Módulo Profesional: Big Data Aplicado

Scala

Method Notations

```
class Persona (val nombre: String, peliculaFavorita: String){
  def meGusta (pelicula: String): Boolean = pelicula == peliculaFavorita
```

```
val maria = new Persona ("Maria","Los vengadores")
//NOTACIONES INFIJOS = OPERADORES
println(maria.meGusta("Los vengadores")) //True
println(maria meGusta "Los vengadores") //True
```

1. Definir una clase Persona con dos parámetros nombre y peliculaFavorita.
2. Definir un método meGusta
3. Instanciar la clase persona
4. Imprimir por pantalla

Method Notations

```scala
class Persona (val nombre: String, peliculaFavorita: String){
  def meGusta (pelicula: String): Boolean = pelicula == peliculaFavorita
  def salirCon (persona: Persona ): String = s"${this.nombre} sale con ${persona.nombre}"
```

```scala
// operadores en Scala
val pedro = new Persona ("Pedro", "Tiburón")
println (maria.salirCon(pedro))
println (maria salirCon pedro)
```

5. Definimos método salirCon dentro de la clase Persona
6. Nueva instancia de la clase
7. Imprime pantalla

Method Notations

```scala
class Persona (val nombre: String, peliculaFavorita: String){
  def meGusta (pelicula: String): Boolean = pelicula == peliculaFavorita
  def salirCon (persona: Persona ): String = s"${this.nombre} sale con ${persona.nombre}"
  def + (persona: Persona ): String = s"${this.nombre} no le gusta ${persona.nombre}" //método llamado + es válido
```

```scala
// operadores en Scala
val pedro = new Persona ("Pedro", "Tiburón")
println (maria + pedro)
println (maria.+(pedro))
```

8. Nuevo método denominado +
9. Imprimir pantalla

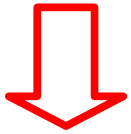
Method Notations

TODOS LOS OPERADORES SON MÉTODOS

```
//TODOS LOS OPERADORES SON MÉTODOS
println(1 + 2)
println (1.+(2))
```

Method Notations

```
def unary_! : String = s"$nombre, que es esto?" //debe haber un espacio entre unary_! y los dos puntos
```



```
//NOTACION PREFIJOS
println(!maria)
println(maria.unary_!)
```

Method Notations

```scala
//NOTACION más PREFIJOS
val x = -1 //equivalente con 1.unary_-
val y = 1.unary_-
val z = 2.unary_+
val j = 3.unary_~
//unary_prefix solo con - + ~ !

println(y) // valor negativo
println(z) //valor positivo
println(j) //El operador ~ tiene sentido en operaciones bit a bit,
//útil en manipulación de bits, como en programación de bajo nivel
// o en el desarrollo de aplicaciones que requieren optimización binaria.
//Este método se encuentra definido para tipos como Int y Long en Scala.
```

Method Notations

```scala
class Persona (val nombre: String, peliculaFavorita: String){
  def meGusta (pelicula: String): Boolean = pelicula == peliculaFavorita
  def salirCon (persona: Persona ): String = s"${this.nombre} sale con ${persona.nombre}"
  def + (persona: Persona ): String = s"${this.nombre} no le gusta ${persona.nombre}" //método llamado + es válido

  def unary_! : String = s"$nombre, que es esto?" //debe haber un espacio entre unary_! y los dos puntos


  def apply(): String = s"Hola, mi nombre es $nombre y me gusta $peliculaFavorita"

}
```

```scala
//apply
println(maria.apply())
println(maria()) //equivalente
```

Ejercicio 1: Mascota

```
/* Ejercicio 1: Mascota y su actividad favorita
 Simula la creación y gestión de una mascota con un nombre y una actividad favorita.
 Datos de la mascota:
   - Nombre
   - Actividad favorita
 Operaciones:
  - Cambiar la actividad favorita a Jugar con la pelota.
  - Mostrar la información actualizada de la mascota.
Clase Mascota:
  - Contiene los parámetros de entrada nombre y actividad favorita
  - Métodos:
    - cambiarActividad(nuevaActividad: String): Cambia la actividad favorita de la mascota.
    - mostrarInfo(): Muestra los detalles de la mascota, incluyendo su nombre y actividad favorita.

*/
```

Ejercicio 2: Tienda

```
Ejercicio TiendaApp:
    Simula las operaciones de venta y reposición en una tienda.
    Datos del producto:
        - nombre: Camiseta
        - precio: 25
        - Cantidad en stock = 20
    Operaciones:
        - Vender 5 camisetas
        - Agregar 10 camisetas al stock
        - Mostrar información actualizada


   Clase Producto:
     - Contiene atributos para el nombre, precio y cantidad en stock.
     - Métodos:
        - vender(cantidad: Int): Reduce el stock si hay suficiente cantidad disponible y la cantidad es mayor que 0.
          Muestra un mensaje de error por pantalla si no hay suficiente stock o si la cantidad es inválida.
        - agregarStock(cantidad: Int): Aumenta el stock si la cantidad es mayor que 0.
          Muestra un mensaje de error por pantalla si la cantidad es inválida.
        - mostrarInfo(): Imprime por pantalla los detalles del producto (Producto, precio y cantidad)
```

Ejercicio 3: Cuenta Bancaria

```
/* Ejercicio 2: Cuenta Bancaria
Simula las operaciones básicas de una cuenta bancaria.
 Datos de la cuenta:
    Número de cuenta: 987654321
   Saldo inicial: 1000
  Operaciones:
    - Depositar 500
    - Retirar 200
    - Intentar retirar 1500 (¡cuidado excede el saldo!)
    - Mostrar información de la cuenta bancaria actualizada
```

Ejercicio 3: Cuenta Bancaria

```
Clase CuentaBancaria:
  - Contiene parámetros de entrada para el número de cuenta, saldo actual de la cuenta.
  - Métodos:
      - depositar(cantidad: Double):
        1.- Incrementa el saldo si la cantidad es mayor que 0.
            Muestra el siguiente mensaje por pantalla con el total depositado y el saldo actual:
              "Se han depositado 500 en la cuenta 987654321. Saldo actual: 1500"
        2.- Muestra el siguiente mensaje por pantalla si la cantidad es menor de 0 o igual a 0:
            " No se puede depositar una cantidad negativa o cero"

        ***Pista: saldo += cantidad

      - retirar(cantidad: Double): Disminuye el saldo si la cantidad es mayor que 0 y
        no excede el saldo disponible.
        1.- Si la cantidad es mayor que saldo muestra el siguiente mensaje por pantalla:
          "No hay suficiente saldo para retirar 1500. Saldo actual: 1300"
        2.- Si la cantidad es menor que saldo muestra el siguiente mensaje por pantalla:
          "Se han retirado 200 de la cuenta 987654321. Saldo actual: 1300"
        3.- Si la cantidad es negativa o 0 muestra el siguiente mensaje por pantalla:
          "No se puede retirar una cantidad negativa o cero"
        ***Pista: saldo -= cantidad
```