Programación Interactiva
Tipos y Operadores

Escuela de Ingeniería de Sistemas y Computación
Facultad de Ingeniería
Universidad del Valle

Tópicos

- Tipos de Datos
- Variables
- Arreglos
- Comentarios en Java
- Operadores
- Instrucciones

Tipos de Datos

- Los tipos de datos especifican el tamaño y el tipo de valor que puede ser almacenado.

- Los tipos de datos en Java pueden ser clasificados como:

    - Tipos primitivos
    - Tipos de referencia

Tipos de Datos (Primitivos)

byte	entero de un byte	(+/-)
short	entero corto	(+/-)
int	Entero	(+/-)
long	entero largo	(+/-)
float	punto flotante, precisión sencilla	(+/-)
double	punto flotante, doble precisión	(+/-)
char	Carácter	
boolean	Boleana o lógica	

Variables

- Las variables son posiciones de memoria en donde se pueden guardar valores. A una variable se le identifica con un nombre, se le define un tipo de dato y almacena un valor.

- La forma básica para declarar una variable es:

  ***tipo* nombre1, nombre2;**

Variables

La declaración de una variable son tres cosas:

3. Le indica al compilador cual es el nombre de la variable.
4. Especifica que tipo de dato puede almacenar la variable.
5. La posición de la declaración (dentro del programa) decide el *alcance* de la variable.

Variables

Una variable se debe declarar e inicializar con un valor antes de utilizarse.

Para inicializar una variable se puede hacer de dos manera:

5. Utilizando un instrucción de asignación
6. Desde el teclado, utilizando el método:

System.in.readLine()

Variables

- **Alcance de una Variable (Scope of Variable)**
  El área del programa donde la variable es accesible se conoce como el alcance.

- Java utiliza tres clases de variables:
  - **de instancia** (*Instance* variables)
  - **de clase** (*Class* variables(global))
  - **locales** (*Local* variables)

Costantes

- Se utilizan para indicar un valor fijo o constante dentro del programa.

- Literales enteros (se asumen del tipo int o long)

  - 100 ó -123456 (por defecto, base 10)
  - 98765L (forzado a long)
  - 2E3 (notación científica)
  - 011 (0 al principio indica octal, base 8), 011 = 9
  - 0xAB (0x al principio indica hexadecimal, base 16), 0xAB = 171

Costantes

- Literales punto flotante (se asumen del tipo double)
  - 3.1416
  - -0.12345
  - 2.56F (forzado a float)
  - 7.8E-2 (notación científica)

Costantes

- Literales booleanos
  - true
  - false

- Literales carácter
  - ’a’
  - ’n’
  - '\’ (diagonal inversa)

Ejemplo

```
class tiposDatos
{
        public static void main(String args[])
        {
                byte ebyte = 100;
                short ecorto = 25000;
                int entero = 1000000;
                long elargo = 31234567890;
                float punflosencilla = 123.457;
                double punflodoble = 123.457;
                char caracter = 'x';
                boolean logica = false;
                ...
```

Ejemplo

```
...
System.out.println("Enteros");
System.out.println("tipo byte:"+ebyte);
System.out.println("tipo short:"+ecorto);
System.out.println("tipo integer:"+entero);
System.out.println("tipo long:"+elargo);
System.out.println("Punto Flotante");
System.out.println("tipo float:"+ punflosencilla);
System.out.println("tipo doublet:"+ punflodoble);
System.out.println("tipo chart:"+ caracter);
System.out.println("tipo booleant:"+ logica);
    }
}
```

Arreglos

- Un arreglo es una **lista** de variables del *mismo tipo* que se referencian con un *nombre común*.
- Un elemento o variable de un arreglo se accesa por su *posición o índice* dentro de la lista.
- Para crear y poder utilizar un arreglo se deben seguir tres pasos:
  - Declaración
  - Construcción
  - Inicialización

Arreglos

- **Declaración:**
  La declaración le indica al compilador el **nombre** del arreglo y cual es el **tipo** de sus elementos o variables.

