



Día, Fecha:

Jueves, 27/07/2023

Hora de inicio:

07:10

Introducción a la programación y computación 1 [D]

José Eduardo Morales García

¿Qué es Java?

- ▶ Java es un lenguaje de programación de alto nivel, desarrollado por Sun Microsystems en 1995. Es uno de los lenguajes más populares en la industria, utilizado para desarrollar aplicaciones para la web, móviles, escritorio y servidores.

Características de Java

- Orientado a objetos: Java permite crear programas utilizando una estructura de objetos y clases.
- Plataforma independiente: Java se ejecuta en una máquina virtual, lo que permite que el mismo código sea ejecutado en diferentes sistemas operativos sin necesidad de recompilarlo.
- Tipado fuerte: Java es un lenguaje de tipado estático, lo que significa que las variables deben ser declaradas con un tipo específico antes de su uso.
- Seguridad: Java cuenta con medidas de seguridad integradas, como la verificación de tipos y la restricción de acceso a los objetos.

Aplicaciones de Java

- Aplicaciones web: Java es utilizado para desarrollar aplicaciones web dinámicas utilizando tecnologías como JavaServer Faces (JSF) y JavaServer Pages (JSP).
- Aplicaciones móviles: Java es el lenguaje de programación utilizado para desarrollar aplicaciones móviles en sistemas operativos como Android.
- Aplicaciones de escritorio: Java permite crear aplicaciones de escritorio con una interfaz gráfica de usuario (GUI) utilizando herramientas como JavaFX.
- Aplicaciones de servidor: Java es utilizado para desarrollar aplicaciones de servidor, como servlets y aplicaciones web en Java Enterprise Edition (JEE).

Ventajas de usar Java

- Amplia comunidad: Java cuenta con una gran comunidad de desarrolladores, lo que significa que hay una gran cantidad de recursos disponibles para aprender y resolver problemas.
- Gran cantidad de librerías: Java cuenta con una gran cantidad de librerías disponibles para realizar tareas comunes, como conectarse a bases de datos o enviar correos electrónicos.
- Escalabilidad: Java es un lenguaje escalable, lo que permite crear aplicaciones pequeñas o grandes según sea necesario.

¿Qué es el JDK de Java?

- El JDK (Java Development Kit) es un conjunto de herramientas de desarrollo de Java que permite crear, ejecutar y depurar aplicaciones Java. Incluye el compilador, la máquina virtual de Java (JVM) y otras herramientas necesarias para desarrollar aplicaciones Java.

Características del JDK

- **Compilador:** El compilador de Java del JDK convierte el código fuente de Java en bytecode, que luego se ejecuta en la JVM.
- **Máquina Virtual de Java (JVM):** La JVM es el componente que ejecuta el bytecode generado por el compilador y se encarga de la gestión de memoria y de la seguridad de la aplicación.
- **Bibliotecas:** El JDK incluye una gran cantidad de bibliotecas estándar de Java que facilitan el desarrollo de aplicaciones.
- **Documentación:** El JDK incluye la documentación oficial de Java, que proporciona información detallada sobre el uso de las bibliotecas y las herramientas del JDK.

Herramientas incluidas en el JDK

- `javac`: El compilador de Java, utilizado para convertir el código fuente en bytecode.
- `java`: La herramienta para ejecutar el bytecode generado por el compilador.
- `javadoc`: La herramienta para generar documentación a partir del código fuente.
- `jdb`: El depurador de Java, utilizado para encontrar y corregir errores en el código.
- `javap`: La herramienta para descompilar bytecode en código fuente.

Conociendo Java



Comentarios en Java

- Los comentarios son una forma de incluir notas o explicaciones dentro de un programa de manera que no son interpretadas por el compilador. En Java existen dos tipos de comentarios: los de una sola línea y los de varias líneas. Los comentarios de una sola línea se inician con "//" y terminan al final de la línea. Los comentarios de varias líneas se inician con "/*" y *terminan con "*/"*.

// Comentario de una línea

/* Comentario
de varias
líneas */

Identificadores en Java

- Los identificadores son nombres asignados a variables, métodos y clases en Java. Los identificadores deben comenzar con una letra, un guión bajo o un signo de dólar. A partir de ahí, pueden contener letras, números y los caracteres "_" y "\$". No pueden ser palabras reservadas de Java.

Ejemplos:

`nombreIDentificador`

`_nombreidentificador`

`$nombreIdentificador`



Tipos de datos en Java

DATOS PRIMITIVOS Y NO
PRIMITIVOS

¿Qué son los datos primitivos en Java?

Los datos primitivos en Java son los tipos de datos básicos que no tienen métodos ni atributos. Estos incluyen `int`, `double`, `boolean` y `char`. Los datos primitivos son almacenados en el stack (pila de memoria) y son más rápidos y eficientes en términos de uso de memoria.

- `int`: para números enteros
- `double`: para números con decimales
- `boolean`: para valores lógicos (`true` o `false`)
- `char`: para caracteres
- `byte`: para números enteros de 8 bits
- `short`: para números enteros de 16 bits
- `long`: para números enteros de 64 bits
- `float`: para números con decimales de 32 bits

Rangos de valores que soportan los datos primitivos

Tipo	Descripción
boolean	Tiene dos valores true o false .
char	Caracteres Unicode de 16 bits. Los caracteres alfa-numéricos son los mismos que los ASCII con el bit alto puesto a 0. El intervalo de valores va desde 0 hasta 65535 (valores de 16-bits sin signo).
byte	Tamaño 8 bits. El intervalo de valores va desde -2^7 hasta 2^7-1 (-128 a 127)
short	Tamaño 16 bits. El intervalo de valores va desde -2^{15} hasta $2^{15}-1$ (-32768 a 32767)
int	Tamaño 32 bits. El intervalo de valores va desde -2^{31} hasta $2^{31}-1$ (-2147483648 a 2147483647)
long	Tamaño 64 bits. El intervalo de valores va desde -2^{63} hasta $2^{63}-1$ (-9223372036854775808 a 9223372036854775807)
float	Tamaño 32 bits. Números en coma flotante de simple precisión. Estándar IEEE 754-1985 (de 1.40239846e-45f a 3.40282347e+38f)
double	Tamaño 64 bits. Números en coma flotante de doble precisión. Estándar IEEE 754-1985. (de 4.94065645841246544e-324d a 1.7976931348623157e+308d.)

¿Qué son los datos no primitivos en Java?

Los datos no primitivos en Java son aquellos que tienen métodos y atributos. Estos incluyen objetos y arreglos. Los datos no primitivos son almacenados en el heap (área de memoria) y son más flexibles en cuanto a su uso, pero requieren más memoria y son más lentos.

- `String`: para cadenas de caracteres
- `ArrayList`: para listas dinámicas de objetos
- `Scanner`: para la lectura de datos desde el teclado
- `File`: para el manejo de archivos
- `Date`: para el manejo de fechas

Diferencias entre datos primitivos y no primitivos en Java

- Los datos primitivos son almacenados en el stack y los no primitivos en el heap.
- Los datos primitivos son más rápidos y eficientes en términos de uso de memoria, mientras que los no primitivos son más flexibles pero requieren más memoria y son más lentos.
- Los datos primitivos no tienen métodos ni atributos, mientras que los no primitivos si los tienen.
- Los datos primitivos son simples y no pueden ser modificados parcialmente, mientras que los no primitivos son más complejos y pueden ser modificados.

Constantes en Java

- ▶ Las constantes son valores que no cambian durante la ejecución de un programa. En Java, se pueden declarar constantes utilizando la palabra clave "final" antes del tipo de dato.

Por ejemplo:

```
final int EDAD_MAXIMA = 100;
```

Una vez que una constante ha sido declarada y asignada un valor, no se puede cambiar su valor durante la ejecución del programa.

¿Qué son los arreglos en Java?



- Los arreglos en Java son estructuras de datos que permiten almacenar y acceder a varios valores de un mismo tipo de forma ordenada. Los arreglos se pueden crear para almacenar cualquier tipo de dato, incluyendo datos primitivos y no primitivos.

Para crear un arreglo en Java, se utiliza la sintaxis:

```
tipoDeDato[] nombreArreglo = new tipoDeDato[tamaño];
```

Los elementos del arreglo se pueden acceder utilizando el nombre del arreglo seguido de un índice entre corchetes, por ejemplo:

```
nombreArreglo[índice];
```



Métodos de ordenamiento para arreglos

ORDENAMIENTO POR BURBUJA

ORDENAMIENTO POR SELECCIÓN

ORDENAMIENTO POR INSERCIÓN

ORDENAMIENTO RÁPIDO (QUICK SORT)

Ordenamiento por burbuja

- El ordenamiento por burbuja es un algoritmo simple que compara cada elemento del arreglo con su siguiente elemento, intercambiándolos si están en un orden incorrecto. Este proceso se repite varias veces hasta que el arreglo está completamente ordenado. Este método es fácil de implementar pero tiene una complejidad $O(n^2)$.

Ordenamiento por selección

- El ordenamiento por selección es un algoritmo que busca el elemento más pequeño del arreglo y lo intercambia con el primer elemento no ordenado. Luego, busca el segundo elemento más pequeño y lo intercambia con el segundo elemento no ordenado, y así sucesivamente. Este método es más eficiente que el ordenamiento por burbuja, con una complejidad $O(n^2)$.

Ordenamiento por inserción

- El ordenamiento por inserción es un algoritmo que toma cada elemento no ordenado del arreglo y lo inserta en su posición correcta en el arreglo ordenado. Este método es similar al ordenamiento manual de un mazo de cartas y es más eficiente que el ordenamiento por burbuja y selección, con una complejidad de $O(n^2)$ en el peor caso, pero $O(n)$ en el mejor caso.

Ordenamiento rápido (Quick Sort)

- El ordenamiento rápido (Quick Sort) es un algoritmo de ordenamiento divide y vencerás que es muy eficiente en términos de complejidad. Este método selecciona un pivote del arreglo y divide el arreglo en dos partes, una con elementos menores al pivote y otra con elementos mayores al pivote. Luego, aplica el mismo proceso de forma recursiva a cada parte del arreglo hasta que todos los elementos estén ordenados. La complejidad de Quick Sort es $O(n \log n)$ en el mejor y peor caso.

The background is a dark purple gradient. It features several abstract shapes: a large light purple circle on the right, a smaller light purple circle above it, a light purple semi-circle on the left, and a pink rectangle in the top right corner.

Parte practica