Argentina Programa

Clase 20: Estructura de datos - Generics - Arrays - List

Agenda

Familiarizarse con los conceptos básicos relacionados a Estructura de datos - Generics - Arrays - List

- Estructura de datos - Generics - Arrays - List
- Práctica Estructura de datos - Generics - Arrays - List.

Estructura de datos

Las estructuras de datos en Java son conjuntos de valores, variables y tipos de datos organizados para soportar operaciones específicas como agregar y eliminar elementos, buscar elementos, ordenar elementos, etc.

Estructura de datos

Algunas de las estructuras de datos más comunes en Java incluyen:

- Arrays
- Listas
- Pilas
- Colas
- Conjuntos
- Mapas

Generics

En Java, los tipos genéricos permiten escribir código que puede ser utilizado con diferentes tipos de datos, sin necesidad de especificarlos en el momento de escribirlo. Esto mejora la reutilización del código y reduce los errores de tiempo de compilación.

Los tipos genéricos se declaran con angle brackets <> y pueden ser utilizados en clases, interfaces y métodos.

Generics

Fueron introducidos en Java 5 y permiten a los desarrolladores crear clases, interfaces y métodos que pueden trabajar con cualquier tipo de datos, sin tener que especificar exactamente qué tipo de datos se utilizará.

Esto significa que pueden ser utilizados para crear colecciones de objetos, estructuras de datos y algoritmos, entre otros, que sean genéricos y sean útiles para muchos tipos diferentes de datos.

Generics

Ejemplo

Contenedores de datos: Por ejemplo, se puede crear una clase genérica MyList que almacene una lista de objetos de cualquier tipo:

```
public class MyList<T> {
	private List<T> list = new ArrayList<>();

	public void add(T item) {
		list.add(item);
	}

	public T get(int index) {
		return list.get(index);
	}
}
```

Generics

Ejemplo

Clases anidadas genéricas: Por ejemplo, se puede escribir una clase anidada genérica MyNestedClass dentro de una clase normal:

```
public class MyClass {
	public static class MyNestedClass<T> {
		private T t;

		public MyNestedClass(T t) {
			this.t = t;
		}
		public T getValue() {
			return t;
		}
	}
```

Arrays

Un array en Java es una estructura de datos que contiene una colección ordenada de elementos del mismo tipo.

Cada elemento en el array se identifica con un índice numérico, comenzando desde 0.

Arrays

Especificando el tamaño del array al momento de la creación.

```
public class Main {
	public static void main(String[] args) {
		int[] numbers = new int[5];
		numbers[0] = 3;
		numbers[1] = 1;
		numbers[2] = 4;
		numbers[3] = 2;
		numbers[4] = 5;
		System.out.println("Números en el array: ");
		for (int i = 0; i < numbers.length; i++) {
			System.out.println("Número en la posición " + i + ": " + numbers[i]);
		}
	}
}
```

Arrays

Especificando los elementos del array al momento de la creación.

```
int[] numbers = {1, 2, 3, 4, 5};
for (int i = 0; i < numbers.length; i++) {
	System.out.println(numbers[i]);
}
```

List

La clase List en Java es una interfaz que representa una estructura de datos de lista.

Una lista es una colección ordenada de elementos que se pueden acceder por un índice.

La clase List proporciona una serie de métodos para insertar, eliminar, buscar y manipular elementos en la lista.

List

Algunas implementaciones comunes de la interfaz List incluyen:

- ArrayList
- LinkedList
- Vector

List - ArrayList

ArrayList es una clase de la biblioteca de Java Collections que representa una lista de objetos.

A diferencia de los arrays tradicionales en Java, ArrayList es dinámico en términos de tamaño, lo que significa que puedes agregar y eliminar elementos después de que la lista se haya creado.

List - ArrayList

```
import java.util.ArrayList;

public class ArrayListExample {
	public static void main(String[] args) {
		// Creación de una lista de enteros
		ArrayList<Integer> numbers = new ArrayList<>();
		// Agregar elementos a la lista
		numbers.add(10);
		numbers.add(20);
		numbers.add(30);

		// Obtener el tamaño de la lista
		System.out.println("Tamaño de la lista: " + numbers.size());

		// Acceder a un elemento en una posición específica
		System.out.println("Tercer elemento: " + numbers.get(1));

		// Eliminar un elemento
		numbers.remove(2);

		// Recorrer la lista usando un bucle for
		System.out.println("Elementos en la lista:");
		for (int i = 0; i < numbers.size(); i++) {
			System.out.println(numbers.get(i));
		}
	}
```

List LinkedList

LinkedList es una clase de la biblioteca de Java Collections que representa una lista enlazada de objetos.

A diferencia de ArrayList, que almacena sus elementos en una matriz, LinkedList almacena sus elementos en una serie de nodos, donde cada nodo contiene un elemento y un enlace a otro nodo.

List LinkedList

```
import java.util.LinkedList;

public class LinkedListExample {
        public static void main(String[] args) {
        // Creación de una lista de cadenas
                LinkedList<String> colors = new LinkedList<>();

                // Agregar elementos a la lista
                colors.add("Red");
                colors.add("Green");
                colors.add("Blue");

                // Obtener el tamaño de la lista
                System.out.println("Tamaño de la lista: " + colors.size());

                // Agregar un elemento al principio de la lista
                colors.addFirst("Orange");

                // Agregar un elemento al final de la lista
                colors.addLast("Pink");

                // Recorrer la lista usando un bucle for
                System.out.println("Elementos en la lista:");
                for (String color : colors) {
                                System.out.println(color);
                }
        }
}
```

List - Vector

Vector es una clase de la biblioteca de Java Collections que representa una estructura de datos de tamaño dinámico que permite agregar, eliminar y acceder a sus elementos de manera eficiente.

Vector es similar a ArrayList, pero Vector es sincronizado, lo que significa que es seguro para ser usado por varios hilos en forma simultánea.

Aunque Vector es una clase antigua en Java y se recomienda usar ArrayList en su lugar, debido a su sincronización.

CONSULTAS?

Muchas Gracias!