



UNIVERSIDADE DA CORUÑA

DESEÑO DAS LINGUAXES DE PROGRAMACIÓN

Práctica 1: Análisis de las características de un conjunto de lenguajes de programación a partir de un caso práctico

AUTORES

Rafael Alcalde Azpiazu
rafael.alcalde.azpiazu
(rafael.alcalde.azpiazu@udc.es)

Eva Suárez García
eva.suarez.garcia
(eva.suarez.garcia@udc.es)

LENGUAJES ANALIZADOS

C
Python
OCaml

A Coruña - 17th September 2017

Contents

| | | |
|----------|---------------------------------------|----------|
| 1 | Implementación en Java | 2 |
| 1.1 | Descripción | 2 |
| 1.2 | Análisis del código | 2 |
| 1.3 | Modificaciones en el diseño | 2 |
| 2 | Implementación en C | 3 |
| 2.1 | Descripción | 3 |
| 2.2 | Análisis | 3 |
| 2.3 | Modificaciones en el diseño | 3 |
| 3 | Implementación en Python | 4 |
| 3.1 | Descripción | 4 |
| 3.2 | Análisis | 4 |
| 3.3 | Modificaciones en el diseño | 4 |
| 4 | Implementación en OCaml | 5 |
| 4.1 | Descripción | 5 |
| 4.2 | Análisis | 5 |
| 4.3 | Modificaciones en el diseño | 5 |

1 Implementación en Java

Para este lenguaje se usó el compilador `javac` en su versión 1.8.0_144.

1.1 Descripción

Java es un lenguaje de programación de alto nivel que sigue el paradigma de orientación a objetos basada en clases, aunque a día de hoy las últimas versiones ya incluyen elementos de otros paradigmas, como funciones-lambda.

Como en sus inicios se pensó en que fuera un lenguaje de propósito general, la compilación del código fuente se traduce a bytecode java, el cual es ejecutado sobre una máquina virtual que si está diseñada para la plataforma final.

Su sintaxis deriva en gran medida de C++, siendo Java un lenguaje de tipado estático y fuerte. Es la propia máquina virtual la encargada de la gestión de memoria mediante un recolector automático de basura, el cual es responsable de gestionar el ciclo de vida, reservando memoria cuando el programador crea un objeto, y liberando memoria cuando se deja de referenciar completamente el objeto usado.

1.2 Análisis del código

Dadas las diferencias entre Pascal (lenguaje estructurado imperativo) y Java (lenguaje orientado a objetos imperativo), la estructura `tNodoA` se han traducido en la clase `Node`, que sigue conteniendo los atributos `key`, `rightChild` y `leftChild`. Como las clases son más abstractas que las estructuras, la programación resultante es de alto nivel y más legible.

Otro elemento que ha cambiado son los punteros y la gestión de memoria manual en Pascal. En Java no hay elementos para manipulación de bajo nivel, por lo que la gestión de memoria es automatizada por la máquina virtual. Esto hace que desaparezca el uso de los punteros `tPosA`.

Relacionado, se ha eliminado el procedure `error`, ya que la máquina virtual de Java, cuando se queda sin memoria, lanza la excepción `OutOfMemoryError`. También se ha eliminado la función `crearNodo`, que se traduce en el constructor de la clase `Node` en Java.

1.3 Modificaciones en el diseño

Por aquí poner todo lo que ha cambiado en la implementación.

2 Implementación en C

2.1 Descripción

2.2 Análisis

2.3 Modificaciones en el diseño

3 Implementación en Python

3.1 Descripción

3.2 Análisis

3.3 Modificaciones en el diseño

4 Implementación en OCaml

4.1 Descripción

4.2 Análisis

4.3 Modificaciones en el diseño