UT 1: Introducción. Elementos de un programa informático.

1. Introducción

La razón principal por la que una persona utiliza un ordenador es para resolver problemas, o en otras palabras, procesar una información para obtener un resultado a partir de unos datos de entrada.

Los ordenadores resuelven los problemas mediante la utilización de programas escritos por los programadores. Los programas de ordenador no son entonces más que métodos para resolver problemas. Por ello, para escribir un programa, lo primero es que el programador sepa resolver el problema que estamos tratando.

El programador debe identificar cuáles son los datos de entrada y a partir de ellos obtener los datos de salida, es decir, la solución, a la que se llegará por medio del procesamiento de la información que se realizará mediante la utilización de un método para resolver el problema que denominaremos algoritmo.

2. Algoritmos

Por algoritmo entendemos un conjunto ordenado y finito de operaciones que permiten resolver un problema.

Los algoritmos son independientes de los lenguajes de programación y de las computadoras donde se ejecutan. Ejemplo: una receta cocina.

2.1. Representación de un algoritmo

Para representar un algoritmo se utilizan:

- **Diagrama de flujo**: Esta técnica utiliza símbolos gráficos para la representación del algoritmo. Suele utilizarse en las fases de análisis.
- **Pseudocódigo**: Esta técnica se basa en el uso de palabras clave en lenguaje natural, constantes.
- **Tablas de decisión**: En una tabla son representadas las posibles condiciones del problema con sus respectivas acciones.

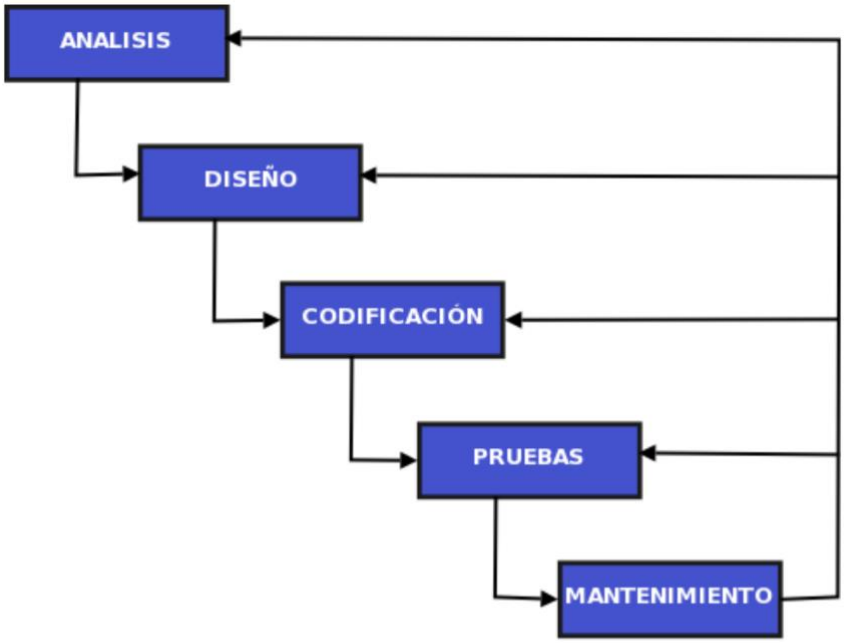
3. Ciclo de vida del software

Ciclo de vida del software: es una sucesión de estados o fases por las cuales pasa un software a lo largo de su "vida".

Normalmente, el ciclo de vida se divide en:

- Fase de resolución del problema.
    - Análisis
    - Diseño

- Fase de implementación.
    - Codificación
    - Pruebas
- Fase de explotación y mantenimiento



Cido de vida ideal en el desarrollo de una aplicación.

4. Lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos y reglas, que pueden ser combinados para expresar algoritmos.

El primer lenguaje de programación que se desarrolla es el lenguaje máquina, compuesto por unos y ceros. Aunque el ordenador es capaz de procesarlo sin problema, es muy tedioso para los seres humanos.

Dado que el cualquier ordenador o dispositivo electrónico solo es capaz de entender lenguaje máquina y los lenguajes de programación se han diseñado para que los utilicen personas, se debe realizar una traducción del lenguaje de programación al código máquina: compilación.

4.1. Compiladores e intérpretes

El compilador es el encargado de traducir un programa escrito en un lenguaje de programación a un programa objeto, en lenguaje o máquina o ensamblador (evolución del lenguaje máquina, aunque sigue siendo tedioso para las personas). Una vez

traducido el programa, su ejecución es independiente del compilador, lo que implica que la interacción del usuario con el mismo se realiza a través del SO.

El intérprete lo que hace es traducir cada sentencia del programa y lo ejecuta. El interprete capta una sentencia del fichero fuente, la analiza e interpreta dando lugar a una ejecución inmediata, sin crearse un archivo o programa objeto. Aunque es algo más lento que un compilador, permite la ejecución línea a línea para poder detectar errores fácilmente.

