2.A. Variables e identificadores

Sitio: Aula Virtual
Curso: Programación_DAM
Libro: 2.A. Variables e identificadores
Imprimido por: LUIS PUJOL
Día: miércoles, 20 de noviembre de 2019, 02:31

Tabla de contenidos

1 Definicion de variables e identificadores.

Un programa maneja datos para hacer cálculos, presentarlos en informes por pantalla o impresora, solicitarlos al usuario, guardarlos en disco, etc. Para poder manejar esos datos, el programa coge los datos del teclado, de un fichero, de una base de datos, ... Estos datos que coge un programa, pasan directamente a la memoria RAM, y para ello hay que usar variables, donde se guardarán estos datos (de forma temporal).
Una variable es una zona en la memoria del computador con un valor que puede ser almacenado para ser usado más tarde en el programa. Las variables vienen determinadas por:

- Un nombre, que permite al programa acceder al valor que contiene en memoria. Debe ser un identificador válido.
- Un tipo de dato, que especifica qué clase de información guarda la variable en esa zona de memoria
- Un rango de valores que puede admitir dicha variable.

Al nombre que le damos a la variable se le llama **identificador**. Los identificadores permiten nombrar los elementos que se están manejando en un programa. Vamos a ver con más detalle ciertos aspectos sobre los identificadores que debemos tener en cuenta.

Las variables se declaran de esta forma:

tipoVariable identificadorVariable;

Por ejemplo: **int edad;**

Esto indica que hemos reservado 4 bytes de memoria RAM (por ser de tipo "int" para poder guardar número enteros, Más tarde veremos los tipos de datos que tiene Java.

Para saber más

Bruce Eckel es el autor de los libros sobre Java y C++, dirigidos a programadores que desean aprender sobre estos lenguajes y sobre la programación orientada a objetos. Este escritor ha tenido la buena costumbre de editar sus libros para que puedan descargarse gratis. Así, podemos acceder de forma totalmente gratuita a la cuarta edición de su libro “Piensa en Java” en el siguiente enlace (en español):

Enlace al libro "Piensa en Java"

A partir de ahora es conveniente que utilices algún manual de apoyo para iniciarte a la programación en Java. Te proponemos el de la serie de Libros “Aprenda Informática como si estuviera en primero”, de la Escuela Superior de Ingenieros Industriales de San Sebastián (Universidad de Navarra):

Acceso al PDF "Aprenda Java como si estuviera en primero"

1.1 Identificadores.

Un **identificador** es el nombre que le da un programador a: variables, métodos, clases, ...

.

En Java es una secuencia ilimitada sin espacios de letras y dígitos Unicode, de forma que **el primer símbolo de la secuencia debe ser una letra, un símbolo de subrayado (_) o el símbolo dólar ($)**. Por ejemplo, son válidos los siguientes identificadores:

```
x5      αtη     NUM_MAX     numCuenta
```

Imagen extraída de curso Programación del MECD.

Ejemplo:

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

    int $;

    int $sueldo;

    Scanner scanner = new Scanner(System.in);
    System.out.print("\n\tInserte un número: ");
    $ = scanner.nextInt();
    System.out.println("\t\tEl valor insertado es " + $);
    System.out.print("\n\tInserta otro número: ");
    $sueldo = scanner.nextInt();
    System.out.println("\t\tEl último número insertado es " + $sueldo);

