Inicio en la programación en Java

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

9 de septiembre de 2022

Organización

Temas

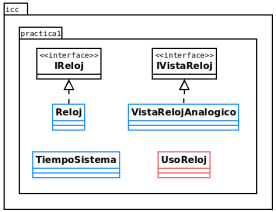
Java

- `Java` es un lenguaje de programación orientado a objetos, aunque aún contiene elementos fuera de ese paradigma.
- Fue diseñado para implementar grandes sistemas de software que funcionaran a través de la red.

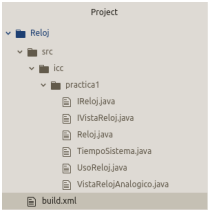
Paquetes y clases

La organización de un proyecto en `Java` se refleja en dos niveles:

En su arquitectura lógica, en forma de *paquetes* y *clases*.



En la organización del código, acomodado en **directorios** y **archivos**.



- A partir de la versión 9, también se incluyen *módulos*[1].

[1] Información en: `http://openjdk.java.net/projects/jigsaw/quick-start`

Temas

Clases de uso

- Para poder reutilizar código entre diferentes proyectos, las funcionalidades provistas por el sistema se implementan dentro de *clases*, plantillas a partir de las cuales se general objetos con datos y sus comportamientos.
- Por otro lado, un programa concreto comenzará su ejecución a partir de una *clase de uso*, que manda llamar, conforme se requiera, las funcionalidades provistas por objetos de clases independientes.
- A semejanza del lenguaje estructurado C, las clases de uso se caracterizan por poseer un método `main` a partir del cual **inicia la ejecución del programa**.

Ejemplo: clase de uso

```
1  package mate.numeros;
2
3  /** Programa para sumar dos números complejos. */
4  public class UsoComplejos {
5
6    /** Punto de entrada. */
7    public static void main(String[] args) {
8      Complejo c1 = new Complejo(4, 5);
9      Complejo c2 = new Complejo(-4, 3);
10     Complejo r = c1.suma(c2);
11     System.out.println("Suma␣" + c1 + "+" + c2 + "=" + r);
12   }
13
14 }
```

Ejemplo

- Las clases de uso también permiten la interacción con el usuario.
- Su nombre no inicia necesariamente con la palabra `Uso`, pero puede servir como indicación del rol de la clase.

```
1  package mate.numeros;
2
3  /** Programa para sumar dos números enteros. */
4  public class Suma {
5    /** @param args argumentos con que fue invocado el programa. */
6    public static void main(String[] args) {
7      int n1 = Integer.parseInt(args[0]);
8      int n2 = Integer.parseInt(args[1]);
9      int r = n1 + n2;
10     System.out.println(args[0] + " + " + args[1] + " = " + r);
11   }
12 }
```

Esto se ejecuta así:

```
$ java mate.numeros.Suma 4 3
4 + 3 = 7
$ java mate.numeros.Suma 353 -23
353 + -23 = 330
```

Tipos

Temas

Tipo de datos

- Un *tipo de datos* especifica la interpretación semática de la información introducida en la computadora, es decir, su significado y, por ende, cómo operar con ella.

Tipos abstracto de datos (TAD)

- Un *tipo abstracto de datos* es una especificación formal algebráica de:
  - Un conjunto de datos
  - las operaciones que pueden realizarse con ellos con:
    1. Las *precondiciones* que se deben cumplir para poder realizar la operación.
    2. Las *postcondiciones* o relaciones que se satisfarán tras haber ejecutado la operación.
- No indica cómo serán representados estos datos en ningún sistema en particular, ni los detalles de cómo se llevarán a cabo las operaciones. Por ello son **abstractos**.

Estructuras de datos

"Las estructuras de datos son las formas de representación interna de datos de la computadora, mediante las que se representa cualquier situación en la computadora, es decir, son los tipos de datos que maneja la máquina.

Por ejemplo, podemos representar a un trabajador mediante los datos nombre del empleado, número de horas trabajadas, cuota por hora, etcétera."

López Román 2011

Temas

Variables

- Siendo un lenguaje **imperativo**, `Java` hace uso de **variables** y **asignación de valores**.
- Las *variables* son localidades de memoria donde se pueden almacenar datos.
- Estos datos también pueden ser modificados a lo largo de la ejecución del programa.

Variables

- Es posible nombrar y referirse a una variable por medio de una cadena de caracteres.

```
1 num = 2.19;
```

- Algunos lenguajes de programación, como Java, son muy estrictos con respecto al tipo de datos que pueden ser almacenados en una variable, pues esto se utilizará para verificar la semántica de la operaciones sobre ellos.

```
1 float num = 2.19f;
```

Otros lenguajes sólo requieren el nombre.

Declaración y definición

Para lenguajes con manejo explícito de tipos, podemos distinguir dos pasos para la creación de variables:

Declaración Es el momento en el cual se aparta la localidad de memoria, se indica el tipo de dato que será almacenado ahí y se le asigna un nombre.