4.2. Tipos de lenguajes de programación

- **Según su nivel**, es decir, proximidad entre el lenguaje de programación y el lenguaje máquina:
    - Lenguajes de bajo nivel: son lenguajes de programación muy parecidos al lenguaje máquina. Son difíciles de usar por el ser humano. Los de ejemplos más característicos son el lenguaje máquina y ensamblador.
    - Lenguajes de alto nivel: son lenguajes que tienen una sintaxis y símbolos parecidos al lenguaje humano. Abstraen al programador del funcionamiento interno de la máquina. Cada vez hay más lenguajes de alto nivel: Java, C++, C, etc.
- **Según la forma de ejecución**:
    - Compilados
    - Interpretados
- **Según el estilo de programación**:
    - Lenguajes de programación estructurados: los programas se organizan en mediante un conjunto de funciones. Hay una función principal que se invoca en el momento de la ejecución y esta irá haciendo uso del resto de funciones, según lo decida el programador.
    - Lenguajes de programación orientados a objetos: se definen una serie de elementos, denominados clases, que pretenden reflejar la realidad que rodea al software. Las clases disponen de una serie de métodos y atributos. Para poder ejecutar una clase, hay que instanciarla, dando lugar a un objeto.

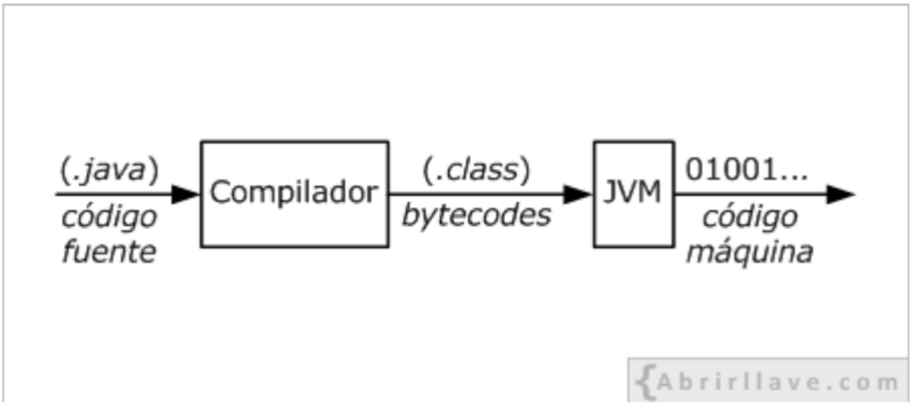
4.3. Java

Fue desarrollado por *Sun Microsystems* en 1995, aunque ya se había presentado como lenguaje de programación de componentes electrónicos en 1991.

Aunque su uso inicial era para la electrónica de consumo, comienza a extenderse debido al auge de la programación web con el desarrollo de los applets, que son pequeños programas que pueden ser interpretados por navegadores web.

Se trata de un lenguaje basado en programación orientada a objetos (POO), que requiere de un intérprete para ejecutarse: JVM o Máquina Virtual de Java.

La JVM compila el código fuente, se resuelven los errores y se genera un código intermedio, denominado *bytecode* (ficheros .class), que puede ser ejecutado en una máquina virtual y además son portables. La máquina virtual dispone de un intérprete que traduce los .class en cada ordenador concreto.



JRE es el *Java Runtime Environment* o, en español, el Entorno de Ejecución de Java, que contiene las herramientas necesarias para ejecutar programas escritos en Java. Dicho de otra forma, el JRE incluye la JVM.

Para comenzar a programar, se suele utilizar un SDK (*Software Development Kit*). En el caso concreto de Java se utiliza el JDK (*Java Development Kit*) que es un subconjunto del SDK.

Si necesitas desarrollar en Java debes instalar el JDK, sin embargo, si solo necesitas ejecutar, con el JRE es suficiente.

https://www.oracle.com/java/technologies/downloads/#jdk17-windows

Durante o después de la instalación habría que tener en cuenta 3 variables del sistema:

