


CREACIÓN DE UNA CLASE GENÉRICA



CLASE GENÉRICA

- ▶ Se trata de una clase parametrizada sobre uno o más tipos.

```
public class Box {  
    private Object object;  
  
    public void set(Object object) {  
        this.object = object;  
    }  
  
    public Object get() {  
        return object;  
    }  
}
```



Este tipo de código es propenso a producir errores

CLASE GENÉRICA A PARTIR DE JAVA SE 5

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

Se asigna el
tipo en
tiempo de
compilación

CLASE GENÉRICA A PARTIR DE JAVA SE 5

```
public class Par<T, S> {  
    private T obj1;  
    private S obj2;  
  
    //Resto de la clase  
  
}
```

CONVENCIÓN SOBRE EL NOMBRE DE PARÁMETROS

Los nombres de tipos de parámetros más usados son:

- ▶ **E** (*element*, elemento)
- ▶ **K** (*key*, clave)
- ▶ **N** (*number*, número)
- ▶ **T** (*type*, tipo)
- ▶ **V** (*value*, valor)
- ▶ **S, U, V, ...** (2º, 3º, 4º, ... tipo)

INSTANCIACIÓN Y OPERADOR DIAMOND

- ▶ Para instanciar un objeto genérico, tenemos que indicar los tipos *dos veces*.

```
Par<String, String> pareja2 =  
    new Par<String, String>("Hola", "Mundo");
```

- ▶ Este estilo es muy *verboso*.
- ▶ Desde Java SE 7 tenemos el operador <> (*diamond*).

```
Par<String, String> pareja2 =  
    new Par<>("Hola", "Mundo");
```

GENÉRICOS CON TIPOS CERRADOS

- Podemos indicar que el tipo parametrizado sea uno en particular (o sus derivados).

```
public class NumericBox<T extends Number> {  
    private T object;  
    //resto de la clase  
}
```

GENÉRICOS CON TIPOS CERRADOS

- ▶ Podemos indicar más de un tipo
- ▶ Solo uno de ellos puede ser una clase.
- ▶ El resto deben ser interfaces
- ▶ La clase a extender debe ser la primera de la lista.

```
public class A {  
    //resto de la clase  
}
```

```
public interface B {  
    //resto de la interfaz  
}
```

```
public class StrangeBox  
    <T extends A & B> {  
  
    //resto de la clase
```

```
}
```


GENÉRICOS CON TIPOS COMODÍN

- ▶ Nos permiten *relajar* el tipo concreto de una clase genérica a un subtipo.
- ▶ Útil con colecciones.

```
public static double sumOfList(List<? extends Number> list)
{
    double s = 0.0;
    for (Number n : list)
        s += n.doubleValue();
    return s;
}
```