# Tipos de Clases - Sobrecarga

Seguidamente algunos ejemplos prácticos

### **Ejemplo 1**

**Ejemplo**: tenemos dos funciones 1) calcular el cuadrado de un número y 2) el número elevado a la 4

```
cuadrado.hs ×
{-
Calcular el cuadrado de un número
calcular el valor de un número elevado a la 4 utilizando la función del inciso anterior
-}
cuadrado :: Double -> Double -- una entrada de tipo Double con una salida de tipo Double
cuadrado x = x^2
elevadoCuatro :: Double -> Double -- reutilizamos cuadrado
elevadoCuatro x = cuadrado(cuadrado x)
```

# **Ejemplo 1**

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
GHCi, version 8.8.4: https://www.haskell.org/ghc/ :? for help
Prelude> :cd haskell
Prelude> :l cuadrado.hs
[1 of 1] Compiling Main
                                   ( cuadrado.hs, interpreted )
Ok, one module loaded.
*Main> cuadrado 5
25.0
*Main> elevadoCuatro 5
625.0
*Main> :t elevadoCuatro 5
elevadoCuatro 5 :: Double
*Main> :t cuatro 5
                                                    la función cuatro 5 no está definida
                                                    pero nos indica el tipo de datos
<interactive>:1:1: error:
   Variable not in scope: cuatro :: Integer -> t
                                                   ingresado por teclado
*Main> :t cuadrado 5
cuadrado 5 :: Double
*Main>
```

## Ejemplo 2 - sobrecarga de tipos

*Ejemplo*: en este archivo tenemos ambas funciones con las firmas que indican que la entrada y la salida deben ser enteros, la solución es <u>usar sobrecarga de tipos</u>

```
cuadrado1.hs ×
{-
Calcular el cuadrado de un número
calcular el valor de un número elevado a la 4 utilizando la función del inciso anterior
-}
cuadrado :: Int -> Int -- una entrada de tipo Double con una salida de tipo Double
cuadrado x = x^2
elevadoCuatro :: Int -> Int -- reutilizamos cuadrado
elevadoCuatro x = cuadrado(cuadrado x)
```

### Ejemplo 2 - sobrecarga de tipos

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
GHCi, version 8.8.4: https://www.haskell.org/ghc/ :? for help
Prelude> :cd haskell
Prelude> :l cuadrado.hs
[1 of 1] Compiling Main
                                   ( cuadrado.hs, interpreted )
Ok, one module loaded.
*Main> cuadrado (-10.5)
110.25
*Main> elevadoCuatro (-10.5)
12155.0625
*Main> :l cuadrado1.hs
[1 of 1] Compiling Main
                                   ( cuadradol.hs, interpreted
                                                                       porque en el otro
Ok, one module loaded.
                                                                       archivo tenemos
*Main> cuadrado (-10.5)
                                                                       definida la función con
<interactive>:6:12: error:

    No instance for (Fractional Int) arising from the literal '10.5'

                                                                       enteros solamente
   • In the expression: 10.5
     In the first argument of 'cuadrado', namely '(- 10.5)'
     In the expression: cuadrado (- 10.5)
```

## Ejemplo 3 - sobrecarga de tipos

*Ejemplo*: en este archivo la sobrecarga de datos nos permite olvidarnos del tipo de datos que ingrese el usuario, mientras sea un número

```
cuadrado2.hs ×
{-
Calcular el cuadrado de un número
calcular el valor de un número elevado a la 4 utilizando la función del inciso anterior
-}
cuadrado :: (Num a)=> a -> a --(un número de un tipo de datos a) =>
cuadrado x = x^2 -- con una entrada de tipo a y una salida de tipo a
elevadoCuatro :: (Num a) => a -> a
elevadoCuatro x = cuadrado(cuadrado x)
```

## Ejemplo 3 - sobrecarga de tipos

```
Terminal
Archivo Editar Ver Buscar Terminal Ayuda
GHCi, version 8.8.4: https://www.haskell.org/ghc/ :? for help
Prelude> :cd haskell
Prelude> :l cuadrado2.sh
target 'cuadrado2.sh' is not a module name or a source file
Prelude> :l cuadrado2.hs
[1 of 1] Compiling Main ( cuadrado2.hs, interpreted )
Ok, one module loaded.
*Main> cuadrado 5
*Main> cuadrado 3.5
12.25
*Main> cuadrado (-5)
*Main> elevadoCuatro 5
625
*Main> elevadoCuatro 3.5
150.0625
*Main> elevadoCuatro (-5)
625
*Main>
```

### Clases de Tipos

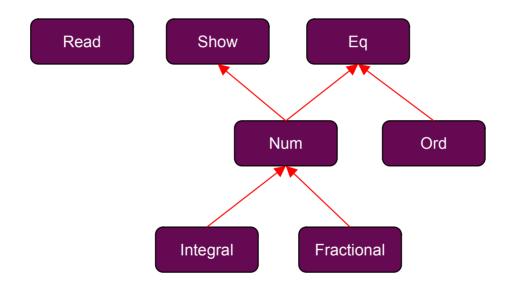
Entonces al usar la <u>clase de tipo</u> Num, podemos especificar que el tipo de la función cuadrado es **a** y que <u>a es miembro del conjunto de tipos Num</u>, o bien, que a pertenece al conjunto Num ( $a \in Num$ ).

Haskell abrevia a ∈ Num de la forma (Num a) y lo coloca antes de una doble flecha =>. Por lo tanto, la entrada y la salida son de tipo a:

cuadrado :: (**Num** a) => a -> a

cuadrado  $x = x^2$ 

# Clases de Tipos



#### Clases de tipos

```
*Main> elevadoCuatro (-5)
625
*Main> :t (+)
(+) :: Num a => a -> a -> a
*Main> :t (<)
(<) :: Ord a => a -> a -> Bool
*Main> :t (==)
(==) :: Eq a => a -> a -> Bool
*Main> :t (/=)
(/=) :: Eq a => a -> a -> Bool
*Main> :t (<=)
(<=) :: Ord a => a -> a -> Bool
*Main> :t (/)
(/) :: Fractional a => a -> a -> a
*Main> :t (mod)
(mod) :: Integral a => a -> a -> a
*Main> mod 5 2
*Main> mod 4 2
```

#### Clases de tipos

```
*Main> elevadoCuatro (-5)
625
*Main> :t (+)
(+) :: Num a => a -> a -> a
*Main> :t (<)
(<) :: Ord a => a -> a -> Bool
*Main> :t (==)
(==) :: Eq a => a -> a -> Bool
*Main> :t (/=)
(/=) :: Eq a => a -> a -> Bool
*Main> :t (<=)
(<=) :: Ord a => a -> a -> Bool
*Main> :t (/)
(/) :: Fractional a => a -> a -> a
*Main> :t (mod)
(mod) :: Integral a => a -> a -> a
*Main> mod 5 2
*Main> mod 4 2
```

Fuente: teoría de la cátredra (70/77, 86/88)