- Hay dos formas para declarar un arreglo:
  - int miarreglo[];
  - float miarreglo[];
  - int[] miarreglo;
  - float[] miarreglo;

Arreglos

- Los paréntesis cuadrados pueden venir después del nombre o del tipo.
- El tamaño del arreglo o el número de elementos que puede guardar se define cuando se construye.

Arreglos

- **Construcción:**
  En el momento de construir o crear un arreglo se establece el número de elementos que almacena. Existen dos formas de construcción:

- Con **new**

  int arreglo[];

  arreglo = **new** int[5];
  int arreglo[] = **new** int[5]

- A la inicialización

  int arreglo[] = {1,2,3,4,5};
  int arreglo[] = {1,2,3,4,5};

Arreglos Multidimensionales

Es un arreglo de arreglos.

- **Definición:**

  int arreglo**[][]**;

  float**[][][]** arreglo;

- **Creación:**

  int arreglo = new int[4][5];

  int arreglo = new int[lineas][columnas];

- **Inicialización:**

  int arreglo[][] = {{1,2,3},{4,5,6},{7,8,9}};

Acceso a los elementos de un Arreglo

- Para obtener o modificar el valor de un elemento almacenado dentro de un arreglo se utiliza su *posición o índice* dentro del arreglo.

arreglo[5];
arreglo[pos];
arreglo[i-1];

Acceso a los elementos de un Arreglo

- El valor del índice se *inicia en 0* para el **primer** elemento y *termina en n-1* para el **n-simo** o **último** elemento, para un arreglo de 3 elementos.

arreglo[0] = 10; *(primer elementos)*
arreglo[1] = 20; *(segundo elementos)*
arreglo[2] = 50; *(último elemento)*

- El atributo length del objeto indica la cantidad de elementos que este puede contener: arreglo.length

Cadena de Caracteres Clase String

- Una cadena de caracteres es una combinación de caracteres concatenados. Son una instancia de la clase **String.**

- Un **literal** de cadena (de caracteres) es una serie de caracteres entrecomillas dobles:

  "esta es un literal de carácteres"

- Para introducir una " en medio de una cadena de caracteres, debe anteponérsele un \

Cadena de Caracteres String

- Para definir, crear e inicializar una variable cuyo tipo sea una cadena de caracteres:

  String cadena = *new* **String**(“abcdefgh 123”);

O simplemente:

  String cadena = “abcdefgh 123”;

O también:

  String cadena = “abc” + ”defgh ” +123

Una cadena vacía:

  String cadena = “”;

Comentarios

Java soporta tres estilos de comentarios:

- Línea Sencilla, comenzando con //

```
// Este comentario es de una linea
```

- Línea Múltiple, inicia con /* y finaliza con */

```
/* Un comentario
   de líneas múltiples */
```

- Para usar con el javadoc, en verdad son los mismos comentarios del tipo anterior, inician con /** y finalizan con */:

```
/**
* @param nombre nombre del empleado
* @return nombre nombre del empleado
*/
```

Operadores

- Un operador es un símbolo que le indica al compilador una cierta "manipulación" u "operación" aritmética o lógica, por lo general se utilizan en **expresiones**.
- Los valores involucrados se conocen como **operandos**.
- Una **expresión** es una combinación de operadores y operandos que regresa **un valor**.

```
(a + b) * (c + d)          //expresión aritmética
a – 3                      //expresión aritmética
(a > b) && (c <= d)        //expresión lógica
```

Operadores

- **El operador de Asignación =**
  Se asigna el valor resultante de una expresión a una variable. El tipo del valor resultante de la expresión debe ser compatible con el tipo de la variable.

  variable = expresión;

  j = 15;
  j = l + 15;
  a = b = c = d = 0;

- En una **asignación** se evalúa primero la expresión del lado **derecho** y luego se hace la **asignación**.

Operadores

- Existen unos operadores cortos de asignación:

x += y	*equivale a*	*x = x + y*
x -= y	*equivale a*	*x = x – y*
*x *= y*	*equivale a*	*x = x * y*
x /= y	*equivale a*	*x = x / y*
x %= y	*equivale a*	*x = x % y*

- En un **operador corto de asignación** se evalúa primero el operador del lado **izquierdo** y luego se hace **la asignación**.

