

**Profesor:** Coronel Castillo, Eric Gustavo

**Integrantes:**

-Juan Rincón

-Marco Ruiz

**Curso:** Desarrollo de Aplicaciones Móviles

**Fecha:** 27/06/17

INDICE

1.Datos Genéricos

1.1. ¿Qué son los datos genéricos? .......................................................................................................... 1

2.Funcion

2.1. Como usamos los genéricos …………………………………………………………………………………………………………. 2

2.2. ¿Qué significan ?, ¿hay diferencias?.................................................................................................. 3

1.Generics

* 1. ¿Que son los Generics?

Los genéricos permiten asignar parámetros a las clases, interfaces, métodos, de forma que sólo admitan los tipos de objetos que se desee. **No** se pueden utilizar con primitivos, pero si con las clases que se corresponden con ellos. Por eso no pue des hacer un genérico tipo int, pero sí un Integer.

Los beneficios que posee es: la comprobación de tipos más fuertes en tiempo de compilación.

La eliminación de casts, aumentando la legibilidad del código y la posibilidad de implementar algoritmos genéricos con tipado seguro.

2. ¿Cómo usamos los genéricos?

Reconoceremos el uso de genéricos en una clase, interfaz o método cuando veamos un solo carácter en mayúsculas encerrado en el operador “diamond”. Por ejemplo:

Como ejemplo, veamos la declaración de la clase ArrayList:

public class ArrayList<E> extends AbstractList implements List, RandomAccess, Cloneable, Serializable

El operador después de ArrayList significa que la clase está implementando genéricos. Si nos fijamos en sus herencias e implementaciones, veremos que AbstractList y List también usan genéricos.

Para usar una clase que implemente genéricos, solo deberemos inicializar y declarar esa clase usando el operador diamante con la clase deseada.

ArrayList<String> str = new ArrayList<String>(); // Un ArrayList que solo permite Strings

Como se puede ver, usamos el operador diamante con la clase String de la siguiente forma: De esta forma, le decimos a Java que queremos un ArrayList que solo acepte instancias de la clase , si intentamos añadir otra cosa, tendremos un bonito error en tiempo de compilación.

Una vez entendida la funcionalidad de los genéricos, podemos decir que su sintaxis es similar (ignorando el operador diamante) a la de la declaración de parámetros de cualquier método, ya que una misma clase o método puede usar todos los genéricos que necesite.

Veamos la declaración de la interfaz Map:

public interface Map<K, V>

Como se ve, utiliza dos genéricos. Esto es que, a la hora de inicializarlo o declararlo deberemos usar dos nombres de Clase en el interior del operador diamante. La funcionalidad, evidentemente, dependerá de la clase. Para que nos entendamos mediante la práctica, voy a poner un ejemplo de que clase que impelementa la interfaz Map y, por tanto, usa dos genéricos. Esta clase es la clase HashMap.

Declaración de la clase, para que sepamos que hacemos:

public class HashMap<K, V> extends AbstractMap implements Cloneable, Serializable

Ejemplo de clase que usa dos genéricos:

1. HashMap <Integer, String> h = new HashMap<Integer, String>();
2. h.put(new Integer(1), "uno");
3. h.put(new Integer(2), "dos");
4. h.put("tres", 3); // Esto dará un error, ya que "tres",3 no se corresponde con la declaración que hemos hecho

En el ejemplo se crea un HashMap (un array asociativo) que relaciona un objeto Integer con un String. En concreto, relaciono un número con su representación lexicográfica. Si intentamods crear una relación llave -> valor que no sea Integer -> String, tendremos un error de compilación, como se ve en la línea 4.

2.2 ¿Qué significan ?, ¿hay diferencias?

Esta es una duda que tiene mucha gente y la respuesta es simple: no. Los genéricos pueden nombrarse como queramos, como si se tratasen de parámetros. Los genéricos tienen nomenclaturas distintas debido a la convención de nombres que Oracle, amigablemente, nos pide que usemos.

* E – Para nombrar “elementos” de una colección
* K – Para nombrar identificadores en una relación asociativa
* N – Para nombrar genéricos que, esperamos, han de ser numéricos
* V – Para nombrar valores en una relación asociativa
* ETC,