Sintaxis (Variable)

<declaración de variable> ::= <tipo> <identificador>,...,<identificador>;
<tipo> ::= <tipo primitivo> | <identificador de clase>
<identificador> ::= (<letra>|_)(<letra>|<dígito>|_)*

```
1 short edad;
2 String nombre, dirección;
3 Color _colorOjos;
4 double _num;
```

Definición Es el momento en el cual se asigna valor a la variable por primera vez.

Sintaxis (Asignación)

<asignación> ::= <identificador> = <valor>;

```
1 edad = 10;
2 nombre = "Mónica";
```

Declaración y definición Es posible abreviar realizando ambas acciones en un sólo enunciado.

```
1 short edad = 10;
2 String nombre = "Mónica";
3 String nombre = "Mónica", dirección = "Alameda␣25";
```

Temas

Tipos primitivos

- Una de las características *no orientadas a objetos* de `Java` es la presencia de tipos primitivos.
- Los tipos primitivos se encuentran cercanamente relacionados con la representación binaria de la máquina, lo cual permite operar con ellos eficientemente.
- Usualmente sus operaciones tienen asociados operadores fáciles de identificar.

Por ejemplo números: $5 + 5$, $6.78 - 8.9$, $10000/5$, $2\hat{}3$, etc.

Tipos primitivos en Java

Tabla: Tipos primitivos

Identificador	**Capacidad**	**Representación en memoria**
boolean	2 bytes	**true** o **false**
char	2 bytes	16 bits, Unicode 2.0
byte	1 byte	8 bits con signo en complemento a 2
short	2 bytes	16 bits con signo en complemento a 2
int	4 bytes	32 bits con signo en complemento a 2
long	8 bytes	64 bits con signo en complemento a 2
float	4 bytes	32 bits de acuerdo al estándar IEEE 754-1985
double	8 bytes	64 bits de acuerdo al estándar IEEE 754-1985

Operadores y precedencia I

Operadores. La primer columna indica la precedencia, entre mayor es el número, primero se realiza esa operación.

	Operandos	**Operador**	**Tipo**	**Asociatividad**
15	postfijo unario	() [] .	Paréntesis Índice de arreglo Selector de miembro	izq a der
14	postfijo unario	++ ——	Postincremento Postdecremento	der a izq

Operadores y precedencia II

13	prefijo unario	++	Preincremento	der a izqu
		−−	Predecremento	
		+	Más	
		-	Menos	
		!	Negación lógica	
		~	Complemento en bits	
		(tipo)	Conversión de tipo	
12	binario infijo	*	Multiplicación	izq a der
		/	División	
		%	Módulo	
11	binario infijo	+	Suma	izq a der
		-	Resta	

Operadores y precedencia III

10	binario infijo	<<	corrimiento de bits a la izquierda	izq a der
		>>	corrimiento de bits a la derecha con extensión de signo	
		>>>	corrimiento de bits a la derecha con llenando con ceros	
9	binario infijo	<	relacional menor que	izq a der
		<=	relacional menor o igual	
		>	relacional mayor que	
		>=	relacional mayor o igual	
		`instanceof`	comparación de tipos (sólo objetos)	

Operadores y precedencia IV

8	binario infijo	== !=	relacional igual a relacional distinto de	izq a der
7	binario infijo	&	AND de bits	izq a der
6	binario infijo	^	OR exclusivo de bits	izq a der
5	binario infijo	|	OR inclusivo de bits	izq a der
4	binario infijo	&&	AND lógico	izq a der
3	binario infijo	||	OR lógico	izq a der
2	ternario infijo	? :	Condicional ternario	izq a der

Operadores y precedencia V

1	binario infijo	= += −= ∗= /= %=	Asignación Autosuma y asignación Autorresta y asignación Automultiplicación y asignación Autodivisión y asignación Automódulo y asignación	der a izq
		&=		
		^=		
		|=		
		<<=		
		>>=		

Operadores y precedencia VI

		>>>=		

Fuente: http://www.cs.bilkent.edu.tr/~guvenir/courses/CS101/op_precedence.html y Valdés y Gurovich 2008.

Temas

Clases e Interfaces

- Diferentes lenguajes proveen al programador de mecanismos para definir sus propios tipos.
- Las interfaces de programación para aplicaciones (APIs) ya incluyen varios tipos definidos de esta manera.
- En `Java` hay dos formas de definir tipos:
  1. Clases
  2. Interfaces

  En este caso, las operaciones correspondientes toman la forma de *métodos*, en lugar de los operadores, que usan los tipos primitivos.

```
1 String hola = "Hola␣a␣todos";
2 String todos = hola.substring(7);
```

Bibliografía

Bibliografía I

- López Román, Leobardo (2011). *Programación estructurada y orientada a objetos. Un enfoque algorítmico*. 3.ª ed. Alfaomega.
- Oracle (3 de oct. de 2020a). *JDK 15 Documentation*. URL: https://docs.oracle.com/en/java/javase/15/.
- — (3 de oct. de 2020b). *OpenJDK*. URL: https://openjdk.java.net/.
- Valdés, Canek Peláez y Elisa Viso Gurovich (1 de mar. de 2008). *Introducción a las Ciencias de la Computación, Manual de Prácticas*.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual