Universidad Diego Portales

Apunte Clase 1: Programación en Java

Yerko Ortiz
2024

Tabla de contenidos

1 Introducción

Durante el transcurso del curso CIT-2006, se hará uso del lenguaje de programación Java, el cual es ampliamente utilizado en la industria y academia. Java es un lenguaje imperativo, orientado a objetos con una sintaxis que recuerda en gran medida a su predecesor C. La concepción del lenguaje fue en la ya extinta Sun Microsystems, donde el computín James Gosling dirigió y lideró el proyecto para que el lenguaje viera la luz del día.

Las razones para utilizar Java durante el curso son:

1. Podrá aprender un lenguaje de programación nuevo.
2. Es un lenguaje ampliamente utilizado en la industria y academia.
3. Existen muchas implementaciones de los algoritmos a estudiar en Java, además de literatura que puede complementar el estudio.
4. Podrá integrar los algoritmos que implemente a alguna aplicación.

Se debe recalcar que para estudiar algoritmos no es realmente necesario programar en una computadora, al final del día los algoritmos que se estudiarán son ideas abstractas que se pueden computar, donde una idea computable perfectamente se puede definir o calcular con lápiz y papel. Pero también hay que admitir que experimentar con algoritmos de primera mano interactuando con un computador es parte de la diversión, por lo que Java será el lenguaje de programación que lo acompañará durante el transcurso del semestre.

1.1 Instalación

Su primera tarea es instalar Java en su computadora, para esto tiene un par de opciones:

- Instalar directamente algún JDK y luego programar con algún editor de texto como VSCode.
- Instalar algún IDE como IntelliJ. Para el caso de la Universidad recuerde que existe un beneficio con Jetbrains para estudiantes.

En resumen instale la opción que le agrade más, no hay ninguna mejor que otra en el contexto del curso, como así mismo no hay restricciones con el sistema operativo, siéntase libre de usar Windows, Linux o MacOs.

1.2 Hola Mundo

Ya es tradición imprimir hola mundo en cada lenguaje de programación que toca aprender, esta vez no será la excepción, para verificar que su ambiente de programación es capaz de compilar y ejecutar programas en Java implemente el siguiente programa:

```
1 public class HelloWorld {
2     public static void main(String[] args) {
3         System.out.println("Hello, world!");
4     }
5 }
```

Para ejecutar el programa puede escribir en su terminal el comando:

```
java HelloWorld.java
```

Se pueden realizar las siguientes observaciones del programa:

1. Existe un método llamado main, el cual es invocado por la maquina virtual de Java (JVM) para ejecutar el programa.
2. El método main es parte de la clase HelloWorld, esto algo propio de Java, en este lenguaje todo programa es parte de una clase.
3. El método main tiene tres palabras clave en su definición:
    a. static: esta keyword permite que el método main sea invocado sin necesidad de instanciar un objeto de la clase HelloWorld. Básicamente el método puede ser invocado directamente como si la clase que lo contiene fuese una librería.
    b. public: esta keyword denota que el método main puede ser accedido desde otras clases distintas a HelloWorld.
    c. void: esta keyword denota que el método main no retornará ningún valor al terminar su ejecución.
4. El método main tiene definido como paramétros de entrada un arreglo de strings llamado args, este argumento permite al usuario ingresar comandos para el programa vía CLI (command line interface).
5. Para imprimir "hello world" el programa hace uso del método println que es parte un objeto llamado System, donde out hace mención al stdout (standar output stream). System.out disponibiliza una serie de métodos para imprimir.

2 Tipos de Datos Primitivos

En Java los tipos de datos primitivos son los siguientes:

1. **Números enteros**: los números enteros en Java son representados con una codificación de complemento a dos.
    a. byte, 8 bit
    b. short, 16 bit
    c. int, 32 bit
    d. long, 64 bit
2. **Números racionales**: los números racionales en Java son representados con el estándar IEEE 754 para números con punto flotante. Esta representación permite realizar aproximaciones bastante precisas cuando se trabaja con números reales o complejos.
    a. float, 32 bit
    b. double, 64 bit
3. **Caracteres**: los caracteres en Java son representados internamente con Unicode, lo cual permite trabajar con codificaciones ASCII, UTF8 y otras.
    a. char, 16 bit
4. **Booleanos**: los tipos de datos booleanos representan una afirmación lógica, un boolean en Java puede tener valor true o false. Este tipo de dato no tiene un tamaño de memoria fijo, la especificación dice que utiliza un número de bits indeterminado.
    a. boolean, true or false

2.1 Expresiones

En Java una expresión es una combinación de literales, variables, operadores, invocaciones a métodos y otras expresiones que resultan en un valor del mismo tipo de dato de sus operandos.

Para experimentar esta idea de primera mano, implemente un programa que imprima las siguientes expresiones:

```
5 * (int) 5.8
3.4 + 5.6
5.0 + 2e-3f
'a' + 10
!true
true && false
```

2.2 Operadores

Para formar expresiones es posible utilizar operadores asociados a algún tipo de dato.

