# Generics

Los ***Generics*** fueron introducidos en la versión 5 de Java (2004) junto con otras muchas novedades suponiendo en su historia una de las mayores modificaciones o al mismo nivel de las novedades introducidas con Java 8 más recientemente al lenguaje Java. Los ***Generics*** son importantes ya que permiten al compilador informar de muchos errores de compilación que hasta el momento solo se descubrirían en tiempo de ejecución, al mismo tiempo permiten eliminar los ***cast*** *(ClassCastException)* simplificando, reduciendo la repetición y aumentando la legibilidad el código.

Los ***Generics*** permiten usar tipos para parametrizar las clases, interfaces y métodos al definirlas. Los beneficios son:

- Comprobación de tipos más fuerte en tiempo de compilación.
- Eliminación de *casts* aumentando la legibilidad del código.
- Posibilidad de implementar algoritmos genéricos, con tipado seguro.

Sobre este último beneficio:
Muchos algoritmos son lógicamente los mismos, independientemente del tipo de datos a los que se apliquen. Por ejemplo, un Quicksort (algoritmo de ordenación) es el mismo si está ordenando elementos de tipo Integer, String, Object, etc. Con los genéricos, puede definir un algoritmo una vez, independientemente de cualquier tipo específico de datos, y luego aplicar ese algoritmo a una amplia variedad de tipos de datos sin ningún esfuerzo adicional.

Ejemplos:

proyecto: **ejemplos**

- class: *GenericsEjemplo1*
- class: *GenericsEjemplo2*
- class: *GenericsEjemplo3*
- class: *GenericsEjemplo4* (Tipos limitados)
- class: *GenericsEjemplo5* (Quicksort)

Sobre Quicksort, lejos de explicar la lógica del algoritmo, a esta altura la buscan en la web y hay explicaciones interactivas maravillosas, nos centramos en como reutilizar el algoritmo con *Generics.*

Bien veamos primero el algoritmo para números (Integer):

```
private static int particion(int arreglo[], int izquierda, int derecha) {
    int pivote = arreglo[izquierda];
    // Ciclo infinito
    while (true) {
        while (arreglo[izquierda] < pivote) {
            izquierda++;
        }
        while (arreglo[derecha] > pivote) {
            derecha--;
        }
        if (izquierda >= derecha) {
            return derecha;
        } else {
            int temporal = arreglo[izquierda];
            arreglo[izquierda] = arreglo[derecha];
            arreglo[derecha] = temporal;
            izquierda++;
            derecha--;
        }
        // El while se repite hasta que izquierda >= derecha
    }
}

private static void quicksort(int arreglo[], int izquierda, int derecha) {
    if (izquierda < derecha) {
        int indiceParticion = particion(arreglo, izquierda, derecha);
        quicksort(arreglo, izquierda, indiceParticion);
        quicksort(arreglo, indiceParticion + 1, derecha);
    }
}
```

y la llamada sería:

```
public static void main(String[] args) {
    int numeros[] = {1, 9, 23, 4, 55, 100, 1, 1, 23};
    System.out.println("Antes de QS: " + Arrays.toString(numeros));
    quicksort(numeros, 0, numeros.length - 1);
    System.out.println("Después de QS: " + Arrays.toString(numeros));
}
```

Ahora bien, vamos a realizar el algoritmo para cadenas (String):

```
private static int particion(String arreglo[], int izquierda, int derecha) {
    // Elegimos el pivote, es el primero
    String pivote = arreglo[izquierda];
    // Ciclo infinito
    while (true) {
        while (arreglo[izquierda].compareTo(pivote) < 0) {
            izquierda++;
        }
        while (arreglo[derecha].compareTo(pivote) > 0) {
            derecha--;
        }
        if (izquierda >= derecha) {
            return derecha;
        } else {
            String temporal = arreglo[izquierda];
            arreglo[izquierda] = arreglo[derecha];
            arreglo[derecha] = temporal;
            izquierda++;
            derecha--;
        }
    }
}

private static void quicksort(String arreglo[], int izquierda, int derecha) {
    if (izquierda < derecha) {
        int indiceParticion = particion(arreglo, izquierda, derecha);
        quicksort(arreglo, izquierda, indiceParticion);
        quicksort(arreglo, indiceParticion + 1, derecha);
    }
}

public static void main(String[] args) {
    String[] nombres = {"Leon", "Chris", "Jill", "Wesker", "Ada"};
    System.out.println("Antes de QS: " + Arrays.toString(nombres));
    quicksort(nombres, 0, nombres.length - 1);
    System.out.println("Después de QS: " + Arrays.toString(nombres));
}
```

¿¿¿Otra vez lo mismo pero para otro tipo de datos???

Ohhhh!!! ¿y ahora quién podrá ayudarnos?

**Siiii *Generics!!!***

En clase vemos el ejemplo, que está en el repositorio Bitbucket como se detallo al inicio.