**MÉTODOS GENÉRICOS**

**MOTIVACIÓN PARA LOS MÉTODOS GENÉRICOS**

* Los métodos sobrecargados son utilizados para realizar operaciones similares con diferentes tipos de datos. A continuación, un ejemplo:

|  |
| --- |
| **package** ejemplo\_01\_imprimir;  **public** **class** Imprimir {    **static** **void** imprimirArreglo(Integer[] arregloEnteros) {  **for** (Integer entero : arregloEnteros) {  System.***out***.printf("%s", entero);  }  System.***out***.println("\n");  }    **static** **void** imprimirArreglo(Double[] arregloDoubles) {  **for** (Double decimal : arregloDoubles) {  System.***out***.printf("%s", decimal);  }  System.***out***.println("\n");  }    **static** **void** imprimirArreglo(Character[] arregloCaracteres) {  **for** (Character letra : arregloCaracteres) {  System.***out***.printf("%s", letra);  }  System.***out***.println("\n");  }    **public** **static** **void** main(String[] args) {    Integer[] arregloEnteros = {1, 2, 3, 4, 5};  Double[] arregloDoubles = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};  Character[] arregloCaracteres = {'H', 'O', 'L', 'A'};    *imprimirArreglo*(arregloEnteros);  *imprimirArreglo*(arregloDoubles);  *imprimirArreglo*(arregloCaracteres);      }  } |

* El código anterior contiene tres métodos con el mismo nombre, pero imprimen diferentes tipos de arreglos. Un método es para imprimir Integers, otro para imprimir Doubles y el último para imprimir Caracteres.
* Pudimos haber utilizado arreglos de tipos primitivos (int, double, char).
* En lugar de esos, utilizamos arreglos de tipo Objeto (Wrapper Classes) para realizar nuestros ejemplos de métodos genéricos.
* Esto es porque solamente los tipos objetos pueden ser utilizados para especificar tipos genéricos en métodos y clases.
* Los tres métodos realizan la misma tarea, imprimir el contenido de un arreglo. Se puede remplazar el tipo de datos (Integer[], Double[], Character[]) con un nombre Genérico. Se utiliza la letra T por convención para un tipo de dato.
* El código quedaría de la siguiente manera.

|  |
| --- |
| Antes: |
| **static** **void** imprimirArreglo(Integer[] arregloEnteros) {  **for** (Integer entero : arregloEnteros) {  System.***out***.printf("%s", entero);  }  System.***out***.println("\n");  }    **static** **void** imprimirArreglo(Double[] arregloDoubles) {  **for** (Double decimal : arregloDoubles) {  System.***out***.printf("%s", decimal);  }  System.***out***.println("\n");  }    **static** **void** imprimirArreglo(Character[] arregloCaracteres) {  **for** (Character letra : arregloCaracteres) {  System.***out***.printf("%s", letra);  }  System.***out***.println("\n");  } |
| Después |
| /\* Metodo Generico \*/  **static** <T> **void** imprimirArreglo(T[] arreglo) {    **for** (T elemento : arreglo) {  System.***out***.printf("%s", elemento);  }    System.***out***.println("\n");  } |

* Lo que hicimos fue remplazar los tres métodos con un método genérico.
* Esto se logra cambiando el tipo de dato, ya no será Integer[], Double y Character[], ahora será T[].
* Ahora solo tenemos un método que imprime los elementos de un arreglo
* (“%s” muestra la representación en forma de String de los elementos del arreglo). Esto se hace para que acepte cualquier tipo de dato. Ya que para double se utiliza %f, y así con los demás.
* Si las operaciones realizadas por varios métodos sobrecargados son idénticas (utilizando argumentos diferentes pero similares), es más conveniente codificarlos utilizando un método genérico.
* Podemos escribir un solo método genérico que puede ser llamado con argumentos de diferentes tipos.
* Basando en los tipos de argumentos pasados al método genérico, el compilador maneja cada llamada al método apropiadamente.
* En **tiempo de compilación** el compilador asegura la **seguridad de tipos** (**type safety**) del código previniendo errores en tiempo de ejecución.
* A continuación, se muestra el código completo.

|  |
| --- |
| **package** ejemplo\_01\_imprimir;  **public** **class** ImprimirA {    /\* Metodo Generico \*/  **static** <T> **void** imprimirArreglo(T[] arreglo) {    **for** (T elemento : arreglo) {  System.***out***.printf("%s", elemento);  }    System.***out***.println("\n");  }    **public** **static** **void** main(String[] args) {    Integer[] arregloEnteros = {1, 2, 3, 4, 5};  Double[] arregloDoubles = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7};  Character[] arregloCaracteres = {'H', 'O', 'L', 'A'};    *imprimirArreglo*(arregloEnteros);  *imprimirArreglo*(arregloDoubles);  *imprimirArreglo*(arregloCaracteres);  }    } |

**Type parameter Section of a Generic Method**

* Todas las declaraciones de métodos genéricos tienen la sección **parámetro de tipo** (**type-parameter** En este ejemplo sería **<T>**) delimitado por paréntesis angulares que están antes del tipo de retorno del método.
* La sección **parámetro de tipo** puede contener uno o más parámetros de tipo, separados por comas.
* Un **parámetro de tipo**, también conocido como tipo de variable, es un identificador que especifica un nombre de tipo genérico.
* Los **parámetros de tipo** pueden ser utilizados para declarar el retorno, parámetros y variables locales en la declaración de un método genérico, y actúan como marcadores de posición para los tipos de los argumentos que son pasados actualmente al método genérico (**actual type arguments)**.
* Los **parámetros de tipo** solo pueden ser representados por tipos objeto (**reference type, wrapper clases).** No se pueden utilizar tipos primitivos (int, double, char)
* Las letras que son más comúnmente utilizadas para los tipos de parámetros (type parameters) son:
  + T para tipo
  + E para elemento
  + K para clave(key)
  + V para valor