Java a fondo

Apéndice A

Ing. Pablo Augusto Sznajdleder

Alfaomega

Nociones básicas de programación

Apéndice A

Contenido

A.1 Introducción

Las aplicaciones que utilizamos en nuestra computadora fueron programadas en algún lenguaje de programación. El "navegador", el "procesador de texto", la "planilla de cálculo", los "juegos", etc., son ejemplos de programas que alguna persona (o equipo de personas) programó y nosotros los ejecutamos y utilizamos.

A.2 Conceptos iniciales

A.2.1 El lenguaje de programación

Escribir un programa implica detallar una secuencia de instrucciones codificándolas en algún lenguaje de programación.

Los lenguajes de programación de alto nivel permiten escribir estas instrucciones con una terminología fácilmente entendible para las personas.

Por ejemplo, en el siguiente programa escribimos la frase "Hola Mundo" en la pantalla.

```
package libro.apendiceA;

public class HolaMundo
{
   public static void main(String[] args)
   {
      System.out.println("Hola Mundo !!!");
   }
}
```

Este programa está escrito en el lenguaje Java y debe interpretarse de la siguiente manera:

- El programa está ubicado en el "paquete" (`package`) `libro.apendiceA`.
- El programa se llama (`class`) `HolaMundo`.
- En su cuerpo principal (`main`) imprime (`println`) en la salida del sistema (`System.out`) la cadena de caracteres `"Hola Mundo !!!"`.

Más allá de que el lector comprenda realmente o no lo que este programa hace, es innegable que el lenguaje de programación tiene una gran similitud con el idioma inglés.

A.2.2 El compilador

Los programas escritos en Java o en cualquier otro lenguaje de programación son entendibles por nosotros (los programadores) pero no lo son para la computadora. La computadora solo entiende lo que se conoce como código de máquina.

El compilador es un programa que toma nuestro programa codificado en algún lenguaje de alto nivel (programa fuente) y genera el mismo programa pero codificado en código de máquina de forma tal que pueda ser ejecutado en la computadora.

A.2.3 Los intérpretes

Algunos lenguajes son interpretados. Esto significa que el compilador genera un código intermedio que será entendible para un programa interpretador, el cual se ejecutará en nuestra computadora.

A.2.4 Las máquinas virtuales

Una máquina virtual es un software que emula a una computadora y tiene la capacidad de ejecutar programas como si fuera una computadora real.

Las máquinas virtuales permiten maximizar la portabilidad de los programas e introducen un nivel de abstracción y protección entre la computadora real y el software que están ejecutando. Esto último es fundamental si pretendemos ejecutar programas que se descargan desde Internet y su origen es incierto.
Java es un lenguaje que corre dentro de su propia máquina virtual. Cuando compilamos un programa fuente obtenemos un código intermedio que se llama "código de *bytes*" (o *bytecode*). Este código es entendible por la *Java Virtual Machine* (JVM) o *Java Runtime Enviroment* (JRE).

En nuestra computadora ejecutamos el JRE y dentro de este ejecutamos el programa Java.

A.3 Recursos de programación

El recurso principal del que disponemos cuando programamos es la memoria de la computadora. En la memoria podemos almacenar temporalmente datos para utilizarlos durante la ejecución del programa.

El acceso a la memoria se realiza a través de lo que denominamos "variables".

A.3.1 Las variables

Una variable representa un espacio de memoria en el que podemos almacenar temporalmente nuestros datos.

Por cada variable que definimos en nuestro programa se reservará una cantidad finita de *bytes* de memoria. Según sea esta cantidad podremos almacenar mayor o menor cantidad de información.

Por ejemplo: en 1 *byte* podemos almacenar valores numéricos de entre -128 y 127. En 2 *bytes* podremos almacenar valores numéricos de entre -32768 y 32767.

En general no estaremos interesados en cuántos *bytes* de memoria vamos a reservar. Lo habitual es que pensemos en qué datos vamos a querer almacenar temporalmente y en función de estos efectuar la reserva correspondiente.

