Plan de Formación Profesional y Continua

cursos gratuitos
de formación profesional

Módulo 1:

El Entorno Java para la Programación

Introducción 👨‍💻

Java es uno de los lenguajes de programación más utilizados en la industria del software en la actualidad. Su popularidad se debe a su versatilidad y amplio uso en el desarrollo de diversos productos.

Este lenguaje **es esencial para implementar nuevas funcionalidades y solucionar problemas en aplicaciones existentes.**

En este módulo aprenderemos uno de los conceptos esenciales en el mundo de la programación: los **algoritmos**. Comenzaremos comprendiendo su importancia y cómo se aplican en Java para resolver problemas de toda índole.

Exploraremos cómo almacenar y manipular información clave dentro de nuestros algoritmos Java a través de las variables.

A lo largo de esta lección, adquirirás las habilidades necesarias para construir la estructura inicial para desarrollar programas básicos en Java de forma eficaz.

Aprendizaje esperado

Al finalizar la lección podrás:

- Reconocer las características fundamentales del lenguaje Java para el desarrollo de aplicaciones eficientes.

- Comprender los principales fundamentos de la programación.

Conociendo Java 👇

¿Qué es Java?

Es un lenguaje de programación de alto nivel y orientado a objetos, que fue desarrollado por Sun Microsystems (actualmente adquirido por Oracle) en la década de 1990. Es conocido por su portabilidad, lo que significa que los programas escritos en Java pueden ejecutarse en diferentes plataformas sin necesidad de realizar cambios significativos en el código fuente.

Características del lenguaje

Java posee algunas características distintivas:

- **Portabilidad:** es altamente portátil, lo que significa que el código Java puede ejecutarse en diferentes plataformas sin cambios significativos. Esto se logra mediante el uso de la Máquina Virtual de Java (JVM), que interpreta el código Java y la ejecuta en cualquier dispositivo o sistema operativo que tenga una implementación de JVM.
- **Orientación a objetos:** está diseñado siguiendo el paradigma de programación orientada a objetos (POO). Esto significa que se basa en la creación y manipulación de objetos, lo que promueve la modularidad, reutilización de código y facilita el mantenimiento y la escalabilidad de las aplicaciones.
- **Seguridad:** tiene un enfoque fuerte en seguridad. El código se ejecuta en un entorno controlado, lo que ayuda a prevenir posibles vulnerabilidades y protege los sistemas de ejecutar código malicioso.
- **Multiplataforma:** permite desarrollar aplicaciones que son independientes de la plataforma, lo que significa que pueden ejecutarse en diferentes sistemas operativos, como Windows, macOS, Linux, entre otros.
- **Bibliotecas y comunidad amplias:** cuenta con una amplia gama de bibliotecas y frameworks disponibles que facilitan el desarrollo de aplicaciones.

¿Qué es la JVM (Java Virtual Machine)? 👩🏻‍💻

JVM, en español **Máquina Virtual de Java** es la encargada de interpretar y ejecutar el código Java. Proporciona portabilidad, gestión automática de la memoria y características de seguridad para las aplicaciones Java, permitiendo que se ejecuten en diferentes sistemas operativos sin necesidad de modificar el código fuente.

JDK (Java Development Kit)

Es un conjunto de herramientas que los desarrolladores utilizan para crear aplicaciones Java. Incluye el compilador de Java, que convierte el código fuente en bytecode, así como otras utilidades y bibliotecas que son necesarias para el desarrollo de software en Java.

JRE (Java Runtime Environment)

Es el entorno de ejecución de Java. Proporciona las bibliotecas y componentes necesarios para ejecutar aplicaciones Java en un sistema. Incluye la JVM, que interpreta y ejecuta el bytecode generado por el compilador.

Bytecode

Es un código intermedio que se genera cuando se compila un programa Java. Es un conjunto de instrucciones de bajo nivel que está diseñado para ser ejecutado por la JVM.
El bytecode es independiente de la plataforma, lo que significa que puede ser ejecutado en cualquier sistema operativo que tenga una implementación de JVM.

En resumen, el **JDK** es un conjunto de herramientas para desarrollar aplicaciones Java, el **JRE** es el entorno de ejecución para ejecutar aplicaciones Java y el **bytecode** es el código intermedio generado por el compilador de Java que es interpretado por la **JVM**.

¿Qué se puede hacer con Java?

Java es un lenguaje versátil y de uso gratuito que crea software localizado y distribuido.
Algunos usos comunes de Java incluyen:

- **Desarrollo de videojuegos** para móviles y computadoras.

- Java a menudo se conoce como **WORA: escribir una vez y ejecutar en cualquier lugar** (por sus siglas en inglés “Write Once and Run Anywhere”), lo que lo hace perfecto para aplicaciones descentralizadas basadas en la nube.

- Se usa para motores de **procesamiento de datos** que pueden trabajar con conjuntos de datos complejos y cantidades masivas de datos en tiempo real.

- Es una fuente inagotable de bibliotecas de machine learning. Su estabilidad y velocidad lo hacen perfecto para el **desarrollo de aplicaciones de inteligencia artificial** como el procesamiento del lenguaje natural y el aprendizaje profundo.

- **Internet de las cosas (IoT)**: Java se utiliza para programar sensores y hardware en dispositivos de periferia que pueden conectarse de forma independiente a Internet.

Evolución del lenguaje

Java ya tiene una larga historia a sus espaldas desde su primera aparición estable en 1995. En la actualidad la última versión estable es su versión 17 y cómo es de esperar su evolución no solo ha sido constante sino también, en ocasiones, revolucionaria:

- **Versión 1.0:** Introducción del lenguaje con las clases principales, la máquina virtual y el API gráfico de AWT.