}
```

En la definición anterior decimos que un identificador es una secuencia ilimitada de caracteres Unicode. Pero… ¿qué es Unicode? Unicode es un código de caracteres o sistema de codificación, un alfabeto que recoge los caracteres de prácticamente todos los idiomas importantes del mundo. Las líneas de código en los programas se escriben usando ese conjunto de caracteres Unicode.

Esto quiere decir que en Java se pueden utilizar varios alfabetos como el Griego, Árabe o Japonés. De esta forma, los programas están más adaptados a los lenguajes e idiomas locales, por lo que son más significativos y fáciles de entender tanto para los programadores que escriben el código, como para los que posteriormente lo tienen que interpretar, para introducir alguna nueva funcionalidad o modificación en la aplicación.

Imagen extraída de curso Programación del MECD.

El estándar Unicode originalmente utilizaba 16 bits, pudiendo representar hasta 65.536 caracteres distintos, que es el resultado de elevar dos a la potencia dieciséis. Actualmente Unicode puede utilizar más o menos bits, dependiendo del formato que se utilice: **UTF-8**, **UTF-16** ó **UTF-32. A cada carácter le corresponde unívocamente un número entero perteneciente al intervalo de 0 a 2 elevado a n, siendo n el número de bits utilizados para representar los caracteres. Por ejemplo, la letra** ñ es el entero 164. Además, el código Unicode es "compatible" con el código ASCII, ya que para los caracteres del código ASCII, Unicode asigna como código los mismos 8 bits, a los que les añade a la izquierda otros 8 bits todos a cero. La conversión de un carácter ASCII a Unicode es inmediata.

Recomendación

Una buena práctica de programación es seleccionar nombres adecuados para las variables, eso ayuda a que el programa se autodocumente, y evita un número excesivo de comentarios para aclarar el código.

Para saber más

Enlace para acceder a la documentación sobre las distintas versiones de Unicode en la página web oficial del estándar:

Documentación sobre Unicode

2 Convenios y reglas para nombrar variables.

A la hora de nombrar un identificador existen una serie de normas de estilo de uso generalizado que, no siendo obligatorias, se usan en la mayor parte del código Java. Estas reglas para la nomenclatura de variables son las siguientes:

- **Java distingue las mayúsculas de las minúsculas**. Por ejemplo, Alumno y alumno son variables diferentes.
- **No se suelen utilizar identificadores que comiencen con «$» o «_»,** además el símbolo del dólar, por convenio, no se utiliza nunca.
- **No se puede utilizar el valor booleano** (true o false) **ni el valor nulo** (null).
- Los **identificadores deben ser lo más descriptivos posibles**. Es mejor usar palabras completas en vez de abreviaturas crípticas. Así nuestro código será más fácil de leer y comprender. En muchos casos también hará que nuestro código se autodocumente. Por ejemplo, si tenemos que darle el nombre a una variable que almacena los datos de un cliente sería recomendable que la misma se llamara algo así como ficheroClientes o manejadorCliente, y no algo poco descriptivo como Cl33.

Imagen extraída de curso Programación del MECD.

Además de estas restricciones, en la siguiente tabla puedes ver otras convenciones, que no siendo obligatorias, sí son recomendables a la hora de crear identificadores en Java.

Convenciones sobre identificadores en Java		
Identificador	**Convención**	**Ejemplo**
Nombre de variable.	Comienza por letra minúscula, y si tienen más de una palabra se colocan juntas y el resto comenzando por mayúsculas.	`numAlumnos, suma`
Nombre de constante.	En letras mayúsculas, separando las palabras con el guión bajo, por convenio el guión bajo no se utiliza en ningún otro sitio.	`TAM_MAX, PI`
Nombre de una clase.	Comienza por letra mayúscula.	`String, MiTipo`
Nombre de un método.	Comienza con letra minúscula.	`modifica_Valor, obtiene_Valor`

Autoevaluación

Un identificador es una secuencia de uno o más símbolos Unicode que cumple las siguientes condiciones. Señala la afirmación correcta.

- Todos los identificadores han de comenzar con una letra, el carácter subrayado (_) o el carácter dólar ($).
- No puede incluir el carácter espacio en blanco.
- Puede tener cualquier longitud, no hay tamaño máximo.
- Todas las anteriores son correctas.

3 Palabras reservadas.

Las palabras reservadas, a veces también llamadas palabras clave o keywords , son secuencias de caracteres formadas con letras ASCII cuyo uso se reserva al lenguaje y, por tanto, **no pueden utilizarse para crear identificadores.**

Imagen extraída de curso Programación del MECD.

Las palabras reservadas en Java son:

Palabras clave en Java				
`abstract`	`continue`	`for`	`new`	`switch`
`assert`	`default`	`goto`	`package`	`synchronized`
`boolean`	`do`	`if`	`private`	`this`
`break`	`double`	`implements`	`protected`	`throw`
`byte`	`else`	`import`	`public`	`throws`
`case`	`enum`	`instanceof`	`return`	`transient`
`catch`	`extends`	`int`	`short`	`try`
`char`	`final`	`interface`	`static`	`void`
`class`	`finally`	`long`	`strictfcode`	`volatile`
`const`	`float`	`native`	`super`	`while`

Hay palabras reservadas que no se utilizan en la actualidad, como es el caso de `const` y `goto`, que apenas se utilizan en la actual implementación del lenguaje Java. Por otro lado, puede haber otro tipo de palabras o texto en el lenguaje que aunque no sean palabras reservadas tampoco se pueden utilizar para crear identificadores. Es el caso de `true` y `false` que, aunque puedan parecer palabras reservadas, porque no se pueden utilizar para ningún otro uso en un programa, técnicamente son **literales booleanos**. Igualmente, null es considerado un literal, no una palabra reservada.

Normalmente, los editores y entornos integrados de desarrollo utilizan colores para diferenciar las palabras reservadas del resto del código, los comentarios, las constantes y literales, etc. De esta forma se facilita la lectura del programa y la detección de errores de sintaxis. Dependiendo de la configuración del entorno se utilizarán unos colores u otros, es posible que los que utilice tu versión de Eclipse o de Netbeans se parezcan a éstos:

