Estructura de un programa Java

1. El lenguaje Java

1.1. ¿Qué es Java?

Java es un lenguaje desarrollado inicialmente Sun, utilizado por Netscape y luego como base para Javascript. Es utilizado para desarrollar aplicaciones para la web como aplicaciones locales, en Intranet y en Internet.

Java es un lenguaje:

- **de objetos** Está conceptualizado para trabajar con objetos
- **independiente de la plataforma** Al trabajar sobre la JVM (Java Virtual Machine), una forma de trabajo que corre sobre cualquier sistema operativo (Linux, Unix, Mac OS, Windows, etc.) permite el desarrollo de aplicaciones independiente de la máquina sobre la que se ejecutará

Características de Java:

- **Robusto** Difícilmente se verá afectado por carga de trabajo
- **Gestión de memoria** Hace uso óptimo de memoria, por lo que puede ser ejecutado en máquinas con memoria estándar
- **No permite el uso de técnicas de programación inadecuadas** Es celoso con la sintaxis, permitiendo una programación ordenada y eficiente
- **Multithreading** Multi-hilo, lo que le permite ejecutar varios procesos de manera paralela
- **Cliente/Servidor** Ideal para aplicaciones en las que uno o más clientes deben interactuar con un servidor
- **Mecanismos de seguridad incorporados** Java permite que las aplicaciones cuenten con niveles de seguridad para la protección de datos al momento de su ejecución
- **Herramientas de documentación incorporadas** Permite documentar una aplicación conforme se va desarrollando, facilitando así la labor del programador.

1.2. El lenguaje de objetos

Java es un lenguaje de objetos debido a que fue creado para trabajar con objetos, todo el ambiente Java es un ambiente de objetos. Otros lenguajes como C++, que es derivado del lenguaje estructurado C, tuvo que adaptarse para trabajar con objetos.

¿Qué es un objeto?

Un objeto es una entidad de software que cumple con las características de encapsulamiento y herencia.

Encapsulamiento de un objeto es su característica de auto-contenido, o sea que la misma unidad de objeto incluye los datos que éste usa (atributos) como los procedimientos (métodos) que actúan sobre los mismos.

Cuando se hace uso de la programación orientada a objetos, se definen clases (las clases definen objetos genéricos) y la forma en que los objetos interactúan entre ellos, a través de mensajes. Al crear un objeto de una clase dada, se dice que se crea una instancia de la clase, o un objeto propiamente dicho. Por ejemplo, una clase podría ser **autos**, y un auto dado (un automóvil tipo sedan) es una instancia de la clase.

La ventaja de esto es que como no hay programas que actúen modificando al objeto, éste se mantiene en cierto modo independiente del resto de la aplicación. Si es necesario modificar el objeto (por ejemplo, para darle más capacidades), esto se puede hacer sin tocar el resto de la aplicación, lo que ahorra mucho tiempo de desarrollo y depuración (*debugging*).

En Java, inclusive, ni siquiera existen las variables globales, (aunque parezca difícil de aceptar), lo que es una gran ventaja desde el punto de vista del desarrollo.

¿Qué es herencia?

Herencia significa que se pueden crear nuevas clases que **hereden** de otras preexistentes; esto simplifica la programación, porque las clases hijas incorporan automáticamente los métodos de las madres. Por ejemplo, nuestra clase **auto** podría heredar de otra más general, **vehículo**, y simplemente redefinir los métodos para el caso particular de los automóviles, lo que significa que, con una buena biblioteca de clases, se puede reutilizar mucho código inclusive sin saber lo que tiene adentro.

2. El programa de computadora

Un programa de computadora contiene en un orden lógico las instrucciones que el computador debe ejecutar para lograr un propósito claramente definido por el programador, ya sea para calcular un valor o para resolver un problema.

Martínez Ladrón de Guevara (2011) menciona que un programa describe cómo un ordenador debe interpretar las órdenes del programador para que ejecute y realice las instrucciones dadas tal como están escritas. Un programador utiliza los elementos que ofrece un lenguaje de programación para diseñar programas que resuelvan problemas concretos o realicen acciones bien definidas.

