Java Clases Genéricas

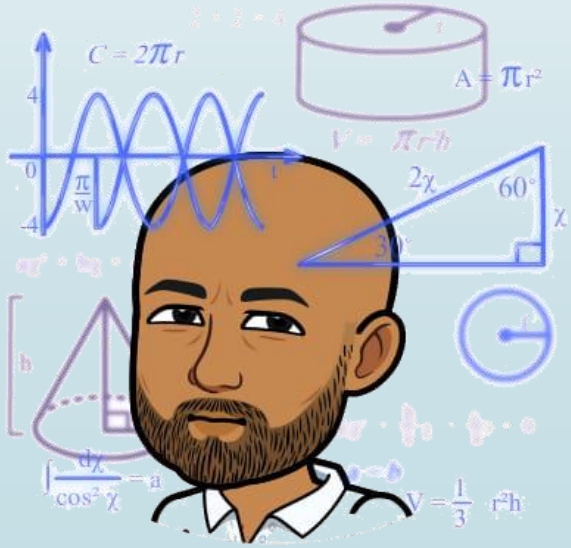
Octavio Robleto

octavio.robleto@gmail.com

https://octaviorobleto.com

Genérico o Tipo Parametrizado

- Cuando creamos una nueva clase, debemos conocer el tipo de dato u objeto con el vamos a trabajar. Hay momentos que necesitamos hacer que dicha clase maneje varios tipos de datos. Cuando necesitemos hacer esto recurrimos a las genéricas.
- También llamados Tipo parametrizados nos permiten crear clases, interfaces y métodos en los que el tipo de datos sobre los que queremos operar se envían como argumentos al instanciar la clase, de esta forma la clase trabajara el dato que le enviamos.



Prof. Octavio Robleto

Como se hacen?

- Al igual que al crear una clase como lo hemos realizado anteriormente colocamos lo siguiente:

1. Modificador de acceso.
2. Palabra reservada class.
3. Nombre de la clase.
4. Apertura llave Angular (Símbolo Menor Qué).
5. Letras genéricas en mayúsculas (si colocamos mas de una separar con comas).
6. Cierre llave Angular (Símbolo Mayor Qué).
7. Si lo necesitamos las herencias e implementaciones de clases e interfaces.
8. Apertura y Cierre de llaves.

Parámetro de Tipo

```
public class Generica<T> {
```

Reglas por convención

Letra	Valor	Descripción
E	Element	(usado bastante por Java Collections Framework)
K	Key	(Clave, usado en mapas)
N	Number	(para números)
T	Type	(Representa un tipo, es decir, un objeto)
V	Value	(representa el valor, también se usa en mapas)
S,U,V etc.	usado para representar otros tipos	

Métodos Genéricos

- Usados dentro de la clase con las letras declaradas.

Métodos

```java
public T getElemetoGenerico() {
    return elemetoGenerico;
}


public void setElemetoGenerico(T elemetoGenerico) {
    this.elemetoGenerico = elemetoGenerico;
}
```

```java
public Generica() {
    // TODO Auto-generated constructor stub
}

public Generica(T elemetoGenerico) {
    this.elemetoGenerico = elemetoGenerico;

}
```

Constructores

Como se instancian?

1. Nombre de la clase a instanciar.
2. Apertura llave Angular (Símbolo Menor Qué).
3. Tipos de Objeto (si son mas de uno separar con comas).
4. Cierre llave Angular (Símbolo Mayor Qué)
5. Identificador del objeto.
6. Instanciación del objeto
7. Nombre de la clase .
8. Apertura y Cierre de llave Angular.
9. Paréntesis del constructor de clase con los argumentos necesarios.

```
Generica<String> miGenerico1 = new Generica<>("Soy un String");
Generica<Integer> miGenerico2 = new Generica<>(2);
Generica<Object> miGenerico3 = new Generica<>(new Object());
```

Importante

- No se aceptan datos de tipo primitivo.

```
Generica<int> miGenerico2 = new Generica<>(2);
```

- Al no aceptar tipo de datos primitivos podemos hacer el uso de los tipo envoltorios o wrappers.
- No se le puede asignar una versión distinta de otro genérico pero de diferente tipo.

```
Generica<String> miGenerico1 = new Generica<>("Soy un String");
Generica<Integer> miGenerico2 = new Generica<>(2);
Generica<String> miGenerico3 = new Generica<>("Soy otro String");

miGenerico1 = miGenerico2;

miGenerico3 = miGenerico1;
```

Error

Aceptado

Prof. Octavio Robleto

Tomar en cuenta

- Nuestro IDE nos puede dar una sugerencia a la hora de instanciar el objeto genérico.

```
Generica<String> miGenerico1 = new Generica<String>("Soy un String");
```

- Se le denomina operador diamante por la forma que tiene el operador “<>” y permite simplificar el manejo de los genéricos asumiendo el datos definido en la declaración del objeto.

```
Generica<String> miGenerico1 = new Generica<>("Soy un String");
```

- Es importante ya que vamos a ver muchos ejemplos en la nube y debemos comenzar a usar la simplificación para una mejor lectura.

Ventajas

- Reutilización de código; podemos escribir una Método / clase / interfaz una vez y usarlo para cualquier tipo que queramos, esto hace que el código sea mas sencillo **“Mas limpio”.**
- Errores en **tiempo de compilación**, validando el tipo de dato que se le envía a la clase genérica con la enviada como argumento que recibe el constructor.
- El uso excesivo de los casting al momento de **No** usar genéricos.