Operadores Aritméticos

+	Usado para sumar
-	Usado para restar
*	Usado para multiplicar
/	Usado para dividir
%	Módulo para buscar el residuo de una división
+=	Operador de asignación que suma
-=	Operador de asignación que resta
*=	Operador de asignación que multiplica
/=	Operador de asignación que divide
%=	Operador de asignación que calcula el módulo
++	Operador de incremento por 1
--	Operador de decremento por 1

Operadores de Incremento y Decremento

++x;	es lo mismo que	x = x + 1;
--x;	es lo mismo que	x = x – 1;
y = ++x;	(incrementa x, luego lo asigna a y)	
y = --xx;	(decrementa x, luego lo asigna a y)	
y = x++;	(asigna el valor de x a y, luego incrementa x)	
y = x--;	(asigna el valor de x a y, luego decrementa x)	

- En una asignación se evalúa primero la expresión del lado derecho y luego se hace la asignación. En un operador corto se evalúa primero el operador del lado izquierdo.

Operadores de Comparación

Operador	Ejemplo	
<	menor	if (a<b)
<=	menor e igual	if (a<=b)
>	mayor	if (a>b)
>=	mayor e igual	if (a>=b)
==	igual	if (a==b)
!=	diferente	if (a!=b)

- El tipo del resultado de una operación de comparación es un boleano (true/false).

Operadores Lógicos y Boleanos

- Operan entre dos operandos del tipo boleano y se obtiene como resultado un valor boleano.

Operador	Resultado
!	No lógico (NOT)
& o &&	Y boleano o lógico (AND)
| o ||	O boleano o lógico (OR)
^	O exclusivo boleano (XOR)
==	igual (equal)
!=	no igual a (Not equal to)
&=	asignación Y (AND)
|=	asignación O (OR)
^=	asignación O exclusivo (XOR)

Operadores Lógicos y Boleanos

- Ejemplo:

a	b	a||b	a&&b	a^b	!a
true	*true*	*true*	*true*	*false*	*false*
true	*false*	*true*	*false*	*true*	*false*
false	*true*	*true*	*false*	*true*	*true*
false	*false*	*false*	*false*	*false*	*true*

Operadores de Bits (Bitwise)

Operador	Descripción
&	Y de bits (AND)
|	O de bits (OR)
^	O exclusivo de bits (XOR)
>>	corrimiento a la derecha (Right shift)
<<	corrimiento a la izquierda (Left shift)
>>>	corrimiento con rellenado de ceros
~	complemento
&=	Y de bits y asignación
|=	O de bits y asignación
^=	O exclusivo y asignación

Forzar o Convertir Tipo (casting)

- Permite la **conversión** del valor de un tipo de dato primitivo o de un objeto, en otro valor compatible.

```
int a;
int a=20,c;
float b = 1.25;
double d;
a = (int) b;
c = a/b;
d = (double)  (a/b);
```

Precedencia de Operadores

- Dentro de una expresión los operadores tienen un orden de precedencia u orden de evaluación que determina su valor, la siguiente es la tabla en orden de precedencia de mayor a menor.

Precedencia de Operadores

Precedencia	Operador	Descripción
1	(), []	prefijos y sufijos
2	++, --, ~, !	complemento,negación
3	new, (tipo)	creación de objetos y forzado
4	*, /, %	multiplicar, dividir, módulo
5	+, -	sumar, restar
6	<<, >>, >>>	corrimientos
7	<, <=, >, >=	Relacionales
8	== , !=	igualdad y desigualdad
9	&	Y de bits y boleano
10	^	O exclusivo de bits y boleano

Precedencia de Operadores

Precedencia	Operador	Descripción
11	|	O de bits y boleano
12	&&	Y lógico
13	||	O lógico
14	?:	Condicional
15	=, += , -=, *= , /= , %=, <<=, >>=, >>>=, &=, ^=, !=	asignación

“Tarea”

- Leer sobre:
  - Qué son las Excepciones en Java ?
  - Qué son Eventos en Java ?
  - Creación de interfaces gráficas en Java
  - Principales clases de Swing