P5

PRÁCTICA EVALUABLE

Desarrollando un traductor de lenguajes de programación

OBJETIVOS DIDÁCTICOS

- Escribir un traductor de lenguajes de programación.
- Consolidar los conceptos estudiados en la unidad 10.
- Repasar conceptos del manejo de cadenas de texto y formateo de salida por pantalla.
- Escribir programas utilizando ficheros como entrada y salida de información.

1. CONTENIDO

A continuación, un listado de los puntos principales de este documento que deberéis leer con detalle para desarrollar con éxito esta práctica:

- Proyecto
- Presentación de conceptos previos
- Enunciado de la práctica
- Criterios de calificación de la práctica.

2. PROYECTO

Vamos a desarrollar un programa que traduzca código fuente del lenguaje BASIC a Java.

- Nombre del proyecto en Eclipse: “TraductorBASICJAVA”

3. PRESENTACIÓN DE CONCEPTOS PREVIOS

El lenguaje BASIC es un lenguaje de programación de los años 70 y 80, de bajo nivel, que consistía en una secuencia de instrucciones ordenadas por un índice numérico, indicando lo que el programa debía hacer. Por ejemplo:

```
10 PRINT “Escribe un numero”
20 INPUT n1
30 PRINT “Escribe otro numero”
40 INPUT n2
50 PRINT “Su suma es”
60 PRINT n1+n2
```

Como puede verse, al inicio de cada instrucción hay número que permite ordenar la secuencia de instrucciones. En el ejemplo anterior, primero se ejecutaría la instrucción PRINT con el número 10, luego el INPUT con el número 20… y así sucesivamente. NO es necesario que los números sean múltiplos de 10, puede ser 1, 2, 3, 4, o 1, 6, 8, 9… lo importante es que el número de una instrucción vaya después de las que queramos que se ejecuten antes que ésa. Así, el siguiente código sería equivalente al anterior, aunque esté desordenado:

```
1 PRINT "Escribe un numero"

3 PRINT "Escribe otro numero"

5 PRINT "Su suma es"

6 PRINT n1+n2

2 INPUT n1

4 INPUT n2
```

Hay que tener en cuenta también que el código fuente puede tener líneas en blanco, que se deben descartar.

4. ENUNCIADO DE LA PRÁCTICA

La práctica consistirá en implementar un traductor sencillo de instrucciones BASIC para construir el programa equivalente en lenguaje JAVA. Así, para el programa del apartado anterior deberíamos producir el siguiente código en JAVA:

```
import java.util.Scanner;

public class Principal {

    public static void main(String[] args) {

        System.out.print("Escribe un numero");

        int n1 = Integer.valueOf((new Scanner(System.in)).next());

        System.out.print("Escribe otro numero");

        int n2= Integer.valueOf((new Scanner(System.in)).next());

        System.out.print("Su suma es");

        System.out.print(n1+n2);

    }

}
```

Distinguiremos únicamente las instrucciones PRINT e INPUT del lenguaje BASIC. Por simplificar, en las instrucciones INPUT siempre almacenaremos datos en variables enteras, como en el ejemplo anterior. Las instrucciones PRINT, por otra parte, pueden o bien imprimir un texto fijo entre comillas dobles, o bien una operación matemática simple con variables.

El programa que deberéis desarrollar deberá hacer los siguiente:

4.1. PROCESAMIENTO DEL CÓDIGO BASIC

- **(0,5 puntos)** En primer lugar, el programa le pedirá al usuario un nombre de fichero de entrada, que contendrá las instrucciones en lenguaje BASIC del programa. Se deberá verificar en este punto que el fichero exista, antes de procesarlo.

- **(1,5 puntos)** Se deberá definir una clase llamada *InstruccionBASIC*, que almacenará los datos de cada instrucción que se procese. En concreto, contendrá como atributos el número de instrucción, el comando en sí (un tipo *Enumerado* que podrá valer PRINT o INPUT), y los parámetros (un *String* con el resto de la instrucción). Además del constructor y los getters/setters, esta clase deberá implementar:
    - La interfaz *Comparable<InstruccionBASIC>*, que permitirá ordenar dos instrucciones de menor a mayor número de instrucción.
    - Un método llamado *imprimirInstruccion*, que volcará la instrucción a consola.

- **(2 puntos)** Una vez verificado que el fichero de entrada existe, se deberán procesar las instrucciones BASIC allí contenidas, generando una lista (*ArrayList*) de objetos de tipo *InstruccionBASIC*, ordenada por número de instrucción.

4.2. SALIDA DEL PROGRAMA

Cuando ya se tenga el listado procesado de instrucciones BASIC, el programa deberá:

- **(2 puntos)** Mostrar en la mitad izquierda de la consola el código BASIC formateado.

- **(1 puntos)** Generar un archivo de salida con el mismo nombre que el de entrada, pero con el sufijo ".java", con el código JAVA generado para ese fichero de entrada. Por ejemplo, si el fichero de entrada se llamaba "codigoBASIC.txt", el de salida se llamará "codigoBASIC.java". Dicho código resultado debe tener una instrucción "import java.util.Scanner;" al principio, una clase (llamada *Principal*) con su método *main*, y dentro, el código BASIC traducido a Java, como se muestra en el ejemplo del apartado 3, debidamente indentado (4 espacios por cada nivel de indentación).

- **(1 puntos)** Mostrar en la mitad derecha de la consola el código JAVA generado, sin contar la clase y ni la cabecera del método main, sólo las instrucciones traducidas (Ver ejemplo de salida de la siguiente página).

Ejemplo de salida por pantalla:

CODIGO BASIC	CODIGO JAVA
`10 PRINT "Escribe un numero"`	`System.out.println("Escribe un numero");`
`20 INPUT n1`	`int n1 = Integer.valueOf((new Scanner(System.in)).next());`
`30 PRINT "Escribe otro numero"`	`System.out.print("Escribe otro numero");`
`40 INPUT n2`	`int n2= Integer.valueOf((new Scanner(System.in)).next());`
`50 PRINT "Su suma es"`	`System.out.print("Su suma es");`
`60 PRINT n1+n2`	`System.out.print(n1+n2);`

5. CRITERIOS DE CALIFICACIÓN

Esta práctica se valorará sobre 8 puntos. En cada ejercicio se especifica su valoración sobre 8.

En el desarrollo del programa se valorará la claridad, estilo y eficiencia en el código.

6. CONSEJOS Y CONSIDERACIONES

Antes de empezar a programar lee bien el enunciado, subraya lo importante, hazte notas, escribe un borrador de las posibles funciones, etc. Intenta entender el problema y cómo realizarlo antes de empezar a escribir código.

No intentes hacerlo todo de golpe. Una estrategia clave en programación es dividir un problema grande en varios problemas pequeños y luego resolverlos uno a uno (es decir, programa y prueba las funciones una a una).

7. RECONOCIMIENTO DE AUTORÍA

Autor de este documento: José Ramón Simó

Esta práctica es una adaptación de una práctica del IES San Vicente (Sant Vicent del Raspeig).