Hola, te saluda nuevamente Ubaldo Acosta. Espero que estés listo para comenzar con esta lección.

Vamos a estudiar el tema de Genéricos en Java.

¿Estás listo? ¡Vamos!

GENÉRICOS EN JAVA

Definición de una clase genérica:

```
//Definimos una clase generica con el operador diamante <>
public class ClaseGenerica<T> {

    //Definimos una variable de tipo generico
    T objeto;

    //Constructor que inicializa el tipo a utilizar
    public ClaseGenerica(T objeto) {
        this.objeto = objeto;
    }

    public void obtenerTipo() {
        System.out.println("El tipo T es: " + objeto.getClass().getName());
    }
}
```

Uso de un tipo o clase genérico:

```
    public static void main(String[] args) {
        // Creamos una instancia de ClaseGenerica para Integer.
        ClaseGenerica<Integer> objetoInt = new ClaseGenerica<Integer>(15);
        objetoInt.obtenerTipo();
    }
```

Vamos a estudiar en esta lección los tipos genéricos en Java. Los tipos genéricos o generics, se introdujeron en la versión 1.5 de Java SE, y fue uno de los mayores cambios para esta versión.

Anteriormente teníamos que saber exactamente el tipo de datos que íbamos a utilizar para poder pasar por ejemplo parámetros a una función, o para instanciar una clase, sin embargo con ayuda de los genéricos podemos dejar pendiente el tipo de dato hasta el momento de instanciar alguna clase genérica, o del paso de un parámetro genérico.

Podemos observar en el código mostrado una clase que define un tipo genérico con el uso del operador diamante <> en la definición de la clase. Esto quiere decir que el tipo lo conoceremos hasta el momento en que sea utilizada esta clase, y por ello se conoce como tipo genérico.

La T significa básicamente un tipo genérico que será reemplazado al momento de utilizar esta clase, por lo que la T es simplemente el nombre que le damos al tipo genérico que vamos a utilizar. Existen varios nombres que podemos utilizar como veremos en la siguiente lámina.

Posteriormente en la clase definimos un atributo del mismo tipo T, el cual lo inicializaremos por medio del constructor de esta clase, y podemos ver que cuando utilizamos esta clase en el método main mostrado, al momento de crear el objeto de esta clase, especificamos por medio del operador diamante el tipo de dato que nos interesa utilizar.

Y aquí es aquí donde el uso de clases genéricas es muy útil, ya que nos permite definir el tipo de dato que queremos utilizar hasta el momento de instanciar un objeto de nuestra clase, y así definir funcionalidad cada vez más genérica valga la redundancia, de hecho al utilizar interfaces, clases abstractas y polimorfismo ya hemos creado varios métodos genéricos ya que pueden ser utilizados por varios tipos de datos, sin embargo, este concepto de Generics va un paso más allá, y nos permite definir el tipo a utilizar y crear métodos aún más genéricos de los que hemos creado.

Finalmente el método obtenerTipo() nos permite mandar a consola el tipo de dato que hemos definido para la clase genérica, y así como hemos utilizado el tipo <Integer> para instanciar nuestra clase genérica, podemos utilizar cualquier tipo de datos, sin embargo no podemos utilizar tipos primitivos, sino solo tipos Object.

