Introducción a Java (II)

Dr. (c) Noé Alejandro Castro Sánchez

Sintaxis de Java

Sintaxis

- Define las reglas que determinar el uso de palabras clave, operadores y variables para construir y evaluar expresiones.
- La sintaxis rige el manejo de:
  - Comentarios
  - Identificadores
  - Palabras clave
  - Literales
  - Expresiones y operadores
  - Variables y tipos de datos
  - Bloques y sentencias

Comentarios

- Comentarios regulares en código fuente:
  - Varias líneas: `/*` y `*/` como en C y C++

```
/* Este es un ejemplo
   de varias líneas
*/
```

  - Una sola línea: se usa la doble diagonal. El comentario inicia desde ella y continua hasta el final de la línea.

```
// Comentario de una sola línea
```

- Comentarios para documentación: utilizado para generar documentarios utilizando *javadoc*
  - Se inicia con doble asterisco

```
/** inicio de comentario
para documentación */
```

Identificadores

- Secuencia de caracteres que nombra una variable, método, clase, interfaz o paquete.
- Características
  - No usar espacios en blanco
  - Puede incluir letras, dígitos, signo de pesos, guión bajo:
    ```
    contador5
    _apellidoPaterno
    saldo$
    ```
  - Sensibles a mayúsculas y minúsculas

Palabras reservadas

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

Literales

- Valor formado por una secuencia de caracteres
- Literales numéricas
  - `123 // Literal int`
- Literales booleanas
  - `true` o `false`
- Literales de caracteres
  - `‘a’, ‘&’, ‘7’`
- Literales de cadena
  - Las cadenas es una combinación de caracteres. Son instancias de la clase *String* (contienen métodos que permiten manipularlas)
    `“hola”, “cadena123”, “12345”`

Operadores y operandos

- Operadores: tipo especial de caracteres que identifican una operación de datos. Además, devuelve un valor.
- Operandos: datos que se conectan y se procesan por el operador.

```
3 + 2 * 5
```

- Características de operadores:
  - Transforman los operandos en un nuevo valor
  - Evalúan los operandos según un orden predefinido

Clasificación de operadores

- Según el número de operandos que requiera
  - Unario (e. g., negación)
  - Binario (e. g., suma, resta)
  - Ternario (único: operador condicional)

- Según la posición del operador respecto al operando
  - Prefijo (antes de; e. g., operador a bit de complemento)
  - Posfijo (después de; e. g., post-incremento)
  - Infijo (entre; e. g., suma)

Operadores

- Tipos de Operadores
  - Asignación
  - Aritméticos
  - Concatenación de cadenas
  - Índice de arreglos
  - Relacionales
  - Condicionales (ternario)
  - Lógicos
  - Manipulación de bits
  - Booleano
  - **Comprobación de tipos de dato referencia**

Tipos de operadores

- Asignación
  - El más simple. Asigna al operando izquierdo, el operando derecho:

```
int num = 27;
num = 32;
```

- Aritméticos
  - Suma, Resta, Multiplicación, Módulo, División, Pre incremento, Pos tincremento, Pre decremento, Post decremento, Resta unaria, Suma unaria

Operados aritméticos

- Suma (`operando1 + operando2`),
  resta (`operando1 - operando2`),
  multiplicación (`operando1 * operando2`),
  división (`operando1 / operando2`)
  - Juntar uno de los operadores previos y el de asignación para lograr la operación del operador con asignación en una sola operación:

```
suma += 2;   // suma = suma + 2;
talla -= 2;  // talla = talla - 2;
x *= 3.5;    // x = x * 3.5;
peso /= 15;  // peso = peso / 15;
```

Operadores aritméticos II

- Módulo (operando1 % operando2)
  - Divide el operando izquierdo entre el derecho y retorna el residuo.

```
numero = 17;
numero %= 10; // ¿valor de numero?
```

- Pre/Post incremento/decremento
  - Añade (incremento) o sustrae (decremento) el valor de `1` a su operando y regresa el valor
    - `++operando` `// preincremento`
    - `operando++` `// postincremento`
    - `--operando` `// predecremento`
    - `operando--` `// postdecremento`

Operadores aritméticos III

- ¿Cuál es el valor de *i* en cada impresión?

```
…
int i = 0;

System.out.println(i++);
System.out.println(i);
System.out.println(++i);
System.out.println(i);
…
```

- Resta/suma unaria
  - Resta: realiza una negación sobre su operando. Si éste es positivo, el resultado es negativo, y viceversa:

```
int i = -5;
System.out.println(-i); // 5
```

  - Suma: devuelve su operando.

