Generics en Java

¿Qué son los Generics en Java?

Los generics permiten usar tipos para parametrizar las clases, interfaces y métodos al definirlas.

Utilizar un tipado generic es utilizar los tipos como “variables”.

Según las convenciones los nombres de los parámetros de tipo usados comúnmente son los siguientes:

E: elemento de una colección.

K: clave.

N: número.

T: tipo.

V: valor.

S, U, V etc: para segundos, terceros y cuartos tipos.

Métodos Generic

Los generics permiten adaptar un método con tipos específicos a un método que sirva para cualquier tipo con el que estés trabajando.

Por ejemplo, supongamos que tenemos un método que suma dos números. Si lo hacemos con tipos específicos es posible que tengamos que crear varias versiones de este método:

```
public Integer Add(Integer a, Integer b)

public Double Add(Double a, Double b)

public Float Add(Float a, Float b)
```

Los generics permiten crear un único método personalizado para cualquier tipo que lo invoca:

```
public <T> T Add(T a, T b)
```

T se sustituye por el tipo que se utilice

Reglas de los métodos Generic

1 - Todas las declaraciones de métodos generics tienen una sección de parámetros de tipo delimitada por paréntesis angulares <T> que precede al tipo de retorno del método:

```
public <T> void miMetodo(T t){…}
```

2 - Cada sección de parámetros de tipo contiene uno o más parámetros de tipo separados por comas. Un parámetro de tipo, también conocido como variable de tipo, es un identificador que especifica un nombre de tipo genérico.

```
public <T,V> void miMetodo(T t, V v){…}
```

3 - Los parámetros de tipo pueden utilizarse para declarar el tipo de retorno y para los tipos de los argumentos pasados al método genérico.

```
public <T,V> T miMetodo(T t, V v){…}
```

4 - El cuerpo de un método genérico se declara como el de cualquier otro método. Hay que tener en cuenta que los parámetros de tipo sólo pueden representar tipos de referencia, no tipos primitivos (como int, double y char).

5 - Podemos crear un método Generic dentro de una clase no Generic.

Ejemplo de método Generic

Vamos a ver como podemos hacer un método genérico que nos permita imprimir Arrays de distintos tipos:

```
Integer[] intArray = { 1, 2, 3, 4, 5 };
Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };
Character[] charArray = { 'H', 'E', 'L', 'L', 'O' };

printArray(intArray);
printArray(doubleArray);
printArray(charArray);
```

```
public static <E> void printArray( E[] elArray ) {
        for(E element : elArray) {
        System.out.printf(element);
}
```

Método Generic

Métodos Generic con parámetros de tipos limitados

Puede haber ocasiones en las que deseemos restringir los tipos que se pueden pasar a un parámetro Generic.

Por ejemplo, un método que opera con números puede querer aceptar sólo instancias de Number o sus subclases. Para eso están los parámetros de tipo limitados.

Para declarar un parámetro de tipo acotado debemos indicar el nombre del parámetro de tipo, seguido de la palabra clave extends, seguida de su límite superior:

```
public static <T extends Comparable<T>> T elMetodo(T x, T y, T z) { … código muy complejo ;)}
```

Este método solo admite, como T, clases que extiendan de la interface Comparable<T>

Clases Generic

Una declaración de clase generic se parece a una declaración de clase no generic, salvo que el nombre de la clase va seguido de una sección de parámetros de tipo.

Al igual que con los métodos generic, la sección de parámetros de tipo de una clase generic puede tener uno o más parámetros de tipo separados por comas.

Estas clases se conocen como clases parametrizadas o tipos parametrizados porque aceptan uno o más parámetros.

```
public class Box<T> {
	private T t;

	public <T> void add(T t) {
		this.t = t;
	}

	public <T> T get() {
		return t;
	}
}
```

Interfaces Generic

Una declaración de interface generic se parece a una declaración de interface no generic, salvo que el nombre de la interface va seguido de una sección de parámetros de tipo.

Al igual que con los métodos generic, la sección de parámetros de tipo de una interface generic puede tener uno o más parámetros de tipo separados por comas.

Cuando una interface es implementada por una clase debemos definir el tipo que emplearemos en la clase:

miClase implements IMovible<String>

```
public interface IMovible<T> {

    void mover(T t, String locationCode);
    T getItemQueSeMueve();
    String getLugarAlQueSeMueve();
}
```

En general, si una clase implementa una interfaz genérica, entonces esa clase también debe ser genérica. Si una clase implementa un tipo específico de interfaz genérica entonces la clase implementadora no necesita ser genérica.

Restricciones de Generic

1 - No se pueden instanciar clases genéricas con tipos primitivos. miClaseGenerica<int,char>…

2 - No se pueden crear instancias de los parámetros de tipo. T t = new T()

3 - No se pueden declarar campos static cuyos tipos son parámetros de tipo. private static T t;

4 - No se pueden usar casts o instanceof con tipos parametrizados.

5 - No se pueden crear arrays de tipos parametrizados.

6 - No se pueden crear, capturar o lanzar tipos parametrizados que extiendan de Throwable.