Las palabras reservadas en granate en Eclipse y en azul en NetBeans
Los comentarios aparecen en verde en Eclipse y en gris en NetBeans
Las variables dentro de una clase aparecen en azul en Eclipse y en verde en NetBeans
Los errores en rojo
Y el resto del código aparece en negro

Puede que te interese cambiar los colores que utiliza Eclipse o Netbeans para la sintaxis de tus programas, por ejemplo si quieres que los comentarios aparezcan en verde en lugar de en gris, o indicar que tienes la autoría de los mismos, en lugar de que te aparezca el nombre de usuario del sistema operativo.

4 Tipos de variables I.

En un programa nos podemos encontrar distintos tipos de variables. Las diferencias entre una variable y otra dependerán de varios factores, por ejemplo, el tipo de datos que representan, si su valor cambia o no a lo largo de todo el programa, o cuál es el papel que llevan a cabo en el programa. De esta forma, el lenguaje de programación Java define los siguientes tipos de variables:

a. **Variables de tipos primitivos y variables referencia.** Variables de tipos primitivos son aquellas que son del tipo boolean, char, byte, short, int, long, float y double. Variables referencia son aquellas cuyo tipo es una clase. Son las que se llaman objetos.

b. **Variables y constantes**, dependiendo de si su valor cambia o no durante la ejecución del programa. La definición de cada tipo sería:

- Variables. Sirven para almacenar los datos durante la ejecución del programa, pueden estar formadas por cualquier tipo de dato primitivo o referencia. Su valor puede cambiar varias veces a lo largo de todo el programa.

o Constantes o variables finales. Son aquellas variables cuyo valor no cambia a lo largo de todo el programa.

c. **Variables miembro y variables locales**, en función del lugar donde aparezcan en el programa. La definición concreta sería:

- Variables miembro. Son las variables que se crean dentro de una clase, fuera de cualquier método. Pueden ser de tipos primitivos o referencias, variables o constantes. En un lenguaje puramente orientado a objetos como es Java, todo se basa en la utilización de objetos, los cuales se crean usando clases. En la siguiente unidad veremos los distintos tipos de variables miembro que se pueden usar.
- Variables locales. Son las variables que se crean y usan dentro de un método o, en general, dentro de cualquier bloque de código. La variable deja de existir cuando la ejecución del bloque de código o el método finaliza. Al igual que las variables miembro, las variables locales también pueden ser de tipos primitivos o referencias.

Imagen extraída de curso Programación del MECD.

Autoevaluación

Relaciona los tipos de variables con la característica correspondiente, escribiendo el número asociado a la característica en el hueco correspondiente.

Las variables... Relación Tienen la característica de que …

Locales.	☐	1. Una vez inicializadas su valor nunca cambia.
Miembro.	☐	2. Van dentro de un método.
Constantes.	☐	3. Van dentro de una clase.

Resolver

5 Tipos de variables II.

El siguiente ejemplo muestra el código para la creación de varios tipos de variables. Como ya veremos en apartados posteriores, las variables necesitan declararse, a veces dando un valor y otras con un valor por defecto. Este programa crea una clase que contiene las siguientes variables:

- **Variable constante llamada** `PI`: esta variable por haberla declarado como constante no podrá cambiar su valor a lo largo de todo el programa.
- **Variable miembro llamada** x**:** Esta variable pertenece a la clase `ejemplovariables`. La variable x puede almacenar valores del tipo primitivo `int`. El valor de esta variable podrá ser modificado en el programa, normalmente por medio de algún otro método que se cree en la clase.
- **Variable local llamada** `valorantiguo` **:** Esta variable es local porque está creada dentro del método `obtenerX()`. Sólo se podrá acceder a ella dentro del método donde está creada, ya que fuera de él no existe.

En apartados posteriores veremos como darle más funcionalidad a nuestros programas, mostrar algún resultado por pantalla, hacer operaciones, etc. Por el momento, si ejecutas el ejemplo anterior simplemente mostrará el mensaje "`GENERACIÓN CORRECTA`", indicando que todo ha ido bien y el programa ha finalizado.

```
/**
 * Aplicación ejemplo de tipos de variables

 *
 * @author FMA
 */
public class ejemplovariables {
    final double PI =3.1415926536;  // PI es una constante
    int x;                          // x es una variable miembro
                                    // de clase ejemplovariables

    int obtenerX(int x) {               // x es un parámetro
            int valorantiguo = this.x;  // valorantiguo es una variabe local
            return valorantiguo;
    }

    // el método main comienza la ejecución de la aplicación
    public static void main(String[] args) {
        // aquí iría el código de nuestra aplicación


    } // fin del método main

} // fin de la clase ejemplovariables
```

Aquí tienes el mismo código copiable:

/**

 * Aplicación ejemplo de tipos de variables

 *

 * **@author** FMA

```
 */
public class ejemplovariables {
   final double PI =3.1415926536;  // PI es una constante
   int x;                          // x es una variable miembro
                                   // de clase ejemplovariables

   int obtenerX(int x) {           // x es un parámetro
         int valorantiguo = this.x;  // valorantiguo es una variable local
         return valorantiguo;
   }

   // el método main comienza la ejecución de la aplicación
   public static void main(String[] args) {
      // aquí irá el código de nuestra aplicación


   } // fin del método main

} // fin de la clase ejemplovariables
```