

**FACULTAD DE INGENIERIA**

**ESCUELA ACADÉMICO PROFESIONAL**

**DE INGENIERIA DE SISTEMAS**

**EXPERIENCIA CURRICULAR DE**

**Metodología de programacion**

TÍTULO

**Java generics**

**AUTORES**

**1° CRISTHAIN VARGAS**

**2° OMAR PUERTAS**

**3° GIANCARLO PACULIA**

**4° MELECIO MOLLEAPAZA**

**5° JHUNIOR MEDINA**

**ASESOR**

Erick Gustavo Coronel Castillo

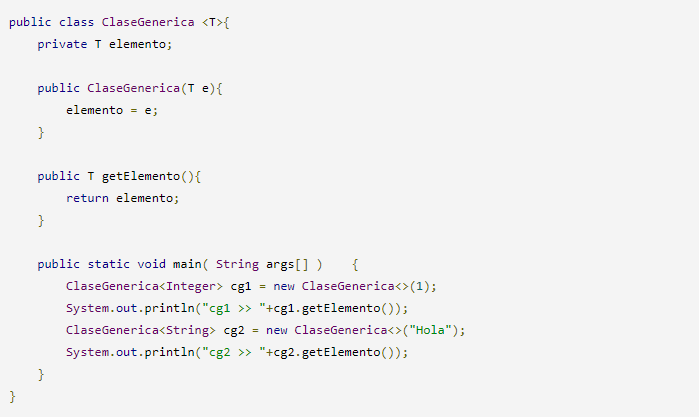
**LIMA – PERÚ**

**2017**

CLASES EN JAVA GENERIC

En Java, cuando definimos una nueva clase, se debe conocer el tipo de dato con el que trabajaremos. Si se desea hacer una operación en específico dentro de la clase, sea cual sea el tipo de dato que va a recibir, podemos hacer uso de los tipos genéricos. Este tipo genérico asumirá el tipo de dato que realmente le pasaremos a la clase

Ejemplo:



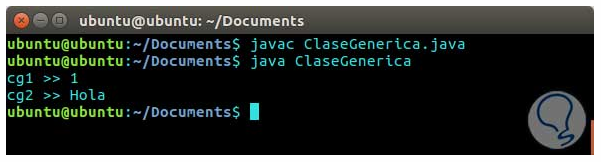
Veamos lo primero la declaración de la clase, puedes apreciar que después del nombre de la clase tenemos <T>, nos indica que es una clase genérica, T podrá ser un String, un Integer, etc. Se sustituirá por un tipo real cuando llegue el momento.

Tenemos una variable llamada elemento, que es del tipo T (es decir genérico). En el constructor rellenaremos está variable con el dato que nos pasen, también del tipo T lógicamente.

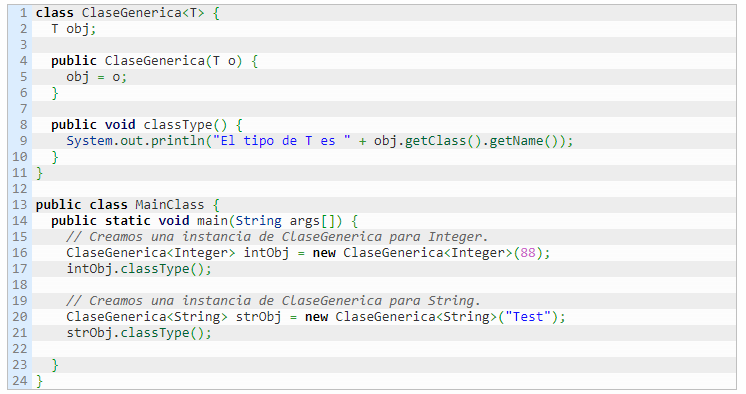
Antes de llegar al main creamos una función para obtener el elemento, por eso retorna un tipo T, no haría falta está clase, tal y como está el código podríamos llamar al elemento directamente, pero para que veas el tipo de devolución de la función no está de más.

Dentro del main lo único que haremos es crear 2 objetos de esa clase, el primero de tipo Integer y el segundo de tipo String y mostramos el elemento para ver que funciona correctamente.

Si compilamos y ejecutamos el código se verá así:



Otro ejemplo:



* T es el tipo genérico que será reemplazado por un tipo real.
* T es el nombre que damos al parámetro genérico.
* Este nombre se sustituirá por el tipo real que se le pasará a la clase.

El resultado será el siguiente:



* En java los genéricos solo funcionan en objetos.

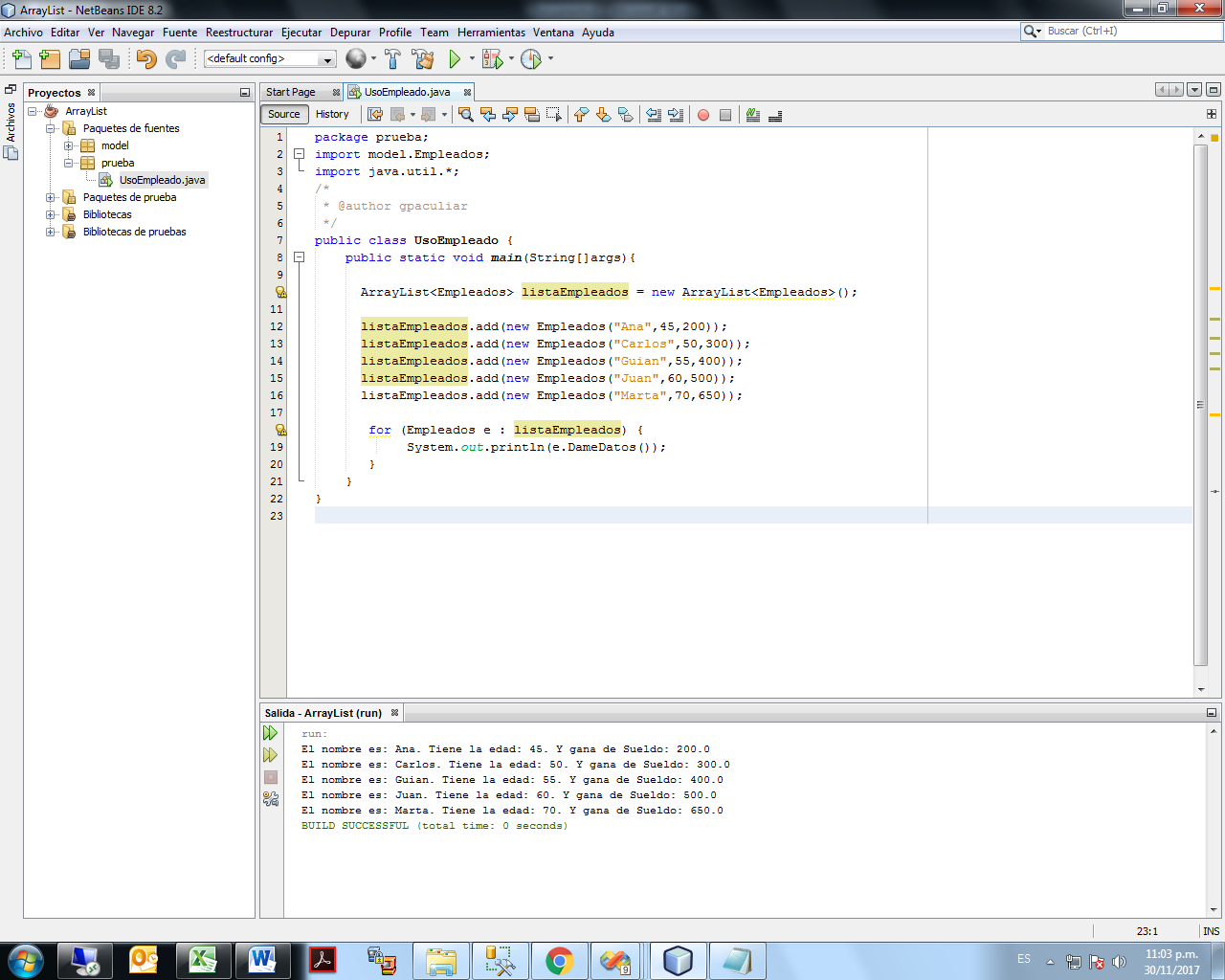


Existen una serie de **convenciones para nombrar a los genéricos**:

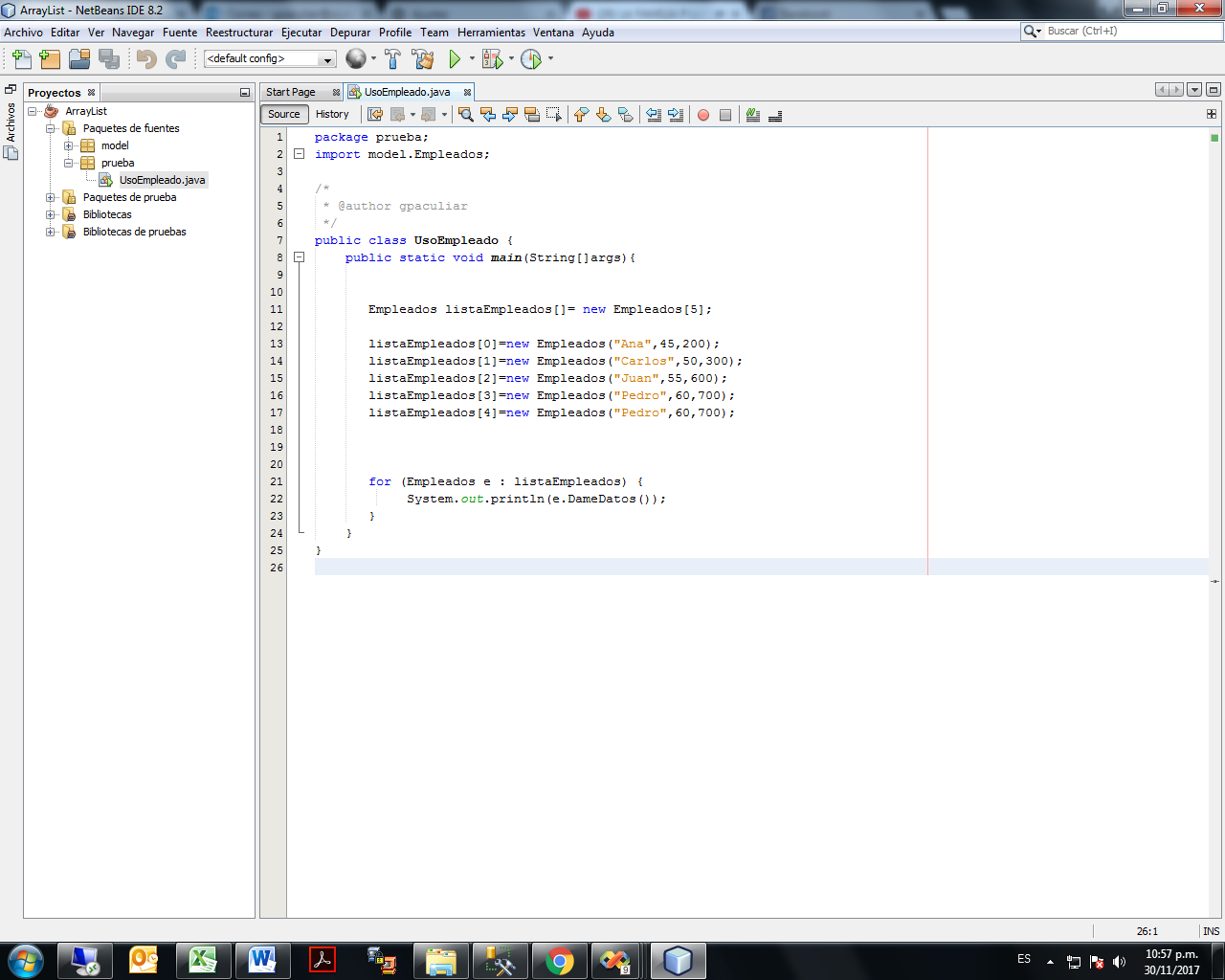
* E – Element (usado bastante por Java Collections Framework)
* K – Key (Llave, usado en mapas)
* N – Number (para números)
* T – Type (Representa un tipo, es decir, una clase)
* V – Value (representa el valor, también se usa en mapas)
* S, U, V etc. – usado para representar otros tipos.

ARRAYLIST

La clase ArrayList en Java, es una clase que permite almacenar datos en memoria de forma similar a los Arrays, con la ventaja de que el numero de elementos que almacena, lo hace de forma dinámica, es decir, que no es necesario declarar su tamaño como pasa con los Arrays.

**Forma normal:**

**Con el método**

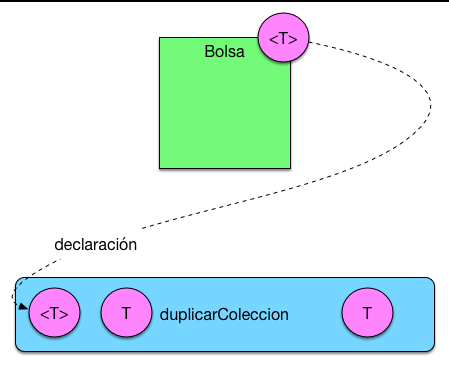
**Generic**:

Java Generic Methods

Sin embargo cuando nos referimos a métodos genéricos estamos hablando de métodos que son genéricos sin pertenecer a una clase genérica. Esto puede parecer raro en un primer momento , lo veremos más claro en código:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | package com.arquitecturajava;    import java.util.ArrayList;  import java.util.Arrays;  import java.util.Collection;    public class PrincipalGenericMethod {  public static void main (String args[]) {    Collection<String> lista= new ArrayList<String>(Arrays.asList("A","B","C"));    Collection<String> nueva= duplicarColleccion(lista);    nueva.forEach(x->System.out.println(x));    }    public static <T> Collection<T> duplicarColleccion(Collection<T> coleccion) {    coleccion.addAll(coleccion);  return coleccion;    }  } |

En este caso e**l método duplicarColleccion** se encarga de volver a copiar los mismos elementos en la colección **para tenerlos duplicados**. Es un método genérico ya que admite cualquier tipo de colección cambiando el tipo T. Pero ademas no pertenece a ninguna clase en concreto ya que el tipo T se declara de forma**explicita en el método**. Para entenderlo de forma más sencilla es como si la declaración <T >de la clase parara al propio método.

[](https://www.arquitecturajava.com/wp-content/uploads/JavaGenericMethods.png)

Es un código que cuesta entender ya que parece que tiene dos parámetros de retorno genéricos y no solo uno. En primer lugar <T> y en segundo lugar Collection<T>. Los Java Generic Methods aportan una mayor flexibilidad a nuestra programación

BIBLIOGRAFIA

Clases y tipos genéricos en Java [página web]. España: Bilbao, (23 de octubre del 2012). [fecha de consulta: 20 de noviembre de 2017].

Recuperado de: <http://jonsegador.com/2012/10/clases-y-tipos-genericos-en-java/>