Grado Superior
Programación

Genéricos

COLEGIO litterator

Curso 2023/24

Profesor
Jonathan Carrero

Java

Genéricos

Introducción

Los tipos genéricos, introducidos en Java 5, permiten definir clases que contienen atributos de un tipo **genérico**.

Por ejemplo, la clase `ArrayList` es un genérico.

Introducción

Los tipos genéricos, introducidos en Java 5, permiten definir clases que contienen atributos de un tipo **genérico**.

Por ejemplo, la clase `ArrayList` es un genérico.

```
ArrayList<Integer> a = new ArrayList<Integer>();
 ArrayList<String> b = new ArrayList<String>();
```

Introducción

Los tipos genéricos, introducidos en Java 5, permiten definir clases que contienen atributos de un tipo **genérico**.

Por ejemplo, la clase `ArrayList` es un genérico.

```
ArrayList<Integer> a = new ArrayList<Integer>();
ArrayList<String> b = new ArrayList<String>();
```

Antes se utilizaba la clase `Object` aprovechando el polimorfismo. Pero manejar sólo la clase `Object` provoca varios problemas, sobre todo porque nos vemos forzados a realizar `casting` constantemente.

Genéricos, ejemplo del mal uso de `Object`

```
public class GenericTableObject {
	final static int MAX_SIZE = 100;
	int numElem;
	Object[] vector;

	public GenericTableObject() {
		this.numElem = 0;
		vector = new Object[MAX_SIZE];
	}
	public void add(Object o) {
		if (this.numElem < MAX_SIZE){
			vector[this.numElem] = o;
			this.numElem++;
	}
	public int getNumElem(){
		return this.numElem;
	}
	public Object getElem(int i){
		if (i < this.numElem)
			return this.vector[i];
		else
			return null;
	}
	public String toString(){
		String s = "";
		for (int i = 0; i < this.numElem; i++)
			s = s + this.vector[i].toString()+ "\n";
		return s;
	}
}
```

Genéricos, ejemplo del mal uso de `Object`

Como los elementos del array están definidos del tipo `Object`, y toda clase hereda de `Object`, es correcto hacer:

El array generado es **heterogéneo**, es decir, contiene elementos que no tienen **ninguna** relación entre ellos.

Genéricos, ejemplo del mal uso de `Object`

Como los elementos del array están definidos del tipo `Object`, y toda clase hereda de `Object`, es correcto hacer:

```
GenericTableObject a = new
GenericTableObject();
a.add(2);
a.add("tp");
a.add(2.0);
System.out.println(a.toString());
```

El array generado es **heterogéneo**, es decir, contiene elementos que no tienen **ninguna** relación entre ellos.

Genéricos, ejemplo del mal uso de `Object`

Como los elementos del array están definidos del tipo `Object`, y toda clase hereda de `Object`, es correcto hacer:

```
GenericTableObject a = new
GenericTableObject();
a.add(2);
a.add("tp");
a.add(2.0);
System.out.println(a.toString());
```

¿Hay errores de compilación? Puede que no, pero si otra clase `A` instancia objetos de tipo `GenericTableObject`, si `A` accede a la información de los atributos de `GenericTableObject`, entonces tendría que forzar haciendo `casting` porque el vector declarado es de tipo `Object`.

El array generado es **heterogéneo**, es decir, contiene elementos que no tienen **ninguna** relación entre ellos.

Genéricos

El uso de genéricos **impide** que los arrays sean heterogéneos. Sin entrar en mucho detalle, la razón principal es que los arrays de Java contienen, en **tiempo de ejecución**, información sobre el tipo de su componente. Así que debemos conocer el tipo del componente cuando creamos el array. Y como no sabemos qué T es en tiempo de ejecución (podría ser `Integer`, `Double`, `Boolean`…), no podemos crear el array.

https://es.stackoverflow.com/questions/4724/c%C3%B3mo-inicializar-un-arreglo-con-par%C3%A1metro-generico-t

Genéricos

El uso de genéricos **impide** que los arrays sean heterogéneos. Sin entrar en mucho detalle, la razón principal es que los arrays de Java contienen, en **tiempo de ejecución**, información sobre el tipo de su componente. Así que debemos conocer el tipo del componente cuando creamos el array. Y como no sabemos qué T es en tiempo de ejecución (podría ser `Integer`, `Double`, `Boolean`…), no podemos crear el array.

https://es.stackoverflow.com/questions/4724/c%C3%B3mo-inicializar-un-arreglo-con-par%C3%A1metro-generico-t

La sintaxis más simple para definir una clase genérica es:

```
Public class NombreClase<T1, …, Tn> {
        // Implementación de la clase
}
```

Genéricos, ejemplo de un buen uso

```
abstract public class Arithmetics<T> {
	protected T x;
	protected T y;

	public Arithmetics(T xx, T yy){
		this.x = xx;
		this.y = yy;
	}
	abstract public T add();
	abstract public T sub();
	abstract public T div();
	abstract public T mult();
}
```

Genéricos, ejemplo de un buen uso

```
public class ArithmeticInteger extends
Arithmetics<Integer>{
	public ArithmeticInteger(Integer xx, Integer yy) {
		super(xx, yy);
	}
	public Integer add() {
		return this.x + this.y;
	}
	public Integer sub() {
		return this.x - this.y;
	}
	public Integer div() {
		return this.x / this.y;
	}
	public Integer mult() {
		return this.x * this.y;
	}
}
```

```
public class ArithmeticDouble extends
Arithmetics<Double> {
	public ArithmeticDouble(Double xx, Double yy) {
		super(xx, yy);
	}
	public Double add() {
		return this.x + this.y;
	}
	public Double sub() {
		return this.x - this.y;
	}
	public Double div() {
		return this.x / this.y;
	}
	public Double mult() {
		return this.x * this.y;
	}
}
```

Genéricos, ejemplo de un buen uso

Los parámetros sin instanciar de una clase genérica **pueden extender** a cualquier otra clase **sea genérica o no**.

```
public class A<T extends B>
public class A<T extends B<T>>
```

Grado Superior
Programación

Genéricos

Curso 2023/24

Profesor
Jonathan Carrero