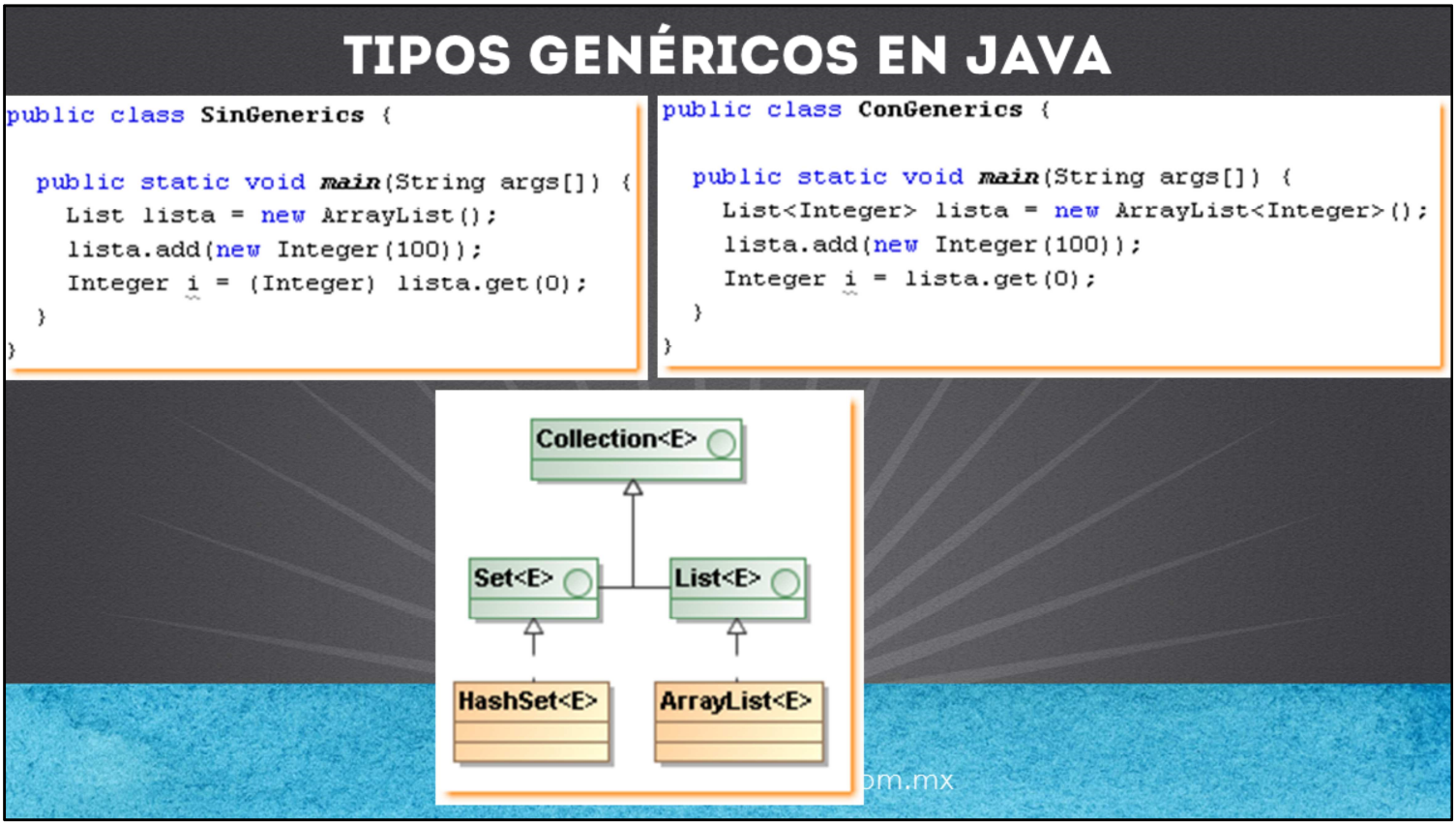
TIPOS GENÉRICOS EN JAVA

Tipos Genéricos que se pueden utilizar:

Nombre Tipo Genérico	Significado del Tipo Genérico
E	Element (utilizado generalmente por el framework de Colecciones de Java)
K	Key (Llave, utilizado en mapas)
N	Number (utilizado para números)
T	Type (representa un tipo, es decir, una clase)
V	Value (representa un valor, también se usa en mapas)
S,U,V etc.	Usado para representar otros tipos.

Al utilizar y crear tipos genéricos, podemos utilizar varios nombres, como la T que ya hemos utilizado. Sin embargo, existen algunas convenciones que podemos seguir al momento de crear nuestras clases genéricas y así utilizar buenas prácticas al momento de crearlas. Obviamente también nos es útil al momento de usar estos tipos genéricos en clases ya creadas dentro del Api de Java, ya que sabremos con mayor certeza para que sirve cada clase y tipo genérico que estamos utilizando.

A continuación veremos cómo aplicar los tipos genéricos al framework de Colecciones de Java.



