Más de Java

Clases Genéricas, Iteradores y operador for-each

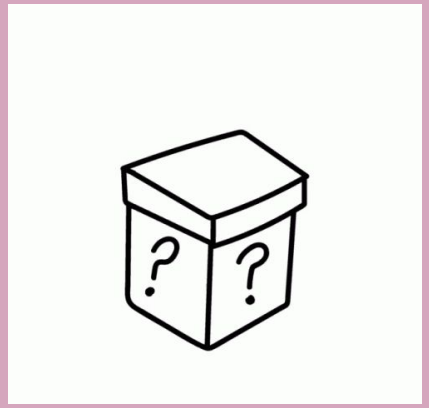
Clases genéricas

Al hacer una clase genérica permite que los tipos sean parámetros al definirla, de tal forma que la clase se especializa en uno o varios tipos específicos.

Para hacer una clase genérica se ponen los tipos a continuación del nombre de la siguiente manera:

class MiClase<T_1, T_2, … , T_n> { … }

Ejemplo



Clase no genérica:

```
public class Caja{
   private Object objeto;
   public void setObjeto(Object objeto) { this.objeto = objeto; }
   public Object getObjeto( ) { return objeto; }
}
```

Clase genérica:

```
public class CajaGenerica<T>{
   private T objeto;
   public void setObjeto(T objeto) { this.objeto = objeto; }
   public T getObjeto( ) { return objeto; }
}
```

Convenciones en el nombrado de tipos

Los nombres de los tipos deben ser una sola letra mayúscula. Los nombres más frecuentes son:

- E - Element
- K - Key
- N - Number
- T - Type
- V - Value
- S, U, V, ... - Tipos adicionales

Con el operador diamante < > se pueden crear instancias de una clase genérica:

```
MiClaseGenerica<Integer> objeto = new MiClaseGenerica<Integer>( );
```

Además ya no es necesario hacer castings o casteos al trabajar con el tipo que recibe la clase genérica.

Ejemplo

Usando la clase no genérica:

```
Caja caja1 = new Caja( );
caja1.setObjeto( “mi cadenita” );
String cadena = ( String ) caja1.getObjeto( );
caja1.setObjeto( new Gato( ) );
```

Usando la clase genérica:

```
CajaGenerica<String> caja2 = new CajaGenerica<String>( );
caja2.setObjeto( “mi cadenita” );
String cadena = caja2.getObjeto( );
// caja2.setObjeto( new Gato( ) );
```

Iteradores

En Java los iteradores permiten visitar cada elemento de una colección independientemente de cómo esté implementada.

```
public interface Iterable<T> {

        public Iterator<T> iterator( );
        :
        :

}
```

```
public interface Iterator<T> {

        public boolean hasNext( );
        public T next ( );
        :
        :

}
```

Ejemplo

```
public class MiColeccion<T> implements Iterable<T> {
   .
   .
   .
   public static void main( String[ ] args ){
      MiColeccion<Persona> alumnos = new MiColeccion<Persona> ( ) ;
      .
      .
      .
      Iterator<Persona> iterador = alumnos.iterator( );
      while ( iterador.hasNext( ) ) {
         Persona alumno = iterador.next ( );
         System.out.println( alumno.getNombre( ) );
      }
   }
}
```

Operador for-each

Si una clase es de tipo Iterable, entonces se puede visitar cada elemento de la colección con el operador for-each, que espera el tipo de cada elemento y la colección:

```
for( Tipo var : coleccion) { … }
```

Ejemplo

Iterando sobre un arreglo de enteros, para obtener el promedio.

```
int[ ] arreglo = new int[ 12 ];
```

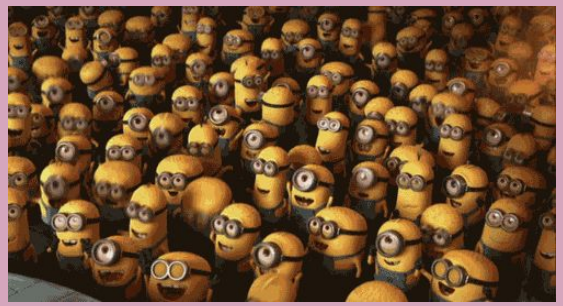
Usando un for tradicional:

```
double suma = 0;
for ( int i = 0; i < arreglo.length; i++ ) {
   suma += arreglo [ i ];
}
double promedio = suma / arreglo.length;
```

Usando un for-each:

```
double suma = 0;
for ( int numero: arreglo ) {
   suma += numero;
}
double promedio = suma / arreglo.length;
```

Ejemplo



```
public class MiColeccion<T> implements Iterable<T> {
    .
    .
    .
   public static void main( String[ ] args ){
      MiColeccion<Persona> alumnos = new MiColeccion<Persona> ( ) ;
      .
      .
      .
      for ( Persona alumno: alumnos ) {
         System.out.println( alumno.getNombre( ) );
      }
}
```

Continuará...