edad 25. Y con val declara una variable inmutable que almacena la instancia.
Println (persona.edad) → accede a la propiedad pública edad que se declaró con val en el constructor.

¿Qué imprime el método saluda?

```
val persona = new Persona ("Carlos", 25)
println(persona.edad)
println(persona.x)
persona.saluda("Daniel")
persona.saluda()
```

```
def saluda(nombre: String): Unit = println(s"${this.nombre} dice: Hola, $nombre"
def saluda(): Unit = println (s"Hola, Yo soy $nombre")
```

Persona.saluda (“Daniel”) → ¿Qué imprime?
Persona.saluda() → ¿Qué imprime?

Resultado de la impresión del método saluda

```
val persona = new Persona ("Carlos", 25)
println(persona.edad)
println(persona.x)
persona.saluda("Daniel")
persona.saluda()
```

```
def saluda(nombre: String): Unit = println(s"${this.nombre} dice: Hola, $nombre"
def saluda(): Unit = println (s"Hola, Yo soy $nombre")
```

Persona.saluda (“Daniel”) → Carlos dice: Hola, Daniel
Persona.saluda() → Hola, Yo soy Carlos.

Resumen

```
class Persona(nombre: String, edad:Int) //Definición de clases
val persona = new Persona("Pedro", 25) //Instancia
class Persona(val nombre: String, edad: Int) //Parámetros vs Campos

def saluda(): String = {....} //Definición de métodos
val personaSaluda = persona.saluda //Llamada a métodos

//sintaxis para métodos sin parámetros
//la palabra clave this
```

Más ejemplos

```
/* Novela y Escritor
  Escritor: nombre, primer apellido, año
   - método nombrecompleto

  Novela: nombre, año de realización, autor
   - Edad del autor
   - EscritoPor (autor)

*/
```

Más ejemplos

```
class Escritor (nombre: String, primerApellido: String, val año: Int){
    def nombreCompleto: String = nombre + " " + primerApellido

}
```

Clase Escritor:
- nombre y primerApellido: strings privados
- año: un valor público (usando val)

Más ejemplos

```
class Novela (nombre: String, año: Int, autor: Escritor) {
  def edadAutor = año - autor.año
  def EscritoPor(autor:Escritor) = autor == this.autor

}
```

Clase Novela:
- nombre: el título de la novela
- año: año de publicación
- autor: instancia de la clase escritor

EdadAutor: calcula la edad que tenía el autor cuando escribió la novela
EscritoPor: devuelve verdadero si el autor es el mismo que escribió la novela

Más ejemplos: ¿Qué imprime por pantalla?

```
val autor = new Escritor("Carlos", "Herranz", 1920)
val novela = new Novela ("Gran Novela", 1930, autor)

println(novela.edadAutor)
println(novela.EscritoPor(autor))
```

```
class Novela (nombre: String, año: Int, autor: Escritor) {
  def edadAutor = año - autor.año
  def EscritoPor(autor:Escritor) = autor == this.autor

}
```

Ejercicio

```
/* Clase contador:
    Parámetros:
        - recibe un valor entero por defecto 0
    Funciones:
        - función para incrementar  + 1 el contador
        - función para decrementar - 1 el contador
        - función de imprimir por pantalla el contador

*/
```