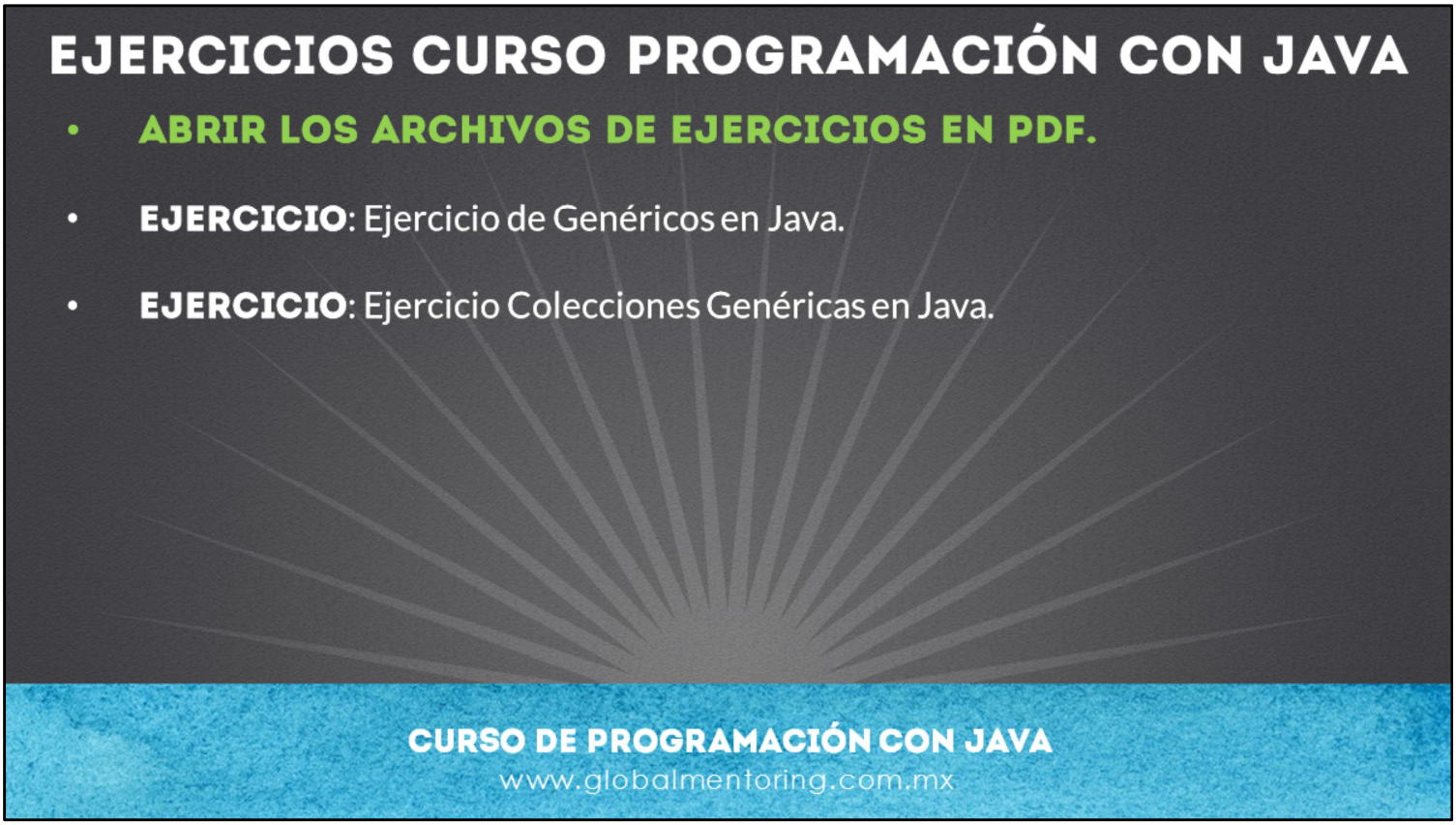
El tema de genéricos, fue mayormente utilizado para cambiar varias de las APIs de Java, pero en particular el API de Framework Collections fue de los más beneficiados, ya que como podemos ver, anteriormente era necesario definir un tipo lista, sin embargo no había una certeza de que almacenara un solo tipo y así evitar problemas de conversión de objetos al momento de extraer sus elementos. Por ello anteriormente las listas almacenaban un tipo object, y al momento de extraer el objeto de la lista el compilador tenia un object, y por tanto era necesario hacer un casting del objeto extraido, como se observa en el código de la clase SinGenerics.

Sin embargo con las colecciones genéricas, podemos declarar una lista de un tipo en concreto, y agregar a esta lista solo los tipos o subtipos compatibles según las reglas de downcasting que mencionamos en el tema de conversión de objetos.

Como podemos observar en el ejemplo de ConGenerics, observamos que para definir la lista estamos utilizando el operador diamante y estamos especificando que los objetos que podemos agregar a esta lista son exclusivamente de tipo Integer o subclases del tipo Integer, de esta manera cuando extraemos los valores de esta lista estamos seguros que serán de un tipo compatible con el tipo Integer, y por tanto no es necesario hacer un casting o conversión del objeto extraído.

Por lo tanto, las colecciones genéricas ofrecen la seguridad de que en tiempo de compilación los datos a almacenar son sólo del tipo especificado, eliminan la necesidad de hacer un casting (conversión de tipos). Solo debemos recordar que se necesita la versión 1.5 o mayor del JDK.

En la figura podemos observar cómo se cambió la definición de las interfaces y clases del API de Coleccions, y ahora se utilizan tipos genéricos en su definición. Veamos a continuación algunos ejemplos del uso de colecciones genéricas.



En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Programación con Java
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx