Programación en Java – Clase 1

Contenido

Un poco de historia

Java es un lenguaje de programación que fue creado por James Gosling y su equipo en Sun Microsystems en la década de 1990. Gosling y su equipo estaban trabajando en un proyecto llamado "Green Project" para crear un lenguaje de programación que pudiera ser utilizado en dispositivos electrónicos como televisores y electrodomésticos.

Inicialmente, el lenguaje se llamaba "Oak", pero luego se cambió el nombre a "Java" debido a problemas legales con el nombre original. La primera versión de Java, Java 1.0, fue lanzada en 1996 y se convirtió en un éxito instantáneo debido a su capacidad para funcionar en múltiples plataformas y sistemas operativos.

En 1999, Sun Microsystems lanzó Java 2, que incluía mejoras significativas en el lenguaje y la plataforma Java. A lo largo de los años, Java ha seguido evolucionando y mejorando, con numerosas versiones lanzadas para agregar nuevas características y mejorar la seguridad y la eficiencia.

Hoy en día, Java es uno de los lenguajes de programación más populares y utilizados en todo el mundo, y es utilizado en una amplia variedad de aplicaciones, desde el desarrollo de aplicaciones web y móviles hasta la programación de sistemas y aplicaciones empresariales.

Recomendaciones y primeros pasos

1) Instalar un entorno de desarrollo integrado (IDE)

El IDE te permitirá escribir, compilar y ejecutar código Java de manera más fácil y eficiente. Algunos de los IDE más populares para Java son Eclipse, NetBeans y IntelliJ IDEA. Puedes descargarlos e instalarlos en tu computadora.

2) Aprender los conceptos básicos de Java

Antes de comenzar a escribir código, debemos comprender los conceptos básicos de Java. Algunos de los conceptos clave que aprenderemos son las variables, los tipos de datos, las estructuras de control de flujo, los arreglos, los métodos, las clases y los objetos.

3) Escribir tu primer programa en Java

Una vez que tengas instalado el IDE y hayas aprendido los conceptos básicos de Java, es hora de escribir tu primer programa. Comenzaremos con un programa simple, como imprimir "Hola, mundo" en la consola. Esto ya se ha hecho una tradición al momento de empezar a programar en cualquier lenguaje.

4) Practicar y experimentar

La programación es un proceso iterativo y requerirá de práctica constante. Es importante que practiquemos y experimentemos con diferentes tipos de programas para desarrollar tus habilidades como programador.

5) Aprender de otros programadores

Un gran recurso para aprender Java es aprovechar la comunidad de programadores. Hay muchos foros, grupos y comunidades donde podemos aprender de otros programadores, hacer preguntas y compartir tus propias experiencias.

¿Dónde escribo en JAVA?

Eclipse: es un entorno de desarrollo integrado (IDE) de código abierto que se utiliza principalmente para desarrollar aplicaciones en Java, aunque también soporta otros lenguajes de programación

JDoodle: Es una herramienta en línea donde puedes compilar, ejecutar y compartir código Java. Ofrece soporte para múltiples versiones de Java y

también tiene características como la ejecución de programas interactivos y la integración con otros lenguajes. Sitio web: JDoodle

Otras alternativas: existen otros software y plataformas donde programar con Java, tales como intellij, Visual Studio Code (requiere configuración), Visual Code, Replit.com (limitado), entre otros.

Creando nuestro primer programa

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

A continuación, explicaremos cada una de las líneas que componen a nuestro programa. Seguramente encontrarás muchas palabras y conceptos nuevos, con el correr de las clases iremos trabajando sobre cada uno.

1) **class Main { }:**
   Es la definición de la clase Main. En Java, todos los programas se escriben en clases. El nombre de la clase **debe coincidir con el nombre del archivo que contiene la clase**.

2) **public static void main(String[] args) { }:**
   Es el método main, que es el punto de entrada del programa. Es el método que se ejecutará cuando se inicie el programa. Los argumentos de la línea de comandos se pasan como un arreglo de cadenas en el parámetro args.

3) **System.out.println("Hola mundo!");**
   Esta es una llamada al método println de la clase System.out. println es un método que imprime una línea de texto en la consola.
   En este caso, estamos imprimiendo la cadena **"Hola mundo!"** en la consola.

