

# GENERIC

UNIDAD 6



# ¿Qué son los “generics” en JAVA?

**Los tipos genéricos (Generics) permiten establecer restricciones a nivel de tipo, haciendo que ciertas clases, interfaces o métodos acepten únicamente los tipos estipulados.**

**Su uso está ligado, mayoritariamente, a las colecciones (Collections), donde ayudan a realizar comprobación de tipos en tiempo de compilación.**

# Uso de Generics

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.

```
public class Box<T> {  
    private T t;  
  
    public T get() {  
        return t;  
    }  
  
    public void set(T t) {  
        this.t = t;  
    }  
}
```


# EJEMPLOS

```
public class Box<T> {  
    private T t;  
  
    public T get() {  
        return t;  
    }  
  
    public void set(T t) {  
        this.t = t;  
    }  
}
```

*la variable t está  
representada por el nuevo  
tipo de dato genérico*



*<T> es nuestro tipo de dato  
genérico. Se utiliza la  
nomenclatura de  
"diamantes" <>*



# EJEMPLOS

```
public class Test {
```

```
public static void main(String[] args) {
```

```
Box<String> unDato = new Box<>();
```

*creamos una variable  
usando la clase  
genérica **Box***

```
unDato.setDato("PRUEBA");
```

*nuestro dato es un String; accesible  
a los métodos de la clase*

```
System.out.println("Nuestro dato es: " + unDato.getDato());
```

```
}
```

```
}
```

# ArrayList y Generics

En este caso, el procedimiento es similar. La referencia a la clase Generics es lo que cambia. En este caso, utilizamos la letra "E"

```
import java.util.ArrayList;

public class Lista<E> {

    private ArrayList<E> elementos;

    public Lista() {
        elementos = new
ArrayList<>();
    }
}
```