A.3.2 Tipos de datos

Los datos pueden clasificarse en tipos "numéricos", "alfanuméricos" y "lógicos". Pero podemos ser más detallistas y pensar que dentro de los tipos numéricos podemos hablar, por ejemplo, de tipos enteros y tipos reales.

Java permite definir variables de los siguientes tipos de datos:

- `byte` (1 *byte*)
- `short` (2 *bytes*)
- `int` (4 *bytes*)
- `long` (8 bytes)
- `char` (2 *bytes*)
- `float` (4 *bytes*)
- `double` (8 *bytes*)
- `boolean` (1 *byte*)

Para ejemplificar esto veremos un programa en el que le pedimos al usuario que ingrese un valor numérico entero y luego le agradecemos por haber ingresado ese valor.

```
package libro.apendiceA;

import java.util.Scanner;

public class IngresaValor
{
   public static void main(String[] args)
   {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Ingrese un valor: ");

      int v = scanner.nextInt();

      System.out.println("Gracias por ingresar el valor: "+v);
   }
}
```

Como podemos ver, almacenamos el valor ingresado por el usuario en la variable `v` cuyo tipo de datos es `int`. Gracias a esto pudimos recordarlo para mostrarlo en el agradecimiento.

A.3.3 Operadores aritméticos

Nuestros programas pueden efectuar operaciones aritméticas. Para esto, disponemos de un conjunto de operadores aritméticos tales como `+` (suma), `-` (resta), `*` (multiplicación) y `/` (división), entre otros.
En el siguiente programa, le pedimos al usuario que ingrese dos valores, los sumamos y mostramos el resultado por la pantalla.

```
package libro.apendiceA;

import java.util.Scanner;

public class SumaValores
{
   public static void main(String[] args)
   {
```

```
      Scanner scanner = new Scanner(System.in);
      System.out.print("Ingrese un valor: ");
      int v1  = scanner.nextInt();

      System.out.print("Ingrese otro valor: ");
      int v2  = scanner.nextInt();

      int suma = v1 + v2;

      System.out.println("La suma de "+v1+" + "+v2+" es: "+suma);
   }
}
```

Hasta aquí hemos analizado ejemplos simples en los que no tuvimos que tomar ninguna decisión.

¿Cómo podríamos hacer un programa que determine si una persona es mayor de 21 años o no?

A.3.4 Estructuras de decisión

Las estructuras de decisión sirven para decidir entre ejecutar una u otra acción en función de que se cumpla o no una determinada condición.

En el siguiente programa, le pedimos al usuario que ingrese su nombre y edad. Luego el programa le mostrará un mensaje indicando si es mayor de edad o no.

```
package libro.apendiceA;

import java.util.Scanner;

public class MayorDeEdad
{
   public static void main(String[] args)
   {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Ingrese su nombre: ");
      String nom = scanner.next();

      System.out.print("Ingrese su edad: ");
      int edad  = scanner.nextInt();

      if( edad >= 21 )
      {
         System.out.println(nom+", Ud. es mayor de edad");
      }
      else
      {
         System.out.println(nom+", eres muy joven aun...");
      }
   }
}
```

A.3.5 Estructuras de repetición

Estas estructuras nos permiten repetir un conjunto de acciones mientras se cumpla una determinada condición.

En el siguiente programa, leemos nombres y edades de personas mientras todas estas sean mayores de edad. Si se filtra algún menor el programa finalizará indicando la causa del problema.

```
package libro.apendiceA;

import java.util.Scanner;

public class SoloMayores
{
   public static void main(String[] args)
   {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Ingrese su nombre: ");
      String nom = scanner.next();

      System.out.print("Ingrese su edad: ");
      int edad  = scanner.nextInt();

      while( edad >=21 )
      {
         System.out.println("Ingreso: "+nom+"("+edad+" anios)");

         System.out.print("Ingrese otro nombre: ");
         nom = scanner.next();

         System.out.print("Ingrese su edad: ");
         edad  = scanner.nextInt();
      }

      System.out.println("Cuidado, un menor entre vosotros...");
   }
}
```