El siguiente programa Java es un clásico que muestra en la terminal el mensaje: "Hola Mundo"

```
/*
Este programa escribe el mensaje "Hola Mundo"
en la terminal del computador
*/
```

```
public class HolaMundo {
     public static void main (String[] args) {
          System.out.println("Hola Mundo");
     }
}
```

A continuación comentamos este programa:

```
/* ... */
```

Los comentarios son ignorados por el compilador y solo son útiles para el programador. Los comentarios ayudan a explicar aspectos relevantes de un programa y lo hacen más legible. En un comentario se puede escribir todo lo que se desee, el texto puede ser de una o más líneas.

Comentario, todo el texto que va entre los caracteres `/*` y `*/` no son tomados en cuenta por el computador, pero si por el programador, ya que pueden contener mensajes informativos como, el propósito del programa ("**Este programa calcula...**"), la fecha ("**Programa escrito en Piura, en junio del 2022**"), o si es parte de otro programa más amplio ("**Este programa cuenta el número de las consultas que hace un cliente dentro del sistema financiero del ....**")

```
public class HolaMundo { }
```

Definición de clase. La definición de la clase comienza por el carácter { y termina con el carácter }. El nombre de la clase lo define el programador.

Nombre de la clase y sus atributos. "**class HolaMundo**" es el nombre de la clase y es también el nombre del archivo que contiene nuestra aplicación (**HolaMundo.java**).

Hay que tener en cuenta el uso de los caracteres mayúscula y minúscula.

"**public**" denota que nuestra clase es de carácter público, es decir, que dentro de nuestra computadora puede ser accedida por otra aplicación que se ejecute dentro de ella.

Los caracteres **{ ... }** denotan el ámbito de nuestra clase, es decir, dónde empieza { y donde acaba } y contiene el método o los métodos que serán utilizados en nuestra aplicación.

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

Definición de método. Después de la definición de clase se escribe la definición del método **main()**. Todos los programas Java deben incluir un método main(). Este método indica las sentencias a realizar cuando se ejecuta un programa. Un método es una secuencia de sentencias ejecutables. Las sentencias de un método quedan delimitadas por los caracteres y que indican el inicio y el fin del método, respectivamente.

Nombre del método y sus atributos. **main** es el nombre del método principal (**main** en inglés significa **principal**).

Este método debe estar siempre presente en nuestra aplicación. Si nuestra aplicación requiere de varios métodos, sólo debe haber un método **main** y los demás pueden tener los nombres que el programador elija para ellos.

void indica que el método, main en este caso, no devolverá resultado alguno al sistema operativo.

static indica que las instrucciones y datos que conforman el método no son influenciados por eventos fuera de él.

public, indica que el método puede ser utilizado por una clase externa, en este caso, por una clase distinta a HolaMundo.

```
System.out.println("Hola Mundo");
```

Sentencia. Dentro del método **main()** se incluye una sentencia para mostrar un texto por la consola. Los textos siempre se escriben entre comillas dobles para diferenciarlos de otros elementos del lenguaje. Todas las sentencias de un programa Java deben terminar con el símbolo punto y coma. Este símbolo indica al compilador que ha finalizado una sentencia.

Se conoce como sentencia o instrucción. **System.out.println** es una sentencia que invoca al método **println** de la clase **System.out**, el cual muestra en la terminal el mensaje que le sigue, en este caso **Hola Mundo**.

El mensaje que esta entre paréntesis debe seguir ciertas condiciones como, estar entre comillas "..." el texto que se mostrará tal y conforme está.

Existe una serie de formatos para mostrar valores numéricos y de otros tipos que se verán más adelante.

Java es un lenguaje que pertenece a la categoría de lenguajes **compilados**, es decir hace uso de un compilador (**javac**), el que después de revisar la sintaxis del programa carga en un archivo intermedio `bytecode`, que será ejecutado por la JVM (*Máquina virtual Java*), todo el código binario necesario para ejecutar cada una de las sentencias del programa.

A continuación se muestra este proceso:

- Programa fuente

```
usuario@host:~$ cat HolaMundo.java
/*
Este programa escribe el mensaje "Hola Mundo"
en la terminal del computador
*/

public class HolaMundo {
     public static void main (String[] args) {
          System.out.println("Hola Mundo");
     }
}


usuario@host:~$
```

- Compilando el programa fuente

```
usuario@host:~$ javac HolaMundo.java
usuario@host:~$
```

- Programa fuente ya compilado

```
0000000000601040 B __bss_start
0000000000601040 b completed.6982
0000000000601030 D __data_start
0000000000601030 W data_start
0000000000400470 t deregister_tm_clones
00000000004004e0 t __do_global_dtors_aux
0000000000600e18 t __do_global_dtors_aux_fini_array_entry
0000000000601038 D __dso_handle
0000000000600e28 d _DYNAMIC
0000000000601040 D _edata
0000000000601048 B _end
00000000004005f4 T _fini
0000000000400500 t frame_dummy
0000000000600e10 t __frame_dummy_init_array_entry
0000000000400760 r __FRAME_END__
0000000000601000 d _GLOBAL_OFFSET_TABLE_
                 w __gmon_start__
00000000004003e0 T _init
0000000000600e18 t __init_array_end
0000000000600e10 t __init_array_start
0000000000400600 R _IO_stdin_used
                 w _ITM_deregisterTMCloneTable
                 w _ITM_registerTMCloneTable
0000000000600e20 d __JCR_END__
0000000000600e20 d __JCR_LIST__
                 w _Jv_RegisterClasses
00000000004005f0 T __libc_csu_fini
0000000000400580 T __libc_csu_init
                 U __libc_start_main@@GLIBC_2.2.5
000000000040052d T main
                 U printf@@GLIBC_2.2.5
00000000004004a0 t register_tm_clones
0000000000400440 T _start
0000000000601040 D __TMC_END__
```

- Ejecutando el programa compilado

```
usuario@host:~$ java HolaMundo
Hola Mundo
usuario@host:~$
```

3. Introducción la Programación Orientada a Objetos (POO)

La sintaxis de Java es cercana a los conceptos básicos de la **Programación Orientada a Objetos** (**POO**). Así la POO es una manera de diseñar y desarrollar software que trata de imitar en su esencia la realidad, tomando de ella algunos conceptos; como el de **objeto**, cuyos rasgos son: la identidad, el estado y el comportamiento.

Puedes leer mayor información en López Dávila (2010).

- La **identidad**, es el nombre que distingue a un objeto de otro.
- El **estado**, son las características que lo describen.
- El **comportamiento**, es lo que puede hacer el objeto.
- Se debe tener presente que los objetos, sean reales o su proyección en software, se **abstraen** en clases.

Por ejemplo: de la **clase perro** pueden existir **dos objetos** *Fido* y *Firuláis* (esta es su **identidad**). *Fido* es un perro de raza San Bernardo, enorme, pinto, de 5 años de edad; mientras que *Firuláis* es un perro de raza labrador, negro, de 3 años (este es su

estado). Ambos perros ladran, merodean, juguetean, comen y duermen (este es su **comportamiento**).

Si nos pidieran que hiciéramos un programa orientado a objetos que simulara lo anterior haríamos la clase **Perro** que tendría las variables **raza, color y edad**, y los métodos **ladrar(), merodear(), juguetear(), comer() y dormir()**. *Firuláis* y *Fido* son los **identificadores** que podríamos usar en una aplicación que pretenda mostrar dos objetos de la clase **Perro**.

Los conceptos de **abstracción** y de **encapsulamiento**, están muy ligados a la POO y tienen que ver con el diseño de programas. Ambos se refieren a que los objetos deben hacer tareas que les son propias y no de otros.

Los objetos de la realidad no dan problemas porque ya existen. No es difícil abstraer la clase Perro ni encapsular su comportamiento porque existe en la realidad. Para llevar esto al ámbito del software analicemos el caso del programa Hola Mundo. Hagamos el proceso de abstracción para encapsular sus características y su comportamiento.

El primer **Hola Mundo** lo popularizó **Brian Kernighan** en los años 70 del siglo XX, en un libro que causó mucho interés en su tiempo y que escribió junto a **Dennis Ritchie**: *The C Programming Language*. Hoy en día, es una tradición presentar los lenguajes con un programa de este tipo, que lo que debe hacer es mostrar la frase "Hola mundo" en la pantalla, y sirve para probar que el lenguaje está debidamente instalado y funcionando.

Entonces, abstrayendo esto, podemos decir que el comportamiento de los objetos del tipo **Hola Mundo** es mostrar un mensaje y su característica, el mensaje mismo.

Lo que sigue es mostrar cómo los elementos de lenguaje Java nos permiten apegarnos a la orientación a objetos; para eso es necesario conocer dichos elementos, pero antes, para finalizar, un breve resumen.

Hemos introducido cinco conceptos de la POO:

1. La **identidad**, que es el nombre que distingue a los objetos
2. El **estado**, que se refiere a sus características o atributos
3. El **comportamiento**, que indica los métodos que se deben programar para que los objetos realicen acciones
4. La **abstracción**, que es el mecanismo mental para aislar su naturaleza
5. El **encapsulamiento**, que exige que sus características y métodos estén bien definidos y no se confundan con los de otros

Dos conceptos muy importantes: la **herencia** y el **polimorfismo**, se verán más adelante, cuando el conocimiento del lenguaje facilite su comprensión.

4. Elementos del lenguaje de programación Java

La siguiente imagen muestra nuestro programa `HolaMundo.java`, el cual está programado con orientación a objetos

El siguiente programa presenta la primera versión del `Hola Mundo` orientado a objetos:

```
/*
Este programa Java escribe un texto en pantalla
*/

public class HolaMundo {
    public static void main (String args[]) {
        System.out.println("Hola mundo");
    }
}
```

Todo programa Java está conformado por:

- Bloques de código
- Comentario
- Identificadores
- Sentencias
- Expresiones
- Operadores
- Metacaracteres
- Palabras reservadas

A continuación explicaremos en que consiste cada uno de ellos:

4.1. Bloques de código

Un bloque de código es el encapsulamiento de un conjunto de sentencias, y posiblemente de otros bloques de código que el programador agrupa para el cometido de un propósito. Se delimita mediante los caracteres y .

Por ejemplo:

```
{
      System.out.println("Hola Mundo");
}
```

4.2. Comentarios

Es texto que se inserta en el código con fines de documentación y para facilitar la lectura del código.

Los tipos de comentarios más usados son:

- `//` Una sola línea de comentario. Se indica con dos caracteres / seguidos. Puede ubicarse en cualquier lugar de la línea. Todo lo que va posterior a `//` se considera como un comentario y no es tomado en cuenta por el compilador

```
// este es un comentario
```

- `/* */` Una o varias líneas de comentario. Empieza con los caracteres `/*` y termina con `*/`. Todo el texto entre estas marcas, que pueden abarcar de una a más lineas, se consideran comentarios.

```
/*
Esta linea es un comentario
*/
```

4.3. Identificadores

Son los nombres que pueden tener las clases, los métodos y las variables. No pueden contener espacios ni caracteres especiales.

El programador tiene libertad para elegir el nombre de las variables, los métodos y de otros elementos de un programa. Existen reglas muy estrictas sobre los nombres que se utilizan como identificadores de clases, de variables o de métodos.

Todo identificador debe empezar con una letra que puede estar seguida de más letras o dígitos. Una letra es cualquier símbolo del alfabeto y el carácter subrayado "_". Un dígito es cualquier carácter entre 0 y 9.

Los identificadores `Hola`, `hola`, `numero`, `numeroPar`, `numeroImpar`, `numero_impar`, `numero_par`, `nombre`, `apellido1` y `apellido2` son válidos. El identificador `1numero` **no es válido** porque empieza con un dígito, no con una letra.

Cualquier identificador que empiece con una letra seguida de más letras o dígitos es válido siempre que no forme parte de las palabras reservadas del lenguaje Java. El lenguaje Java distingue entre letras mayúsculas y minúsculas, esto significa que los identificadores `numeroPar` y `numeropar` son distintos.

Existen unas normas básicas para los identificadores que se deben respetar.

- Los nombres de **variables** y **métodos** empiezan con **minúsculas**. Si se trata de un nombre compuesto cada palabra debe empezar con mayúscula y no se debe utilizar el guión bajo para separar las palabras. Por ejemplo, los identificadores `calcularSueldo`, `setNombre` o `getNombre` son válidos.

- Los nombres de **clases** empiezan siempre con **mayúsculas**. En los nombres compuestos, cada palabra comienza con mayúscula y no se debe utilizar el guión bajo para separar las palabras. Por ejemplo, `HolaMundo`, `PerimetroCircunferencia`, `Alumno` o `Profesor` son nombres válidos.

- Los nombres de **constantes** se escriben en **mayúsculas**. Para nombres compuestos se utiliza el guion bajo para separar las palabras. Por ejemplo, `PI`, `MINIMO`, `MAXIMO` o `TOTAL_ELEMENTOS` son nombres válidos.

Tabla 1: Convención para escribir Identificadores

Tipo de identificador	Convención	Ejemplo
Clase	Comienza con mayúscula	`HolaMundo`
Método	Comienza con minúscula	`escribeTotal()`
Variable	Comienza con minúscula	`saldo`

Fuente López Dávila (2010)

Deben respetar ciertas convenciones según se indica en la tabla 1:

Si un identificador está formado por más de un vocablo (por ejemplo `Hola Mundo`), a partir del segundo las iniciales deben ser mayúsculas (`HolaMundo`).

Los nombres de las clases deben ser sustantivos, los de los métodos deben ser verbos y las variables deben expresar con claridad su propósito.

4.4. Sentencias

Son las órdenes que se debe ejecutar el programa y terminan siempre con un punto y coma (;)

Por ejemplo:

```
String nombre;
```

4.5. Expresiones

Las expresiones son entidades formadas por uno o más miembros unidos mediante operadores que los pueden evalúr o relacionar.

Por ejemplo;

```
saludo = "Hola Mundo";
```

4.6. Operadores

Los operadores son caracteres especiales para hacer acciones específicas y son el mecanismo con el cual los objetos interactúan relacionando los datos y devolviendo nuevos valores. Se clasifican así:

- Aritméticos
- Relacionales y lógicos
- De asignación
- Metacaracteres

Son caracteres particulares con propósito de control y con un significado puntual en sentencias y bloques de código:

```
( [ { \ ^ - $ | ] } ) ? * +
```

4.7. Palabras reservadas

Son un conjunto de palabras que realizan tareas especiales, delimitan el alcance de los objetos, sus datos y sus métodos, etc. Se pueden clasificar del modo siguiente:

- Tipos de datos
- Sentencias condicionales
- Sentencias iterativas
- Tratamiento de excepciones
- Estructura de datos
- Modificadores y control de acceso

5. Variables, constantes, tipos de datos

5.1. Variables

Java, como todo lenguaje de programación, utiliza variables para almacenar valores, realizar cálculos, los valores almacenados en memoria pueden ser modificados, mostrados por la pantalla, almacenados en disco, enviados por la red, etc.

Importante es saber que una variable almacena un único valor, y se define por un nombre y un tipo. El tipo establece el rango de valores que puede almacenar.

El nombre de una variable permite hacer referencia a ella. Este nombre debe cumplir las reglas que se aplican a los identificadores. El tipo, indica el formato de los valores que puede almacenar la variable, y éstos son: cadenas de caracteres, valores lógicos, números enteros, números reales o tipos de datos complejos. El rango indica los valores que puede tomar la variable.

Por ejemplo, una variable de tipo número entero, con nombre `mesNacimiento` puede almacenar valores positivos y negativos, lo que no tiene sentido cuando se trata de meses del año. El rango válido para esta variable sería de 1 a 12.

Para declarar una variable en Java se indica el tipo y su nombre.

```
int mesNacimiento;
```

En este ejemplo se indica que la variable es de tipo entero (`int`) y su nombre es `mesNacimiento`. Una vez declarada una variable, se puede utilizar en cualquier parte del programa referenciándola por su nombre. Para almacenar un valor en una variable se utiliza el operador de asignación (=) y a continuación se indica el valor que se le asigna, por ejemplo 2.

```
mesNacimiento = 2;
```

A partir de este momento la variable `mesNacimiento` almacena el valor 2 y cualquier referencia a ella utiliza este valor. Por ejemplo, si se imprime el valor de la variable por la consola, muestra el valor 2.

```
System.out.print(mesNacimiento);
```

Java permite declarar e inicializar una variable en una sola sentencia.

```
int diaNacimiento = 20;
int mesNacimiento = 4;
```

En este ejemplo se declara la variable `diaNacimiento` con el valor 20 y la variable `mesNacimiento` con valor 4.

5.2. Constantes

Una constante, a diferencia de una variable, es un nombre con un valor que no cambia durante la ejecución del programa.

Si se declara una constante, entonces se debe utilizar el delimitador **`final`** y es necesario indicar su valor. En la siguiente declaración se define el valor de la constante **`pi`** de tipo double para almacenar un número real.

```
final double PI = 3.1415926536;
```

Es conveniente utilizar nombres apropiados para las variables. **El nombre de una variable debe indicar para qué sirve**. El nombre de una variable debe explicarse por sí mismo para mejorar la legibilidad del programa.

Si se desea declarar más de una variable a la vez se deben separar las variables en la sentencia de la declaración.

```
int dia=20, mes=4, año=2020;
```

En este ejemplo se declaran las variables enteras `dia`, `mes` y `año`. La variable `día` se inicializa con el valor 20, `mes` con 4 y `año` con 2020. La siguiente declaración es equivalente a la anterior.

```
int dia=20;
int mes=4;
int año=2020;
```

5.3. Tipos de datos

Tipos primitivos

Las variables de Java pueden ser de un tipo primitivo de datos o una referencia a un objeto. Los tipos primitivos permiten representar valores básicos. Estos tipos se clasifican en números enteros, números reales, caracteres y valores booleanos.

Números enteros

Los enteros representan números sin punto decimal ni fracción decimal, pueden ser positivos y negativos con distintos rangos de valores, desde cientos a trillones. Los tipos enteros de Java son `byte`, `int`, `short` y `long`.

Números reales

Un tipo real, es aquel valor que tiene punto decimal, pudiendo tener o no fracción decimal. Existen dos tipos de números reales en Java, `float` y `double`. La diferencia

Tabla 2: Tipos primitivos de datos en Java

Tipo	Descripción	Valor mínimo y máximo
`byte`	Entero con signo	`-128` a `127`
`short`	Entero con signo	`-32768` a `32767`
`int`	Entero con signo	`-2147483648` a `2147483647`
`long`	Entero con signo	`-9221703685477580`8 a
		`+9221703685477580`7
`float`	Real de precisión simple	`±3.40282347e+38` a
		`±1.40239846e-45`
`double`	Real de precisión doble	`±1.7976931348623157e+309` a
		`±4.94065645841246544e-324`
`char`	Caracteres Unicode	$\backslash u0000$ a $\backslash uFFFF$
`boolean`	Valores lógicos	`true`, `false`

Fuente: Martínez Ladrón de Guevara (2011).

entre ellos está en el número de decimales que tienen capacidad para expresar y en sus rangos de valores.

Caracteres

El tipo char permite representar cualquier carácter Unicode. Los caracteres `Unicode` contienen todos los caracteres del alfabeto de la lengua castellana.

Booleano

Se utiliza para representar los valores lógicos verdadero y falso. Solo tiene dos valores `true` y `false`.

La tabla 2 resume los tipos primitivos de Java.

6. Nomenclatura habitual en la programación en Java

Tal como lo describe García de Jalón y cols. (2000), Java es sensible a las mayúsculas y minúsculas. Así, las variables `masa`, `Masa` y `MASA` son consideradas variables completamente diferentes.

Las reglas del lenguaje respecto a los nombres de variables son muy amplias y permiten mucha libertad al programador, pero es habitual seguir ciertas normas que facilitan la lectura y el mantenimiento de los programas de ordenador.

Se recomienda seguir las siguientes instrucciones:

- En Java es habitual utilizar nombres con minúsculas, con las excepciones que se indican en los puntos siguientes.
- Cuando un nombre consta de varias palabras es habitual poner una a continuación de otra, sin espacios, poniendo con mayúscula la primera letra de la palabra que sigue a otra (Ejemplos: `elMayor()`, `VentanaCerrable`, `RectanguloGrafico`, `addWindowListener()`).

- Los nombres de clases e interfaces comienzan siempre por mayúscula (Ejemplos: `Geometria`, `Rectangulo`, `Dibujable`, `Graphics`, `ArrayList`, `Iterator`).
- Los nombres de objetos, los nombres de métodos y variables miembro, y los nombres de las variables locales de los métodos, comienzan siempre por minúscula (Ejemplos: `main()`, `dibujar()`, `numRectangulos`, `x`, `y`, `r`).
- Los nombres de las variables finales, es decir de las constantes, se definen siempre con mayúsculas (Ejemplo: `PI`)

Referencias

García de Jalón, J., Rodríguez, J. I., Mingo, I., Imaz, A., Brazález, A., Larzabal, A., ... García, J. (2000). *Aprende java como si estuviera en primero*. tecnun.

López Dávila, J. C. (2010). *Curso de java* (1.ª ed.; C. training & consulting, Ed.). 3CT.

Martínez Ladrón de Guevara, J. (2011). *Fundamentos de programación en java* (U. C. de Madrid, Ed.). G-TEC.