Tipos de datos

Existen dos tipos de datos principales: los tipos de datos primitivos y los tipos de datos de referencia.

a) Los tipos de datos primitivos son aquellos que representan valores simples y no tienen métodos. Hay ocho tipos de datos primitivos en Java:

- byte: representa un número entero de 8 bits con signo.
- short: representa un número entero de 16 bits con signo.
- int: representa un número entero de 32 bits con signo.
- long: representa un número entero de 64 bits con signo.
- float: representa un número de punto flotante de precisión simple de 32 bits.
- double: representa un número de punto flotante de doble precisión de 64 bits.
- char: representa un carácter Unicode de 16 bits.
- boolean: representa un valor booleano true o false.

b) Los tipos de datos de referencia son aquellos que representan objetos y tienen métodos. Estos tipos de datos se definen mediante clases,

interfaces y arreglos. Algunos ejemplos de tipos de datos de referencia en Java son String, Integer, Double, ArrayList, HashMap, entre otros.

Es importante tener en cuenta que los tipos de datos primitivos en Java son más eficientes en términos de memoria y velocidad que los tipos de datos de referencia, ya que estos últimos requieren de una instancia de objeto que ocupa más memoria y tienen un mayor costo de acceso a sus métodos.

Declaración de variables

Para declarar una variable primitiva debemos especificar el tipo de dato y el nombre de la variable. A continuación, te acercamos un ejemplo:

```
class Main {
    public static void main(String[] args) {

        // Declaración de variables primitivas
        byte edad = 25;
        short codigo = 1234;
        int cantidad = 1000;
        long telefono = 55555555L;
        float precio = 999.9f;
        double sueldo = 75000.50;
        char inicial = 'J';
        boolean activo = true;

        // Impresión de variables
        System.out.println("Edad: " + edad);
```

```
            System.out.println("Código: " + codigo);
            System.out.println("Cantidad: " + cantidad);
            System.out.println("Teléfono: " + telefono);
            System.out.println("Precio: " + precio);
            System.out.println("Sueldo: " + sueldo);
            System.out.println("Inicial: " + inicial);
            System.out.println("Activo: " + activo);
        }
}
```

En este ejemplo, estamos declarando variables primitivas de los siguientes tipos: byte, short, int, long, float, double, char, y boolean.

Cada variable se inicializa con un valor específico. Luego, imprimimos el valor de cada variable en la consola utilizando el método println de la clase System.out.

Lectura de datos desde el teclado

Existen diferentes formas de leer datos desde el mundo exterior, para nosotros por el momento ese contacto será a través del teclado.

```
import java.util.Scanner;

class Main {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Ingrese un entero: ");
            int numero = scanner.nextInt();
            System.out.println("Ingrese un número de punto flotante: ");
```

```
        double flotante = scanner.nextDouble();
        System.out.println("Ingrese un carácter: ");
        char caracter = scanner.next().charAt(0);
        System.out.println("Ingrese un valor booleano (true o false): ");
        boolean bool = scanner.nextBoolean();

        // Mostrar los valores ingresados por teclado
        System.out.println("Número entero ingresado: " + numero);
        System.out.println("Número de punto flotante ingresado: " +
        flotante);
        System.out.println("Carácter ingresado: " + caracter);
        System.out.println("Valor booleano ingresado: " + bool);
        scanner.close();
    }
}
```

En este ejemplo, estamos usando la clase Scanner de Java (necesitamos importarlo a nuestro espacio de trabajo, para eso usamos la palabra import.) para leer los valores ingresados por el usuario desde teclado.
Primero, creamos un objeto Scanner llamado scanner y lo inicializamos con System.in, que es una entrada de teclado estándar.
Luego, leemos un entero, un número de punto flotante, un carácter y un valor booleano usando los métodos nextInt(), nextDouble(), next().charAt(0) y nextBoolean(), respectivamente, del objeto Scanner.
Finalmente, imprimimos los valores ingresados por teclado utilizando variables primitivas.

AVANZADO: Veamos a continuación, otra estrategia y en particular que diferencias poseen estas formas de resolución:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


class Main {
        public static void main(String[] args) throws IOException  {
        BufferedReader reader = new BufferedReader(new
        InputStreamReader(System.in));

        System.out.println("Ingrese su nombre: ");
        String nombre = reader.readLine();

        System.out.println("Ingrese su edad: ");
        int edad = Integer.parseInt(reader.readLine());

        // Mostrar los valores ingresados por teclado
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
        reader.close();
  }
}
```

En este ejemplo, estamos utilizando la clase BufferedReader para leer una línea completa de texto desde el teclado.

Primero, creamos un objeto BufferedReader llamado reader y lo inicializamos con una instancia de InputStreamReader, que lee la entrada estándar del sistema (System.in).
Luego, leemos una línea de texto para el nombre y otra para la edad, usando el método readLine() del objeto BufferedReader.
Finalmente, imprimimos los valores ingresados por el usuario utilizando variables.

Este código es más complejo que el anterior. No te preocupes si no lo entiendes en este momento, lo retomaremos más adelante en el curso.
Con esto, podemos observar que existen distintas maneras posibles para llegar a resolver una situación.