Genéricos

Por: Sebastián Sánchez

Imaginate que hacés la mejor estructura del universo, pero necesitas una para Strings, otra para Documentos, otra para Libros. Tendrías que hacer una lista enlazada para cada tipo de objeto; esto implicaría copiar y pegar el texto de tu clase *n* veces, rompiendo de esa manera el fin de la optimización de código, o sea, hacer todo mal.

Este día te traigo la solución, los genéricos. En diversos lenguajes de programación podemos establecer clases y metodos genéricos, estos aceptan el tipo de objeto que sea (hay excepciones, no todo es color de rosa). Esto tiene sus desventajas, debemos sobrecargar algunos operadores o metodos en ciertos casos, como una busqueda o una comparación, pero nos ayuda a ahorrarnos centenares de lineas de código.

Genéricos en C++

Para indicar que una clase es una clase genérica debemos agregarle un **template** antes de declararla, de esta manera:

```
template <class T>
class Nodo
{
     T *elemento;
     ...
}

template <class T>
class Lista
{
     Nodo *cabeza;
     ...
}
```

T puede ser cualquier tipo de dato que podemos almacenar, así podemos definirlo después al inicializar el objeto. Además, todos sus metodos deben tener un **template** de la misma manera.

```
template <class T>
void List<T>::insertar(T* object)
{
     ...
}
```

Al incluir este archivo en algún otro, tuve varios problemas, es importante incluir el archivo .cpp (donde están todas las definiciones) y no el .h (el header, donde se declaran atributos y métodos):

```
#include "MiClase.cpp"
```

Además, al archivo .cpp le agregamos la siguiente sentencia en la cabecera:

```
#pragma once
#include "MiClase.h"
```

Esto le indica al compilador que el archivo debe incluirse una vez en el proyecto. Luego podemos implementar nuestra clase de esta manera, para el ejemplo, una lista:

```
Lista<Documento> *lista = new Lista<Documento>();
Lista<String> *lista = new Lista<String>();
Lista<Libro> *lista = new Lista<Libro>();
Lista<Cosa> *lista = new Lista<Cosa>();
```

Genéricos en Java

El proceso en Java es más sencillo, basta con agregar un pequeño elemento en la creación de nuestra clase:

```
public class MiClase <T>
{
     T elemento;
     ...
}
```

La incializamos de la misma manera, pero los “diamantes” pueden estar vacíos en la instancia:

```
MiClase<Documento> cosa = new MiClase<>();
```

En Java el tipo genérico no puede recibir como parámetro un tipo de dato primitivo, se lanzará una excepción.

Genéricos en C#

Los genéricos en C# se declaran de la misma forma que en Java, podemos definir *n* cantidad de tipos genéricos. Además, la inicialización es exactamente igual que en C++, estos no dejan el “diamante” vacío.

Como ejemplo:

```
public class MiClase <U, T>
{
     U elementoUno;
     T elementoDos;
     ...
}

MiClase<string, int> objeto = new MiClase<string, int>();
```