Il tipo Generics in Java

I **Generics** sono stati introdotti a partire dalla versione J2SE 5, il termine ha un significato di **tipo parametrizzato**. I tipi parametrizzati rivestono particolare importanza perché consentono di creare classi, interfacce e metodi per i quali il tipo di dato sul quale si opera può essere specificato come parametro. Parliamo quindi di classi, interfacce e metodi generici.

Attraverso un codice generico possiamo realizzare un algoritmo che astrae dal tipo di dato specifico sul quale opera. Ad esempio un algoritmo per l'ordinamento di entità numeriche è lo stesso qualunque sia il tipo numerico che si utilizza; il tipo numerico utilizzato rappresenta quindi un parametro.

Vediamo un semplice esempio. Supponiamo di voler modellare il comportamento di un braccio meccanico che deve afferrare un bottiglia con un determinato contenuto posta su un tavolo. Il comportamento del braccio sarà lo stesso qualunque sia il contenuto della bottiglia che afferra. Il movimento del braccio può essere visto come l'algoritmo indipendente dal tipo di bottiglia sul quale agisce. Il contenuto della bottiglia determina invece un parametro generico. Ora modelliamo il tutto in classi Java che facciano uso di Generics.

Iniziamo con il definire due semplici classi per le tipologie di contenuto di una bottiglia:

```
package it.html.java.generics;

public class Acqua {

	@Override
	public String toString(){
		return " una bottiglia d'acqua";
	}
}

public class Vino {

	@Override
	public String toString(){
		return " una bottiglia di vino";
	}
}
```

Definiamo quindi la classe `Bottiglia` come classe Generics:

```
package it.html.java.generics;

public class Bottiglia<T> {

	private T contenuto;

	public Bottiglia(T t){
		contenuto=t;
	}

	public T getContenuto() {
		return contenuto;
	}

}
```

Una classe Generics specifica l'uso di un parametro di tipo con la notazione "simbolo minore – carattere – simbolo maggiore", nel nostro esempio, una convezione sintattica prevede di usare lettere maiuscole per il carattere e alcuni caratteri tipici sono T, E, V e K. Nel nostro caso abbiamo

definito una bottiglia in cui il parametro variabile è rappresentato dal suo contenuto `T`, il parametro `T` può essere visto come un segnaposto. All'atto della creazione dell'oggetto sarà sostituito, come vedremo tra poco, da un tipo classe specifico.

Continuiamo con la definizione della classe relativa al braccio meccanico:

```
package it.html.java.generics;

public class BraccioAutomatico {

	public void prendiBottiglia(Bottiglia<?> bottiglia){
		System.out.println("Ho preso"+bottiglia.getContenuto());
	}
}
```

Ignorate per il momento il significato preciso del carattere `?`, è sufficiente comprendere che stiamo dichiarando una "bottiglia generica", indipendentemente dal suo contenuto. La classe modella quindi il comportamento del braccio che è in grado di afferrare una bottiglia senza preoccuparsi del suo contenuto. Completiamo invece l'esempio con una classe `Demo` che illustri il funzionamento:

```
package it.html.java.generics;

public class Demo {

	public static void main(String[] args) {
		Bottiglia<Acqua> bottiglia1= new Bottiglia<Acqua>(new Acqua());
		Bottiglia<Vino>  bottiglia2= new Bottiglia<Vino>(new Vino());

		BraccioAutomatico braccio = new BraccioAutomatico();
		braccio.prendiBottiglia(bottiglia1);
		braccio.prendiBottiglia(bottiglia2);
	}

}
```

Se eseguiamo il codice avremo come stampa sulla console:

```
Ho preso una bottiglia d'acqua
Ho preso una bottiglia di vino
```

Possiamo notare come in fase di creazione dell'oggetto `Bottiglia` sia stato specificato, al posto del parametro, il tipo classe `Acqua` in un caso e il tipo classe `Vino` nell'altro, definendo cosi due tipi di bottiglia. La classe `BraccioAutomatico` eseguirà quindi la stessa azione per due tipi di bottiglie differenti. Non siamo poi limitati all'uso di un solo parametro, possiamo infatti definire **classi generiche con più parametri**:

```
public class GenericClass<T,E,K> {...}
```

Per garantire compatibilità con le versioni precedenti alla J2SE 5, l'implementazione in Java dei Generics viene realizzata attraverso la tecnica nota con il nome di **Erasure**. Quando viene compilato il codice sorgente, tutti i parametri di tipo vengono sostituiti dal loro tipo limite (vedremo successivamente il significato) che in mancanza di una specifica diversa è `Object`.

Avviene quindi il cast appropriato da parte del compilatore che si preoccupa di gestire la compatibilità del tipo. In sostanza i Generics sono semplicemente un meccanismo del codice sorgente e non esistono parametri di tipi in fase di esecuzione.

http://www.html.it/pag/18028/il-tipo-generics-in-java/