Funciones y datos

Abstracción funcional: Es poder representar conjuntos brindando operaciones sin pensar en cómo se realizan

- Elevar el nivel conceptual
Pensar en alto nivel (no nos preocupamos por los detalles de la implementación)
- Incrementar la modularidad
Podemos reutilizar componentes (escalabilidad)
- Fortalecer el poder expresivo del lenguaje
Podemos agregar tipos de datos que se pueden ser utilzados de forma NATURAL

La abstracción de datos (separación de implementación y comportamiento) es
una piedra angular de la ingeniería de software

require impone una precondición a quien invoca una función
require <== Antes de instanciar
assert verificar una condición que uno quiere que se cumpla en ese punto de la ejecución
assert <== Despues de instanciar

Cualquier método que tenga un sólo parámetro del mismo tipo que la clase (o sea, que pueda ser visto como operador binario), puede ser u como operador infijo.
r .suma(s) puede escribirse r suma s
r .mult(s) puede escribirse r mult s
r .menorQue(s) puede escribirse r menorQue s
r .max(s) puede escribirse r max s

La herencia es en esencia la capacidad de construir nuevas clases (nuevas abstracciones de datos) a partir de clases existentes. O sea, la herencia es un mecanismo para la construcción incremental de abstracciones de datos.
Para este fin, Scala provee:
Clases abstractas
Herencia simple
Rasgos (traits)

Los lenguajes OO (incluyendo a Scala) implementan lo que se denomina despacho dinámico de métodos

Ligadura dinamica

```
abstract class Animal() .... emitirSonido()
class Perro() extends Animal
class Gato() extends Animal

val objAnimal = new Perro() <-- Animal (SI)
objAnimal.emitirSOnido() "guau"
val objAnimal2 = new Gato() <-- Animal (SI
objAnimal2.emitirSonido() "miau"
POLIMORFISMO
```