Instituto Tecnológico Superior De Lerdo

Aplicaciones Móviles

Práctica: 1.12

Profesor: Ing. Jesús salas Marín

Alumno: Luis Andres Rodriguez Campos

Carrera: Ing. Informática

Sección: A

Grado: 8

Numero de control: 17231573

Creamos una clase con constructores que tomaran un parámetro de tipo string.

```
class InitOrderDemo(name: String) {
    val firstProperty = " Primera propiedad: $name".also(::println)

    init {
        println("Primer bloque inicializador que imprime ${name}")
    }

    val secondProperty = " Segunda propiedad: ${name.length}".also(::println)

    init {
        println("Segundo bloque inicializador que imprime ${name.length}")
    }
}
fun main() {
    InitOrderDemo( name: "hello")
}
```

Nos dará el siguiente resultado.

```
I/System.out:  Primera propiedad: hello
    Primer bloque inicializador que imprime hello
I/System.out:  Segunda propiedad: 5
    Segundo bloque inicializador que imprime 5
```

Ahora crearemos la siguiente clase donde utilizaremos inicializadores declarados en el cuerpo de la clase.

```
class Customer(name: String) {
    val customerKey = name.toUpperCase()
}
```

Creamos una clase para simular una personas y alguna de su información.

```
class Person(
        val firstName: String,
        val lastName: String,
        var age: Int, // trailing comma
) { /*...*/ }
```

Creamos una clase persona y añadiremos su herencia a una clase niño, para que acepte la misma información.

```
class Customer public @Inject constructor(name: String) { /*...*/ }
class Person {
    var children: MutableList<Person> = mutableListOf()
    constructor(parent: Person) {
        parent.children.add(this)
    }
```

Ahora imprimimos los constructores en caso de que estén declarados

```
class Constructors {
    init {
        println("Init block")
    }
    constructor(i: Int) {
        println("Constructor $i")
    }
}
```

En caso de querer un constructor privado utilizamos la siguiente línea

```
class DontCreateMe private constructor () { /*...*/ }
```

Las instancias pueden llamar al constructor como si fueran una función aquí un ejemplo de ello.

```
val invoice = Invoice()
val customer = Customer("Joe Smith")
```