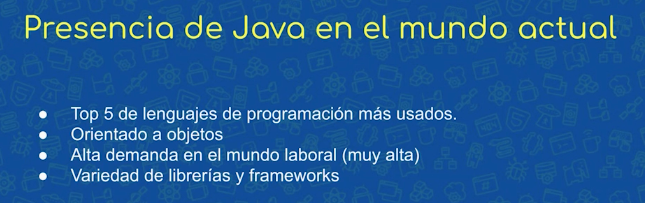
# Curso de Java JRE y JDK: compile y ejecute su primer programa

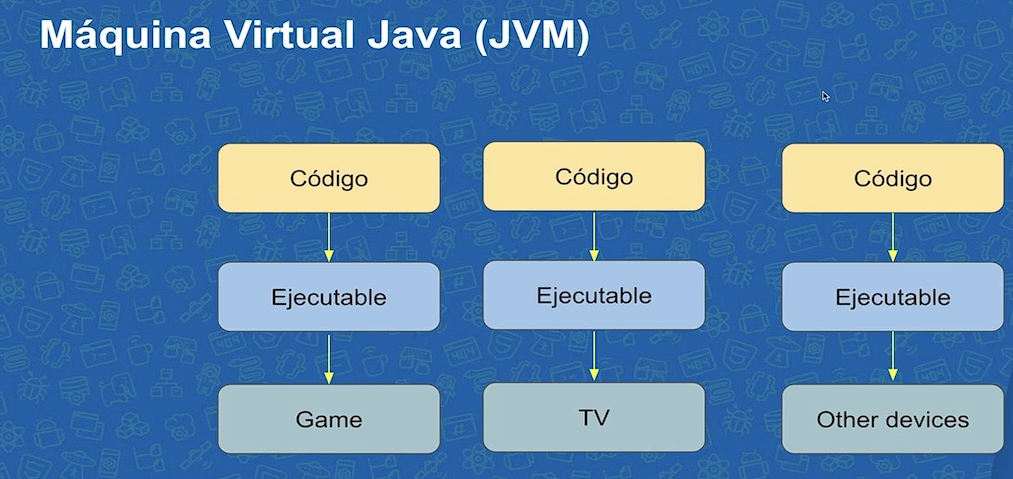
**Realice este curso para Java y:**

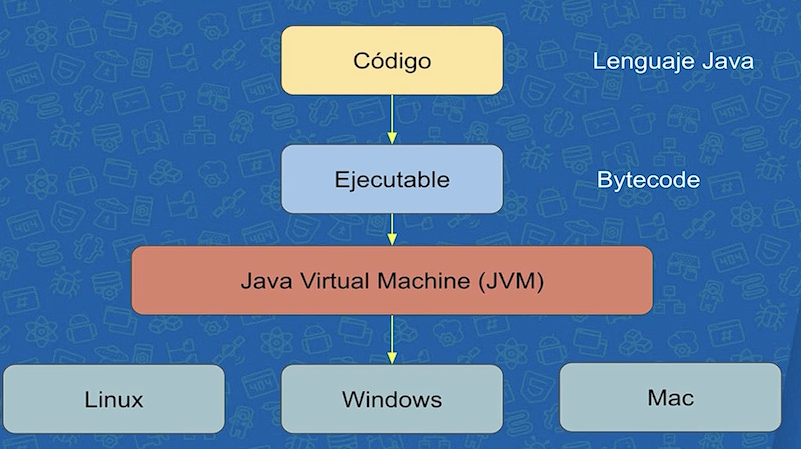
* ¿JVM? ¿JDK? ¿JRE? ¿Qué son esas siglas?
* Compilando y ejecutando código Java
* Aprenda a usar Eclipse
* Variables y control de flujo
* Conozca los principales tipos de Java

# Que es Java:











# Lo que aprendimos:

En la clase introductoria ya hemos aprendido varios temas fundamentales sobre Java.

Hablamos sobre las principales características de Java como:

* Orientado a objetos.
* Parecido a C++.
* Muchas librerías y una gran comunidad.

Además de eso, aprendimos:

* La diferencia entre código fuente y Bytecode.
* Para ejecutar el Bytecode necesitamos tener la máquina virtual de Java.
* El Bytecode es independiente del sistema operativo.

Vimos también los principales componentes de la plataforma Java, que son:

* Java Virtual Machine (JVM).
* Lenguaje Java.
* Librerias Java (API).

# 

# El primer Programa:

El mundo Java está lleno de siglas de 3 o 4 letras comenzando con la letra J. Ya debes conocer las dos más famosas, JRE y JDK. El primero es el ambiente de ejecución, el segundo son las herramientas de desarrollo *junto* con el ambiente de ejecución. Simplificando, podemos decir que:

JDK = JRE + Herramientas de desarrolloCOPIA EL CÓDIGO

Existe una tercera sigla, JVM (Java Virtual Machine), que también hemos usado durante el curso. La responsabilidad de la Java Virtual Machine es ejecutar el Bytecode, entonces ¿Cuál es la diferencia entre el JRE y la JVM si ambos ejecutan Bytecode?

La respuesta es simple: El JRE (Nuestro ambiente de ejecución) contiene la JVM, pero también posee varias librerías incluidas. Es decir, para ejecutar una aplicación Java no solo es necesario tener la JVM, también necesitamos de otras librerías.

Entonces, podemos simplificar:

JRE = JVM + LibreríasCOPIA EL CÓDIGO

Es importante resaltar que no se puede descargar la JVM individualmente. Siempre tendremos que descargar el JRE que tiene la JVM y el conjunto de librerías.

# 

Java es un lenguaje de programación que se actualiza periódicamente por Oracle, la empresa responsable de su desarrollo. Cada nueva versión de Java trae consigo nuevas características, mejoras de rendimiento, correcciones de errores y actualizaciones de seguridad. Estas versiones se numeran, siguiendo un patrón específico.

Cuando se lanza una nueva versión, puede incluir nuevas bibliotecas, clases, métodos y otros recursos que los desarrolladores pueden utilizar para crear aplicaciones Java más eficientes y con menos errores.

Aquí hay algunos ejemplos de algunas de las principales versiones de Java y sus características:

**Java 8**: Introdujo la programación funcional, incluyendo la interfaz java.util.function, que permite el uso de expresiones lambda. Además, se agregó una nueva API de fecha y hora que proporciona una forma más simple y segura de manejar fechas y horas.

**Java 11**: Introdujo el sistema de módulos de Java, que ayuda a simplificar la creación y mantenimiento de aplicaciones complejas. Además, se agregó la clase HttpClient, que admite comunicaciones HTTP/2.

**Java 15**: Agregó características como la palabra clave sealed, que permite que las clases restrinjan qué otras clases pueden extenderlas o implementarlas, y también agregó mejoras a la API Records, que ayuda a simplificar la creación de clases de datos inmutables.

**Java 17**: introduce nuevas características y mejoras, como patrones de coincidencia que mejoran la sintaxis al trabajar con estructuras de datos complejas. Además, se mejoran el rendimiento del recolector de basura para reducir la latencia en las aplicaciones Java. También se agregan funcionalidades a los registros, que son clases inmutables y compactas utilizadas para representar datos, incluyendo la capacidad de definir registros locales dentro de métodos. Otra adición importante son las nuevas clases y métodos en el paquete java.util para trabajar con estructuras de datos persistentes, lo que permite realizar cambios en los datos sin modificar las estructuras originales. Por último, se agrega soporte para CGroups en la API de Java, lo que permite una mejor administración de recursos en entornos de contenedores.

Al actualizar a una nueva versión de Java, es importante tener en cuenta la compatibilidad con versiones anteriores. A veces, se eliminan o modifican características o funcionalidades en una nueva versión, lo que puede afectar el código existente. Por esta razón, es importante probar su código al actualizar a una nueva versión de Java.

Además, es posible que coexistan diferentes versiones de Java en un sistema, lo que permite que las aplicaciones se ejecuten en versiones específicas de la JVM (Java Virtual Machine) para garantizar la compatibilidad con el código existente.

# Lo que aprendimos:

En esta clase escribiste tu primer código Java y aprendimos:

* Cuál es la diferencia entre JRE y JDK.
* Cómo compilar el código fuente de Java desde la línea de comandos (javac).
* Cómo ejecutar Bytecode en la línea de comando (java).
* Un programa Java debe escribirse dentro de una clase (class).
* Cada instrucción Java debe terminar con ;.
* Para abrir y cerrar un bloque usaremos las llaves {}.
* Un programa Java tiene una entrada que es una función (método) main.
* Para imprimir algo en la consola, usamos la declaración System.out.println ().

# Programando con Eclipse:

# 

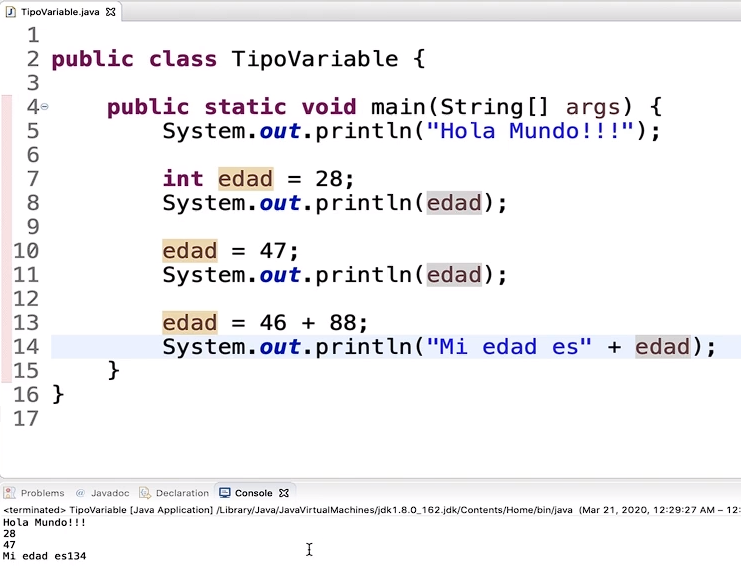
# Lo que aprendimos:

Este capítulo presentó:

* El papel de un IDE y su diferencia con respecto a un editor.
* Cómo descargar el IDE de Eclipse.
* ¿Para qué sirve workspace?
* El concepto de perspectiva.
* Cómo crear un proyecto Java, incluidas las clases y cómo ejecutarlo.
* Cómo mostrar diferentes views.

# Tipos y Variables:

# Tipo Int:



# Tipo Double:

# 

# Conversiones:

# 

# Lo que aprendimos:

n esta clase comenzamos nuestro aprendizaje con variables y tipos primitivos de Java. Los tipos vistos con más detalle fueron int (entero) y double (decimal). Los cuales usamos para hacer operaciones aritméticas y también concatenar con texto.

Durante el capítulo hablamos de buenas prácticas al nombrar clases y también variables. Comenzamos las clases con mayúscula y nuestras funciones y variables con minúsculas. Hablaremos más sobre esto en el futuro.

Fue posible comprender un poco de conversión de tipos y cómo podemos pasar un valor de un tipo para una variable de otro. En algunos casos no necesitamos hacer nada, ya que el casting es implícito y en otros debemos dejar en claro al compilador que sabemos lo que estamos haciendo, mostrando entre paréntesis el tipo que queremos usar.

# Trabajando con caracteres:

# Caracteres y string:

# 

# Variables y memoria:

# Lo que aprendimos:

Este capítulo presentó:

* El concepto y cómo declarar char y String.
* Cómo concatenar Strings.
* Atajo para crear método main.
* Las variables almacenan valores y no referencias.

# Practicando Condicionales:

# Test if:

# 

# Boolean:

# 

# Scope e inicializacion:

# 

# Lo que aprendimos:

En esta clase, aprendimos:

* Cómo usar el if.
* Cómo usar las operaciones lógicas AND (&&) y OR (||).
* Trabajar con el alcance de las variables.

También hemos visto algunos atajos en Eclipse:

* main Ctrl + espacio, para generar el método main.
* ctrl + shift + f, para formatear el código fuente.
* sysout + ctrl + espacio, para generar la instrucción System.out.println ().

# Controlando Flujo con Loops:

# Ciclo while:

# 

# Scope Ciclos:

# 

# Ciclo for:

# 

# Ciclos anidados:

# 

# Ciclos break:

# 

# Lo que aprendimos:

En este capítulo aprendimos:

* La sintaxis de while y for.
* El operador +=.
* El operador ++.
* Bucles anidados.
* La funcionalidad Break.