1. Números enteros (byte, short, int, long):

```
Suma +
Resta  -
Multiplicación  *
División  /
Resto  %
Incremento/Decremento por prefijo ++i/--i
Incremento/Decremento por postfijo i++/i--
```

2. Números racionales(float, double):

```
Suma +
Resta  -
Multiplicación  *
División  /
```

3. Booleanos (boolean):

```
and &&
or ||
not !
xor ^
```

4. Caracteres (char): Es posible realizar operaciones aritméticas con caracteres, pero estas son realizadas en casos muy específicos cuando se trabaja con alguna codificación bien conocida como ASCII.

5. Operadores de comparación (aplicables a cualquier tipo de dato primitivo)

```
Igual =
Menor o igual <=
Mayor o igual >=
Mayor >
Menor <
```

3 Statements

Un statement es una instrucción que altera el flujo de control de un programa computacional o tiene un efecto secundario en el mismo. En español la palabra que más se acerca a lo que un statement de programación representa es sentencia, no obstante para evitar ambigüedades durante el curso nos referiremos a este grupo de instrucciones como statements.

3.1 Declaración de variables

Java es un lenguaje fuertemente tipado, esto quiere decir que toda variable debe estar asociada a un tipo de dato. Para declarar una variable es posible anteponer el tipo de dato y luego asignarle un identificador (nombre).

```
int x;

double variable;

char caracter;
```

3.2 Asignación

Con el operador = es posible asignar un valor a una variable, siempre y cuando dicho valor sea del mismo tipo que la variable.

```
int x;
x = 10;
```

Es posible escribir la declaración y asignación en una misma linea:

```
int x = 10;
```

Es posible combinar operadores aritméticos con asignación para operar una variable con una expresión en particular:

```
int x = 20;
x \= 5;
```

3.3 Condicionales

Para controlar el flujo del programa cuando hay múltiples opciones es posible hacer uso de los statements if-else o switch-case.

3.3.1 If Else

El statement if-else tiene la siguiente estructura:

```
if (<boolean-expr>) {
    <body>
} else if (<boolean-expr>) {
    <body>
} else {
    <body>
}
```

Es posible evaluar una expresión booleana que en caso de ser cierta permiten ejecutar el código en el body asociado de la expresión. En caso que ninguna expresión sea cierta, es posible especificar una serie de instrucciones en la última evaluación (else).

El siguiente statement imprime true si un número es par y false en caso que no lo sea.

```
if (x % 2 == 0) {
    System.out.println(true);
} else {
    System.out.println(false);
}
```

3.3.2 Switch case

El statement switch-case permite ejecutar una acción específica de acuerdo a el valor de una expresión. El caso a ejecutarse depende exclusivamente si existe un match entre la expresión evaluada en el switch y la expresión del case. En caso de no existir match es posible tener una acción por default. Las expresiones a evaluar deben ser siempre del mismo tipo.

```
switch(<expr>) {
    case <expr1>:
        <body>
        break;
    case <expr2>:
        <body>
        break;
    default:
        <body>
        break;
}
```

3.4 Ciclos

Para controlar el flujo del programa y ejecutar instrucciones de manera repetida de acuerdo a alguna condición es posible hacer uso los statements for y while.

3.4.1 While

El statement while permite ejecutar repetidamente un conjunto de instrucciones mientras la expresión que evalua sea verdadera. En general el statement while tiene la siguiente estructura:

```
while(<boolean-expr>) {
    <body>
}
```

3.4.2 For

El ciclo for por su lado permite inicializar alguna variable, tener una expresión booleana que decide si el ciclo continua o no, además de alguna expresión que incremente la variable inicializada y permita que el ciclo llegue a término en algún momento.

```
for(<initialization>; <boolen-expr>; <increment>) {
    <body>
}
```

3.4.3 Break and Continue

Existen dos keywords que pueden resultar útiles en algún momento:

1. break: esta keyword interrumpe un ciclo, terminándolo en el acto.
2. continue: esta keyword interrumpe un ciclo, haciendo que este pase a la siguiente iteración de manera inmediata.

4 Métodos Estáticos

Un método encapsula un conjunto de instrucciones, esto permite abstraer operaciones complejas en un mismo método y disminuir el código duplicado. Un método recibe valores de algún tipo de dato como argumento y retorna un valor o produce un efecto secundario en el programa que invoca el método. Todo método estático está pensado para ser invocado sin tener que instanciar el objeto de la clase que lo define, por lo que estos suelen tener la siguiente estructura:

```
public static <data-type> <method-name> (<type1> <arg1>, ..., <typen> <argn>) {
    <body>
    return <expr>
}
```

Un ejemplo podría ser un método que calcula la suma de dos números. Este recibe como argumentos dos enteros y retorna un entero como salida.

```
public static int sum(int x, int y) {
    return x + y;
}
```

5 Arreglos

Un arreglo en Java es definido por la clase Array. Un arreglo contiene una secuencia de valores del mismo tipo.

1. Declarar e instanciar un arreglo

```
int[] a = new int[n];
```

2. Iterar

```
int[] a = new int[n];
for(int i = 0; i < a.length; i++) {
    a[i] = 0
}
```

3. Asignar un valor a una posición

```
a[i] = 10;
```

4. Evaluar el valor en una posición

```
if (a[5] == 10) {
    System.out.println("ten");
}
```