Grado en Ingeniería Informática

Metodología de la Programación

Sesión 9

Genericidad

Docentes:

Raúl Marticorena

Félix Nogal

Índice de contenidos

1. Introducción

En este tema se introduce el concepto y uso de **genericidad en Java**, directamente a través de la realización de una serie de modificaciones sobre el código obtenido en la **sesión previa de prácticas, sobre herencia.**

Para ello, una vez presentada la teoría, se solicita implementar y usar una estructura de datos genérica.

2. Objetivos

El objetivo es familiarizarse con el uso de clases genéricas en Java. Se plantea un ejercicio a resolver en dos partes:

- **Construir** una clase genérica, relativamente simple, reutilizando para ello interfaces y clases genéricas ya disponibles en el paquete `java.util`, incluidas en el núcleo de Java.
- **Utilizar** dicha clase genérica, con distintos tipos de datos, para comprobar las posibilidades de reutilización.

3. Contenido

3.1 Ejercicio con lista acotada

- **Construir** una **clase genérica** denominada `genericidad.Lista`, que puede guardar **cualquier tipo de elemento (cualquier tipo de objeto)**.
- En el constructor de dicha clase se pasa el **número de elementos máximo** que puede almacenar, al tratarse de una **lista acotada en tamaño**.
- Para su implementación interna, se combinará el uso de la **interfaz genérica** **`java.util.List`** y la **clase genérica concreta** **`java.util.ArrayList`**[1]. Por lo tanto para implementar los métodos se delegará en atributos `List` y en objetos `ArrayList` que **ya incluyen métodos equivalentes a los solicitados (se deben reutilizar, simplificando la implementación)**. Se recomienda leer la documentación en línea (ver 5. Bibliografía).
- Recordemos que para realizar una reserva inicial de tamaño, es necesario inicializar el `ArrayList` con elementos **`null`** en el constructor de la clase `Lista`, tal y como se muestra en Código 1, para que se dimensione (reserve espacio) de manera adecuada, en base al número de elementos indicado:

```
for (int i = 0 ; i < número ; i++) { // suponemos una reserva de número
  arrayImplementacion.add(null);
}
```

Código 1: Ejemplo de reserva de tamaño inicial añadiendo nulos, tantos como indique número

Se implementarán los siguientes métodos en la clase `Lista`:

1 Se recomienda consultar la documentación en línea. Ver apartado 6Recursos.

- `set`: para establecer un **valor** en una determinada **posición** (argumentos posición y valor). Si la posición no está en los límites de la lista, se ignora la petición (no se hace nada). Si la posición estaba ocupada, se sustituye el objeto actual por el pasado como argumento. Este método tiene un tipo de retorno **`void`**.
- `get`: para leer el **valor** almacenado en una posición (argumento posición). Si la posición no está en los límites de la lista se retorna un valor **`null`**.
- `size`: devuelve el tamaño de la lista (entero). El método no recibe argumentos. Debe coincidir con el número inicial de nulos que se añadieron en el constructor incialmente, y ser un valor constante. No coincide con el número de elementos añadidos en tiempo de ejecución posteriormente, diferentes de **`null`**.

- **Utilizar** dicha clase `genericidad.Lista`, en el `main` de una clase `genericidad.Principal`[2] para:
  1. Declarar, instanciar y rellenar una **lista de notas musicales** aleatoriamente mediante un método auxiliar estático `rellenarNotas` y mostrar luego su contenido en pantalla en un método auxiliar estático `mostrarNotas`, invocando al método `toString()` de la clase `Nota`.
     - La lista una vez declarada e instanciada **solo** debe poder almacenar objetos de tipo `Nota`.
  2. Declarar, instanciar y rellenar una **lista de instrumentos musicales** aleatoriamente mediante un método auxiliar estático `rellenarInstrumentos` y mostrar luego su contenido en pantalla en un método auxiliar estático `mostrarInstrumentos`, su contenido invocando al método `toString()` de la clase `Instrumento`. Nota: puede ser necesario redefinir el método `toString()` en dicha clase para una visualización más amigable que muestre el estado del objeto.
     - La lista una vez declarada e instanciada **solo** debe poder almacenar objetos de tipo `Instrumento`.
  3. Crear un único método estático `mostrar` que sea reutilizado en la visualización de **ambas** listas de los pasos 1 y 2 (en lugar de tener dos métodos diferentes `mostrarNotas` y `mostarInstrumentos` para mostrar una u otra lista).
     - Repasar el concepto de subtipado en genericidad y su problemática.
     - Revisar los conceptos avanzados del tema de genericidad (**acotación** por subtipado, tipo desconocido, **genericidad restringida**, **paramétros genéricos de método**, etc.) para resolver este ejercicio.
  4. Comprobar el efecto de **acotar** o **restringir** el tipo de parámetro formal de la clase `genericidad.Lista` a `Instrumento`. Comentar (con `/*` y `*/`) el código introducido para manejar la lista de notas musicales (dado que debería generar errores una vez cambiada la acotación) y modificar la signatura del método `mostrar` para poder acceder a los métodos de `Instrumento`.
     - Comprobar que el uso de la genericidad restringida **limita** el posible conjunto de tipos sustitución, pero por otro lado **amplia** el posible conjunto de métodos que se pueden utilizar ahora.

2 Reutilizar el código del Guión de Prácticas: Herencia.

4. Resumen

En esta sesión de prácticas hace un uso práctico de la genericidad, tanto en su variante general, como con genericidad restringida. A la hora de utilizar las listas genéricas creadas, es necesario tener en cuenta que son clases genéricas, con sus correspondiente parámetros formales y sus posibles sustituciones.

5. Bibliografía

[Meyer, 1998] Construcción de Software Orientado a Objetos (1998) Meyer, B. Editorial Prentice Hall, 2ª Ed. Capítulo 10. Genericidad

[Eckel 2007]. Eckel, B. (2007). Piensa en Java. (4ª Edición). Prentice Hall.

[Naftalin & Wadler, 2006] Java Generics and Collections (2006) M. Naftalin & P. Wadler. Ed. O'Reilly

[Meyer, 2009] Meyer, B. (2009) Touch of Class. Learning to program well with objects and contracts. Springer

6. Recursos

Bibliografía complementaria:

[Oracle, 2020] The Java Tutorials. Lesson: Generics (Updated) (2020). Disponible en http://docs.oracle.com/javase/tutorial/java/generics/

[Oracle, 2018] List (Java Platform SE 11). Documentación en línea de la interfaz disponible en https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html

[Oracle, 2018] ArrayList (Java Platform SE 11). Documentación en línea de la clase disponible en https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html

Licencia

Autor: Raúl Marticorena

Área de Lenguajes y Sistemas Informáticos

Departamento de Ingeniería Informática

Escuela Politécnica Superior

UNIVERSIDAD DE BURGOS

2020