Operador de concatenación de cadenas

- Operador binario que concatena el operando izquierdo con el operando derecho
  - *operando1* + *operando2*

```
System.out.println("Concantena " + "cadenas");
// Resultado: Concatena cadenas
```

- Puede concatenar datos primitivos a cadenas
  - `int edad = 22;`

```
System.out.println("Edad: " + edad);
// Resultado: Edad: 22
```

Ejercicio II

- Modelar la clase *televisor* (usar diagramas de clases)
  - Identificar atributos
  - Identificar métodos
- Crear clase *PruebaTelevisor*
  - Declarar método main()
  - Crear un objeto de tipo *televisor*
  - Disminuir 3 canales
  - Aumentar volumen
  - Por cada modificación de canal/volumen, debe imprimir el nuevo valor: *Canal 7*

Operadores relacionales

- Proveen la base para tomar una decisión
- Comparan los valores de sus operandos numéricos

==	Igualdad	a == b
!=	Distinto	a != b
<	Menor que	a < b
>	Mayor que	a > b
<=	Menor o igual que	a <= b
>=	Mayor o igual que	a >= b

Operador condicional (ternario)

- Elige uno de dos operandos según el resultado de la evaluación de una expresión

```
expresión ? operando1 : operando2;
```

  - Si la expresión es verdadera, se retorna operando1, sino operando2:

```
boolean b = true;
int x = (b == true) ? 1 : 0;
```

  - Es equivalente a:

```
if (b == true)
   x = 1;
else
   x = 0;
```

Operadores lógicos

- AND (&&)
  - Operador binario que produce un resultado booleano. Sus operandos producen también valores lógicos. El resultado será *true*, si ambos operandos son verdaderos. Si uno o ambos son falsos, el resultado será *false*.

```
int dia = 7, hora = 9;
boolean seguir_durmiendo = false;

if  (dia == 7 && hora < 8)
   seguir_durmiendo = true;
else
   seguir_durmiendo = false;
```

Operadores lógicos (II)

- OR (||)
  - Operador binario que produce un resultado booleano. Sus operandos producen también valores lógicos. El resultado será *true*, si al menos uno de los operandos es verdadero. Si ambos son falsos, el resultado será *false*.

```
int edad = 51;
boolean descuento = false;

if (edad < 18 || edad > 50)
   descuento = true;
```

Operadores lógicos (III)

- NOT (!)
  - Operador unario también conocido como negación lógica. El operador cambia el valor booleano de su operando.

```
boolean valor1 = false, valor2 = !valor1;

if (valor2 == true)
   System.out.println(“verdadero”);
```

Operador de índice de arreglo

- Operador unario que permite acceder a la posición de un arreglo denotada por el operando, asignando a esa posición un valor o regresando el valor contenido en ella

```
nombre_arreglo[operando]

num = mi_arreglo[i]; // regresando valor
mi_arreglo[i] = num; // asignando valor
```

Precedencia de operadores

- Al descender se pierde precedencia. Operadores en la misma fila, tienen la misma precedencia.

```
10 / 5 * 2;
```

Operator
[] . (params) E++ E--
unary operators: -E !E ~E ++E --E
new (type)E
* / %
+ -
>> << >>>
< > <= >=
== !=
&
^
|
&&
||
? :
= += -= *= /= %= >>= <<= &= ^= |=

Variables

- Nombre que se asocia a una porción de memoria, que almacena algún dato.
- Java tiene tres tipos de variables:
  - De instancia (o datos miembro). Define los atributos de un objeto.
  - De clase. Similares a las variables de instancia, con la excepción de que los valores que guardan son los mismos para todos los objetos de una determinada clase.
  - Locales. Se utilizan dentro de los métodos.

Variables (II)

```
class Circulo
{
   static final double PI=3.1416; // De clase
   double radio; // De instancia

   //...

   double calcularArea()
   {
       double area=PI*radio*radio; // Local
       return area;
   }
}
```

Radiografía de los Métodos

```
<mod>* <tipo_retorno> <nombre> ([<args>*])
{
    <instrucciones>*
}
```

<nombre> cualquier identificador legal
<mod> modificador: (opcional) public, private, static, …
<tipo_retorno> dato primitivo (void si no devuelve nada)
<argumentos> lista de argumentos válidos separados por comas
(tipo_dato identificador)
<instrucciones> conjunto de código

…

```
void set_radio(int r)
{
    radio = r;
}
```

```
// Invocación
Circulo cir1 = new Circulo();
cir1.agrega_radio(25);
```

Ejercicio III

- Generar la clase de perros y crear un objeto a través del cual se describa un perro en particular.
  - El constructor recibe la raza del perro
- Crear métodos para:
  - Poner nombre y edad
  - Devolver datos antes indicados

Ejercicio IV

- Generar una clase para las memorias USB
- Implementar métodos:
  - grabar_informacion: argumento cantidad de Gigas a almacenar
    Devuelve true si pudo grabar, false:
    - no hay espacio suficiente
    - protegido contra lectura/escritura
  - borrar_informacion: argumento cantidad de Gigas a borrar. Devuelve true si pudo borrar, false si está protegido
  - espacio_disponible: devolverá cantidad de Gigas disponibles.
  - proteccion: argumento booleano indicando si se protege o no.

Ejercicio IV (segunda parte)

- El constructor recibirá la capacidad y marca de la memoria.
- Implementar una segunda clase donde se pueda interactuar con el objeto:
  - Grabar más del espacio disponible
  - Activar protección e intentar eliminar información
  - Solicitar espacio disponible
  - Etc.

Ejercicio IV (tercera parte)

- Cada operación que se realice, debe estar acompañada por una impresión en pantalla que la detalle:
  - Grabar 8 gigas:
    "Se almacenaron 8 Gigas de información"
  - Imprimir espacio disponible
    "Existen 8 Gigas de espacio disponible"
- Cada operación se debe validar:
  - Activar protección:
    "Protección activada"
  - Grabar información
    "No pudo completarse la operación"

Clase String

- Representación de cadenas:
  - C: secuencia de caracteres terminados con el carácter nulo
  - Java: objetos de la clase String