- **Versión 1.1:** Inclusión de clases adicionales, estándar de JavaBeans y API de JDBC para conexión a bases de datos.

- **Versión 1.2:** Incorporación del framework de Collections y el API de Swing para interfaces de ventanas más complejas.

- **Versión 1.3:** Avances en la arquitectura de la máquina virtual con la introducción de la máquina HotSpot con compilación JIT.

- **Versión 1.4:** Fuerte soporte de XML, expresiones regulares, criptografía, entre otros.

- **Versión 1.5:** Inclusión de tipos Genéricos, metadatos con el uso de anotaciones y ampliación de APIs para programación concurrente.

- **Versión 1.6:** Introducción de un API de compilación "on-the-fly" para gestionar servicios web.
- **Versión 1.7:** Mejora de la máquina virtual con nuevos recolectores de basura.

- **Versión 1.8:** Apertura a la programación funcional con el uso de expresiones Lambda y Streams, revisión de APIs y actualización de la gestión de fechas.

- **Versión 11:** el cambio fundamental en Java 11, es sin duda JavaFX que ha sido eliminado de la implementación estándar de la tecnología, para convertirse en un módulo independiente. También coincide con un importante cambio de política de uso por parte de Oracle y el comienzo de un soporte extendido a usuarios premium.

Estas versiones representan importantes hitos en la evolución de Java, incorporando nuevas funcionalidades y mejoras en el lenguaje y en las bibliotecas disponibles.

Actualmente existen nuevas versiones estables, pero la mayoría de las implementaciones se ejecutan en **Java 1.8** y **Java 11.**

Descarga e instalación del JDK

La escritura de aplicaciones y applets de Java necesita herramientas de desarrollo como JDK. JDK incluye Java Runtime Environment, el compilador Java y las API de Java. Familiarizarse resulta fácil para los programadores nuevos y con experiencia.

A continuación verás el paso a paso para poder descargar e instalar el JDK:

1. Ve al sitio web oficial de Oracle Java SE en:
https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html

2. Deberás ir hacia abajo en la página para poder descargar la versión de Java 8. En la sección "**Java SE Development Kit 8uXXX**", donde "XXX" representa la última versión disponible, haz clic en el enlace de descarga correspondiente a tu sistema operativo (Windows, macOS o Linux).

3. Acepta los términos y condiciones del acuerdo de licencia.
4. Selecciona el archivo de instalación adecuado para tu sistema operativo y descárgalo en tu computadora.

5. Una vez que la descarga haya finalizado, ejecuta el archivo de instalación.

6. Sigue las instrucciones del asistente para completar el proceso de instalación. Asegúrate de seleccionar la ubicación donde se instalará el JDK (recomendamos que el directorio de JDK se encuentre en la carpeta Archivos de Programa) .

7. Una vez que la instalación se haya completado, verifica si el JDK 8 se ha realizado correctamente abriendo una ventana de terminal o símbolo del sistema y ejecutando el comando **"java -version".** Deberías ver información sobre la versión del JDK instalado.

¡Listo! Ahora tienes JDK 8 instalado en tu sistema y estás listo para comenzar a desarrollar aplicaciones Java con esta versión.

¿Qué es un IDE?

IDE es la abreviatura en inglés de **Entorno de Desarrollo Integrado**. Es un sistema de software para el diseño de aplicaciones, que brinda varias herramientas comunes para desarrolladores en una sola interfaz gráfica. Combina un editor de texto avanzado con características como resaltado de sintaxis, autocompletado de código y sugerencias, facilitando la escritura de código.

Otras características que es común encontrar en un IDE son: navegador del código fuente, buscador, integración con sistemas de control de versión como Git, asistentes para reestructurar (refactor), indicador de errores y advertencias de malas prácticas y ejecutar pruebas (tests).

Eclipse IDE

Eclipse es el IDE más usado para el desarrollo en Java. Es de código abierto y gratuito. Permite extender sus funciones mediante el desarrollo de plugins.

Existen muchas versiones de Eclipse para programar en diferentes lenguajes e incluso posee varias versiones para Java.

Descarga e instalación del IDE

1. Ve al sitio web oficial de Eclipse en: https://www.eclipse.org/downloads/

En esa página encontrarás el botón de descarga, al hacerle click te llevará a la siguiente pantalla:

2. Una vez que se haya completado la descarga, abre el archivo de instalación. Verás que se despliegan varias opciones de IDE para seleccionar. Para este caso, recomendamos descargar la versión **Eclipse IDE for Enterprise Java and Web Developers**

3. Sigue las instrucciones del asistente de instalación para configurar Eclipse. Puedes elegir la ubicación de instalación y las opciones adicionales según tus preferencias.

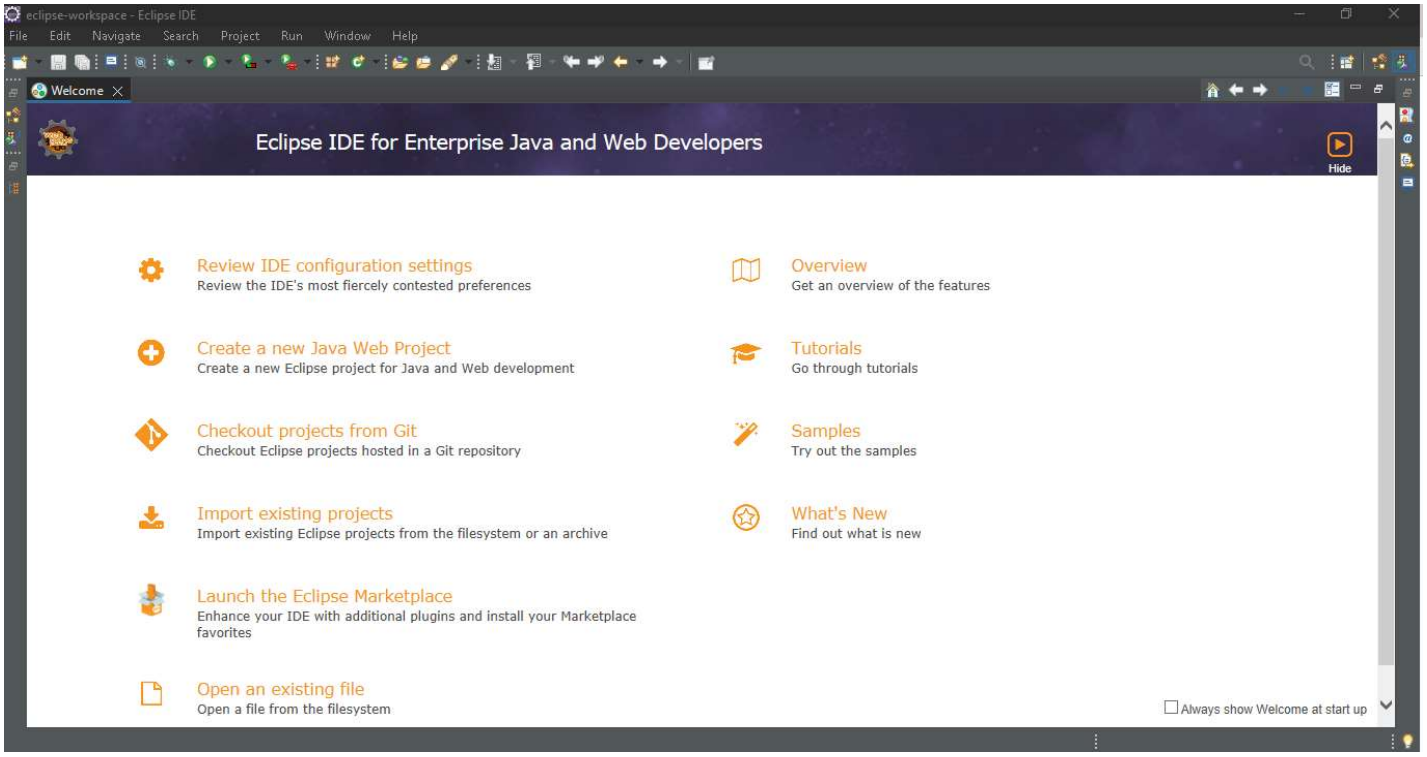
4. Una vez que la instalación haya finalizado, inicia Eclipse. Te pedirá que selecciones un "Workspace" (espacio de trabajo), que es la ubicación donde se almacenarán tus proyectos.

5. Después de seleccionar el "Workspace", aparecerá la ventana principal de Eclipse. ¡Ahora estás listo para empezar a desarrollar!

Recuerda que Eclipse es altamente personalizable y puedes instalar complementos y ajustar la configuración según tus necesidades.

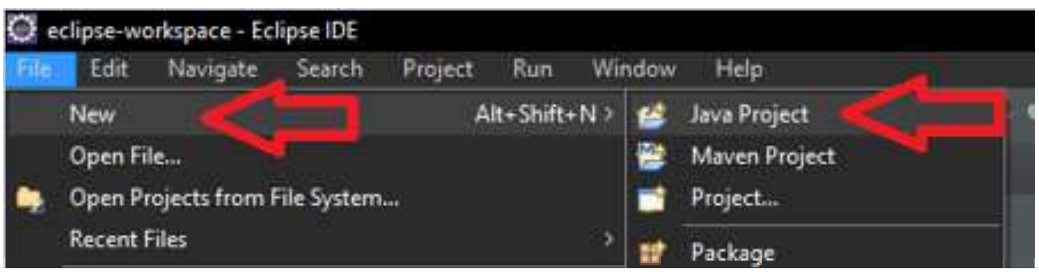
Creación de un programa en Java

Vamos a crear un programa desde cero. Para esto empezaremos por abrir el Eclipse IDE, donde verás la pantalla de bienvenida:

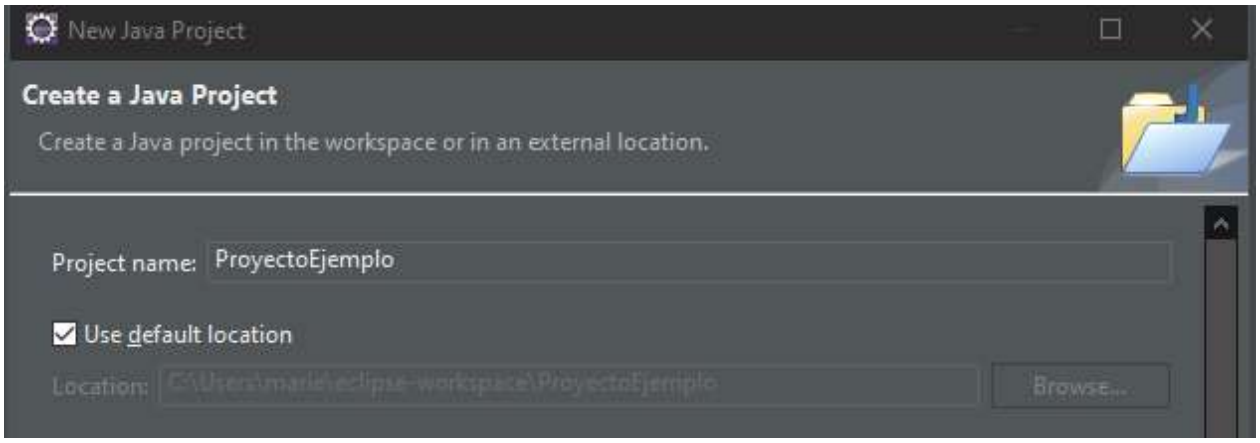


Esta pantalla ofrece información sobre el IDE y posee algunos atajos para poder acceder a diversas funcionalidades. Puedes cerrar la pestaña desde la parte superior izquierda.

Luego, dirígete hacia el menú superior y selecciona la primera opción: **File.**
Al hacer click, se desplegarán las opciones en donde debes seleccionar **New**, y a continuación **Java Project**.



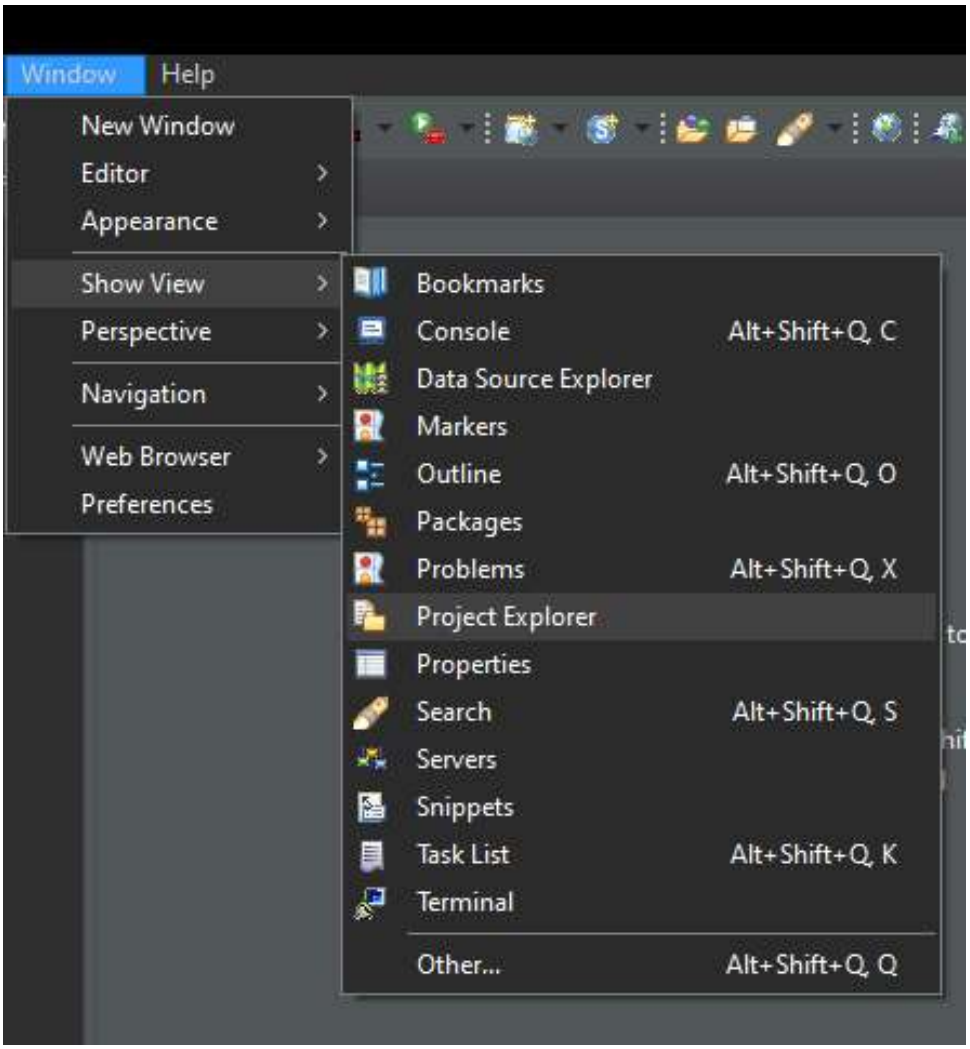
A continuación se abrirá una nueva ventana con el nombre **New Java Project**, en dónde inicialmente sólo colocaremos el nombre de nuestro proyecto. Para este ejemplo, utilizaremos el nombre **ProyectoEjemplo**.



Al final de la ventana, encontraremos el botón **Next >** donde haremos click.

La siguiente ventana nos muestra las configuraciones por defecto del proyecto que estamos generando. Por ahora no realizaremos más cambios y por último presionamos **Finish**.

Ahora volvimos a la ventana principal, donde deberías ver el explorador de proyectos del lado izquierdo. De no ser así, se habilita haciendo click en la barra de menú, en la pestaña **Window**, opción **Show View**, y por último **Project Explorer**. De esta manera tendrás en la vista todo el esqueleto de tu primer proyecto Java.

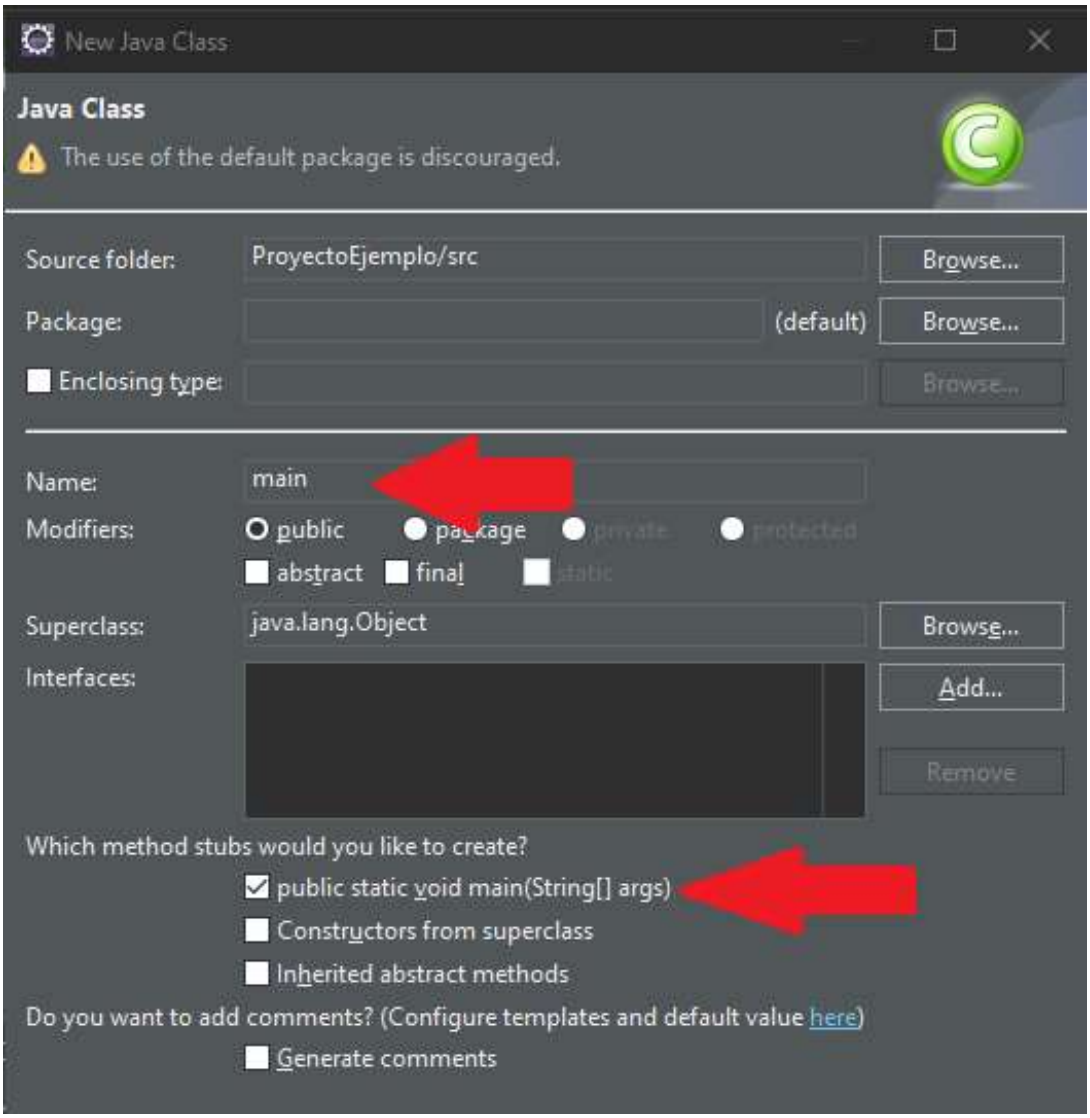


Ahora que puedes ver el esqueleto de tu proyecto del lado izquierdo, vamos a hacer click en la carpeta **“src”**, donde haremos click derecho para seleccionar la opción **New**, y luego la opción **Class.**

Se abrirá una nueva ventana de configuración, donde colocaremos el nombre de nuestro primer archivo ejecutable.

Para esto, primero debemos ponerle un nombre al archivo Class, en nuestro caso le colocaremos de nombre “**main**”.

Utilizamos la palabra main porque es el archivo donde estará el **método Main,** que es el punto de entrada de un programa ejecutable; es donde se inicia y finaliza el control del programa. Para que este método se encuentre en nuestra Class, vamos a tildar la opción **public static void main(String[] args),** y terminamos presionando Finish.



Ya tenemos nuestra primera clase ejecutable para comenzar a escribir código Java. Este debe escribirse entre las llaves de apertura y cierre, donde podemos ver el texto **//TODO Auto-generated method stub.**

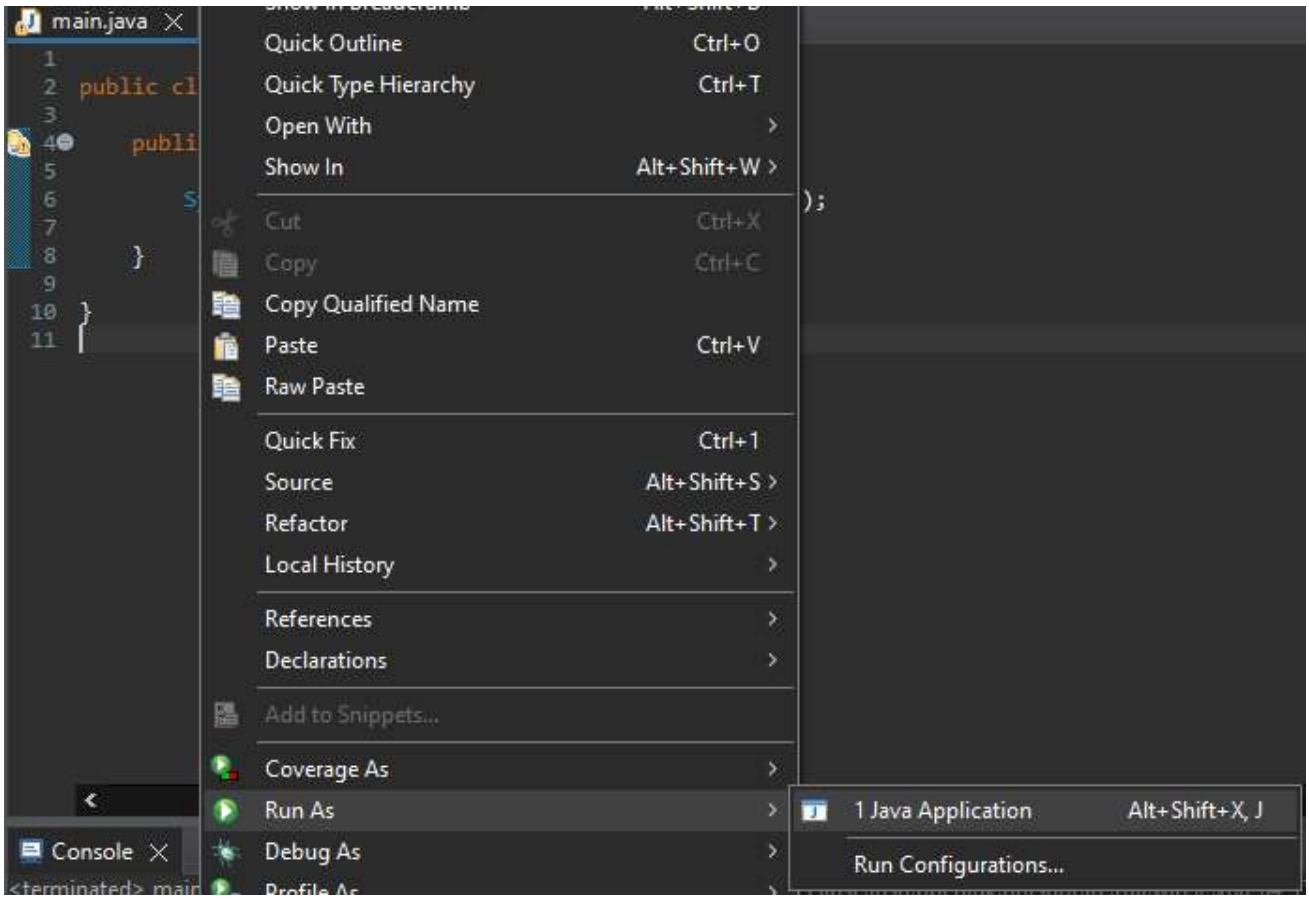
Este renglón se encuentra en color gris porque es un **“comentario”**. Los comentarios sirven para introducir líneas de texto o código de referencia que **no serán ejecutados** por el compilador. También se utiliza para escribir descripciones o documentar las funcionalidades de cada método.

En Java, los comentarios comienzan con dos barras seguidas (**//**).

```java
public class main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Para ejecutar un programa en Java, debemos hacer **click derecho** en cualquier parte del editor y seleccionar **Run As**, y luego **Java Application**.

También puedes presionar el botón con flecha verde que se encuentra en la parte superior del IDE, donde está la barra de herramientas.



Algoritmos en Java

Un algoritmo es un proceso o un **conjunto de reglas necesarias para realizar cálculos u otras operaciones** de resolución de problemas, especialmente por parte de una computadora.

La definición formal de un algoritmo es que **contiene un conjunto finito de instrucciones que se llevan a cabo en un orden específico para realizar una tarea específica.** No es el programa o código completo; es solo una solución (lógica) de un problema, que se puede representar de diversas maneras.

Se puede entender tomando el ejemplo de cocinar en base a una nueva receta. Para esto, se leen las instrucciones y los pasos y se ejecutan uno por uno, en la secuencia especificada. El resultado así obtenido es que el nuevo plato queda perfectamente cocinado. **Cada vez que usas tu teléfono, computadora, computadora portátil o calculadora, estás usando algoritmos**. De manera similar, los algoritmos ayudan a realizar una tarea específica en la programación para obtener el resultado esperado.

Los algoritmos diseñados son independientes del lenguaje, es decir, **son simplemente instrucciones que se pueden implementar en cualquier lenguaje** y, sin embargo, el resultado será el mismo.

Partes de un algoritmo

- **Entrada**: son todos los datos que se introducen al algoritmo para que pueda ejecutarse en base a dicha información.
- **Procesamiento**: con lo que ha recibido en la entrada, el algoritmo llevará a cabo una serie de cálculos para dar con la solución al problema.
- **Salida**: los resultados que se han obtenido del procesamiento de datos se mostrarán en la salida del algoritmo.

Características de un algoritmo

Para crear un algoritmo se deben tener en cuenta sus características:

- **Secuenciales**: se procesan consecutivamente. Responden a una secuencia de pasos determinados.
- **Precisos**: tienen que ser objetivos en la resolución del problema.
- Ordenados: deben leerse y ejecutarse en un orden específico.
- **Finitos**: deben contar con un número concreto de pasos.
- **Concretos**: deben mostrar una solución al problema especificado.
- **Definidos**: antes los mismos datos de entrada, siempre debe conseguirse los mismos datos de salida.

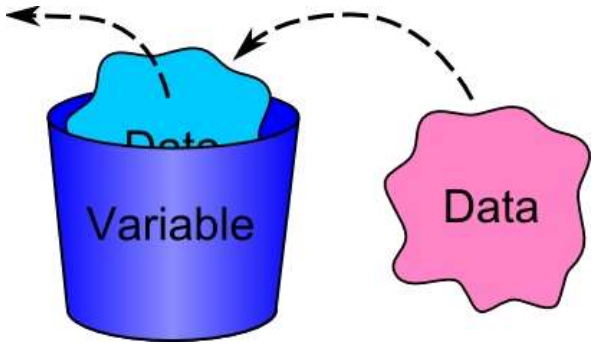
Variables en Java

Las variables son elementos fundamentales de la programación. **Actúan como contenedores que almacenan y manipulan diferentes tipos de datos.**
Nos permiten guardar información y acceder a ella en cualquier momento dentro de nuestro programa.

¿Qué es una variable?

Es un **espacio en la memoria de la computadora** que permite almacenar temporalmente un dato durante la ejecución de un proceso, su contenido puede cambiar durante la ejecución del programa.

Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con el cual podamos identificarla dentro de un algoritmo.



Por otro lado, una **constante** es un dato que **no cambia** durante la ejecución del programa.

Durante la escritura del código, usamos variables y constantes a los que asignamos valores. Las expresiones usarán las variables y constantes para construir cualquier tipo de tratamiento automatizado con ellas.

Características de las variables

- Deben comenzar con letra.
- No deben contener espacios en blanco.

- No deben contener meta caracteres.
- No se podrán utilizar palabras reservadas como nombres de variable.
- Las variables deben ser únicas e irrepetibles.

Por ejemplo, si tenemos una variables donde queremos guardar un tipo de dato numérico, podríamos hacerlo de la siguiente manera:

sueldoBruto = 1500

sueldo_Bruto = 1500

Ambos nombres de variable (**identificadores**) son válidos, ya que no tienen caracteres especiales ni espacios en blanco y son lo suficientemente descriptivos por sí mismos para saber cuál es la información que guardan.

Java es un lenguaje de tipado estático, por lo cual **todas las variables deben ser declaradas antes de ser utilizadas.** Estas tendrán un tipo de dato definido (ya sea un tipo de dato primitivo o una clase) y un nombre de identificador. El tipo de dato se asignará a la hora de definir la variable.

```
int numero = 2;

String cadena = "Hola";

double decimal = 2.4;

boolean flag = true;
```

El operador ‘ = ’ funciona para asignarle un valor a cada variable. Esto se conoce como inicialización. En la mayoría de los casos es necesario inicializar la variable al momento de crearla, pero no es obligatorio.

Variables locales y variables globales

Una **variable local** se declara dentro de un bloque de código específico, como una función o un bucle. **Su alcance está limitado a ese bloque en particular,** lo que significa que solo puede ser accedida y utilizada dentro de ese contexto. Fuera de ese bloque, la variable local no existe y no se puede acceder a su valor.

Por otro lado, una **variable global** se declara fuera de cualquier bloque de código, generalmente al comienzo del programa. **Su alcance se extiende a lo largo de todo el programa**, lo que significa que se puede acceder a ella y utilizarla desde cualquier parte del código. Es visible y accesible tanto dentro de funciones como fuera de ellas.

La diferencia clave entre las variables locales y globales radica en su alcance y visibilidad. Las variables locales son útiles cuando se necesita un dato temporal o cuando se desea restringir el acceso a un valor específico a un bloque de código en particular. Por otro lado, las variables globales son útiles cuando se necesita un valor compartido y accesible en diferentes partes del programa.

Tipos de datos primitivos ⬇️

Un tipo de datos es un **conjunto de valores que tienen una característica en común y que responden a unas operaciones determinadas**.

En un sistema informático trabajamos con datos y los lenguajes de programación necesitan saber cuál es el tipo de ese dato, para saber los valores posibles que podrían tener y las cosas que se permiten hacer con ellos.

Por ejemplo, 2 es un número entero. Lo podré sumar, restar y realizar diversas operaciones matemáticas con otros números. Mientras que "Santiago" es una cadena de caracteres, que la podré comparar con otras cadenas, concatenar otras cadenas, etc.

En todos los lenguajes de programación encontramos una clasificación de tipos de datos siempre presente, **los tipos de datos simples y los tipos de datos compuestos.**

Tipos de datos simples

Los tipos de datos simples, también llamados **tipos de datos primitivos** o tipos de datos básicos, son aquellos que contienen un elemento único de un tipo de datos particular y no se pueden descomponer en varios datos independientes.

Aquí tienes una lista de tipos de datos simples comunes en la mayoría de los lenguajes de programación:

- Numérico
- Número Entero
- Número Real (con decimales)
- Caracter
- Cadena de caracteres (string)
- Booleano (verdadero o falso)
- Enumerado (un conjunto de valores limitado)

Entre todos los tipos de datos mencionados en el listado, los más comunes son los siguientes:

- **Datos numéricos:** Permiten representar valores de forma numérica, ya sean números enteros o números reales (con decimales). Los datos de tipo numérico permiten realizar operaciones aritméticas como la suma, resta, multiplicación, potencia, etc.

- **Datos lógicos:** Son aquellos que solo pueden tener **dos valores: verdadero (true) o falso (false)**.
  Llamamos a estos tipos de datos **"booleanos"** por la palabra en inglés "Boolean". Este tipo de datos se utiliza cuando se producen comparaciones entre valores de otros tipos o cuando queremos tomar decisiones con estructuras condicionales.

- **Datos de tipo cadena (string)**: Los datos de tipo cadena, también llamados "string" o datos alfanuméricos, consisten en una secuencia de caracteres, ya sean números, letras o signos diversos que aparecen en el teclado. Este tipo de dato es muy común en las aplicaciones, ya que nos permiten guardar nombres, direcciones, emails y contraseñas.
  Los valores de este tipo de dato **se representan generalmente encerrado entre comillas (**" esto es un string **").** Depende del lenguaje particular si se usan comillas simples, dobles o si es posible usar ambas.

Estos tipos de datos son predefinidos por el lenguaje. La biblioteca Java proporciona clases asociadas a estos tipos que proporcionan métodos que facilitan su manejo.

byte: como su propio nombre denota, emplea un solo byte (8 bits) de almacenamiento. Esto permite almacenar valores en el rango [-128 , 127].

short: usa el doble de almacenamiento que el anterior, lo cual hace posible representar cualquier valor en el rango [-32.768, 32.767].

int: emplea 4 bytes de almacenamiento y es el tipo de dato entero más empleado. El rango de valores que puede representar va de -2,147,483,648 a 2,147,483,647.

long: es el tipo entero de mayor tamaño, 8 bytes (64 bits), con un rango de valores desde -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.

float: conocido como tipo de precisión simple, emplea un total de 32 bits. Con este tipo de datos es posible representar números en el rango de 1.4E-45 a 3.4028235E38. Esto significa que puede almacenar números en el rango negativo y positivo, incluyendo valores decimales, pero con una precisión limitada debido a su naturaleza de precisión simple.

double: se utiliza para representar números de punto flotante de precisión doble. Sigue un esquema de almacenamiento similar al anterior, pero usando 64 bits en lugar de 32. Esto le permite representar valores en el rango de 4.9E-324 a 1.79769313486231570E308..

El tipo double ofrece un rango de representación más amplio y una mayor precisión en comparación con el tipo float. Sin embargo, también ocupa más espacio de memoria. Por lo tanto, es importante seleccionar el tipo de dato adecuado según las necesidades específicas de tu programa.

char: el tipo de dato caracter es un simple caracter unicode de 16 bits. Su valor mínimo es de '\u0000' (En entero es 0) y su valor máximo es de '\uffff' (en entero es 65,535). Los datos de tipo caracter en Java se escriben entre comillas simples, por ejemplo **'a'**, o también indicando su valor Unicode, por ejemplo '\u0061'.

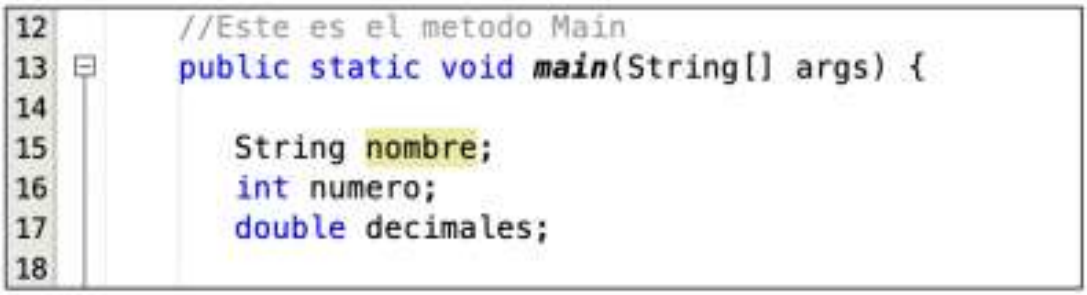
String: Además de los tipos de datos primitivos el lenguaje de programación Java provee también un soporte especial para cadenas de caracteres a través de la clase String.

Encerrando la cadena de caracteres con comillas dobles se creará de manera automática una nueva instancia de un objeto tipo String.

```
String cadena = "Hola";
```

Los objetos String son inmutables, esto significa que una vez creados, sus valores no pueden ser cambiados. Si bien esta clase no es técnicamente un tipo de dato primitivo, el lenguaje le da un soporte especial para que actúe como tal.

¿Cómo se ve en Java la declaración de una variable + tipo de dato?



Tipos de datos compuestos

Los tipos de datos compuestos, que también se denominan **tipos de datos complejos** o estructurados, se componen de **agrupaciones de tipos de datos simples**.

Algunos ejemplos de tipos de datos compuestos son:

- **Array:** es un conjunto de valores que **se almacenan en una misma referencia de la memoria** y donde todos sus valores son del mismo tipo simple.
- **Estructuras:** es la unión de varios tipos de datos simples, pero que pueden tener tipos distintos.

- **Objetos:** es un tipo en el que podemos tener distintos tipos de datos simples, pero donde además tenemos operaciones asociadas a esos objetos.

Estos tipos de datos complejos también dependen del lenguaje de programación que estemos utilizando.

Cierre ✍️

Java es un lenguaje de programación ampliamente utilizado en la industria del software. La evolución de Java a lo largo de sus versiones ha introducido nuevas características y mejoras significativas.

Eclipse IDE es un entorno de desarrollo integrado que facilita la escritura, compilación y depuración de código Java, ofreciendo herramientas y características adicionales para agilizar el proceso de desarrollo.

Con Eclipse IDE y las herramientas proporcionadas por el JDK, los desarrolladores pueden crear y ejecutar aplicaciones Java de manera eficiente y productiva.

También exploramos los fundamentos de la programación en Java, como declarar las variables y cómo utilizarlas para almacenar y manipular datos en nuestros programas.

Además, hemos descubierto los diferentes tipos de datos primitivos que nos brinda este lenguaje y que nos permiten trabajar con números, caracteres y valores booleanos.

Referencias 📚

- Amazon AWS | What is Java?
- Eclipse | Download Eclipse Technology that is right for you
- Javatpoint | How to use Eclipse for Java
- Javatpoint | Data Structure Algorithm
- Geeks for geeks | Introduction to algorithms

Muchas gracias

Nos vemos en la próxima unidad

alkemy
fomentar empleo
Ministerio de Trabajo, Empleo y Seguridad Social Argentina