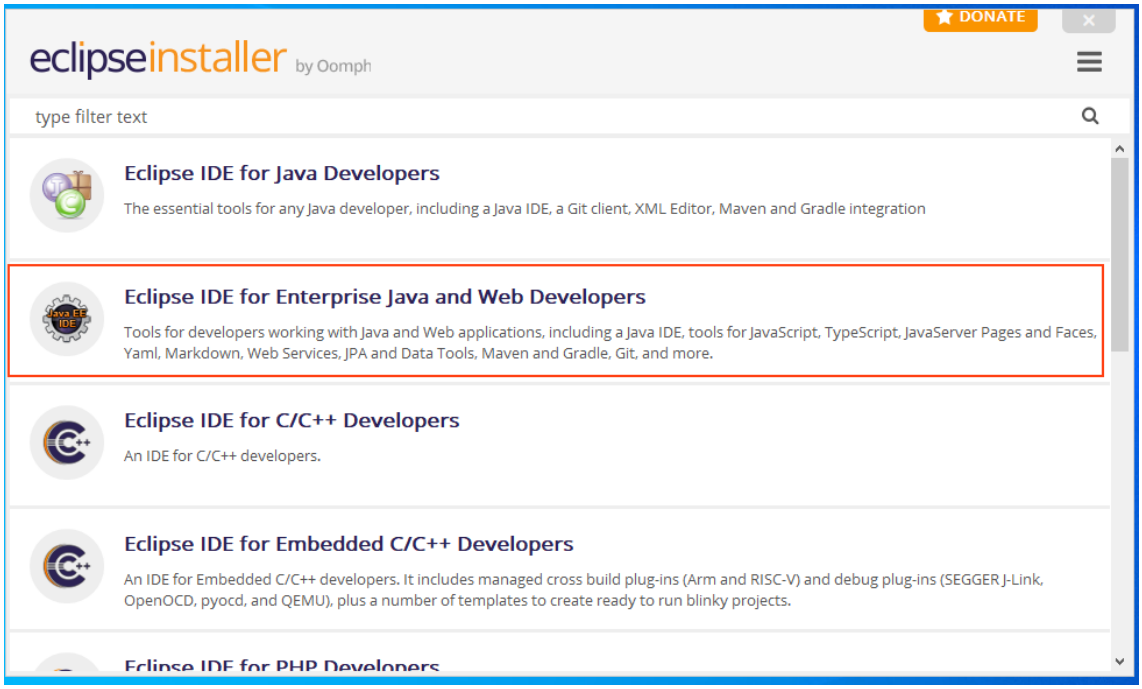
- **PATH**: variable que contiene rutas por defecto a los programas que indiquemos. La razón principal del uso de esta variable es que el comando java debe de estar disponible estemos en la carpeta que estemos. Dicho comando (junto con el resto de comandos del SDK) está en la carpeta bin dentro de la carpeta en la que hemos instalado el SDK.

- **JAVA_HOME**: variable utilizada por la mayoría de aplicaciones basadas en Java que contiene la ruta a la carpeta en la que se instaló el SDK.
- **CLASSPATH**: es una variable similar al PATH que sirve para indicar rutas a las carpetas en las que se almacenarán aplicaciones Java. Dicho de una manera más técnica: contiene las rutas de todos los *filesystems* de Java.

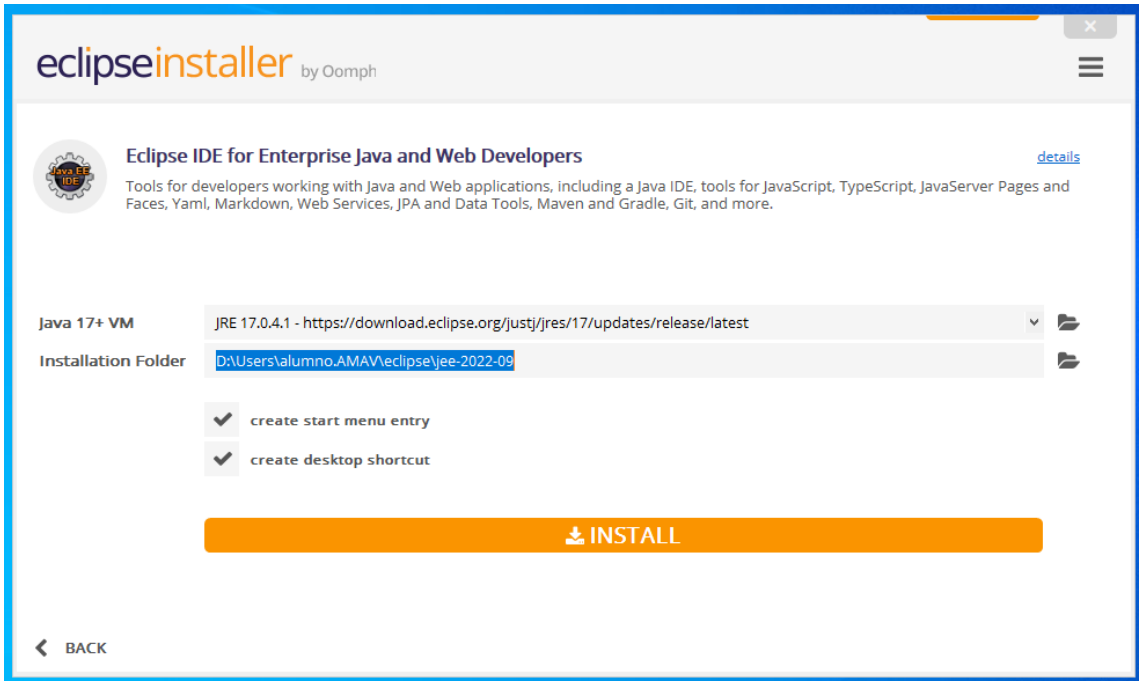
Hoy en día la mayoría de programadores utilizan IDEs o Entornos de Desarrollo Integrados, que son programas que agrupan varias de las herramientas vistas anteriormente. Suelen disponer de un editor de código, un compilador, un depurador, etc.

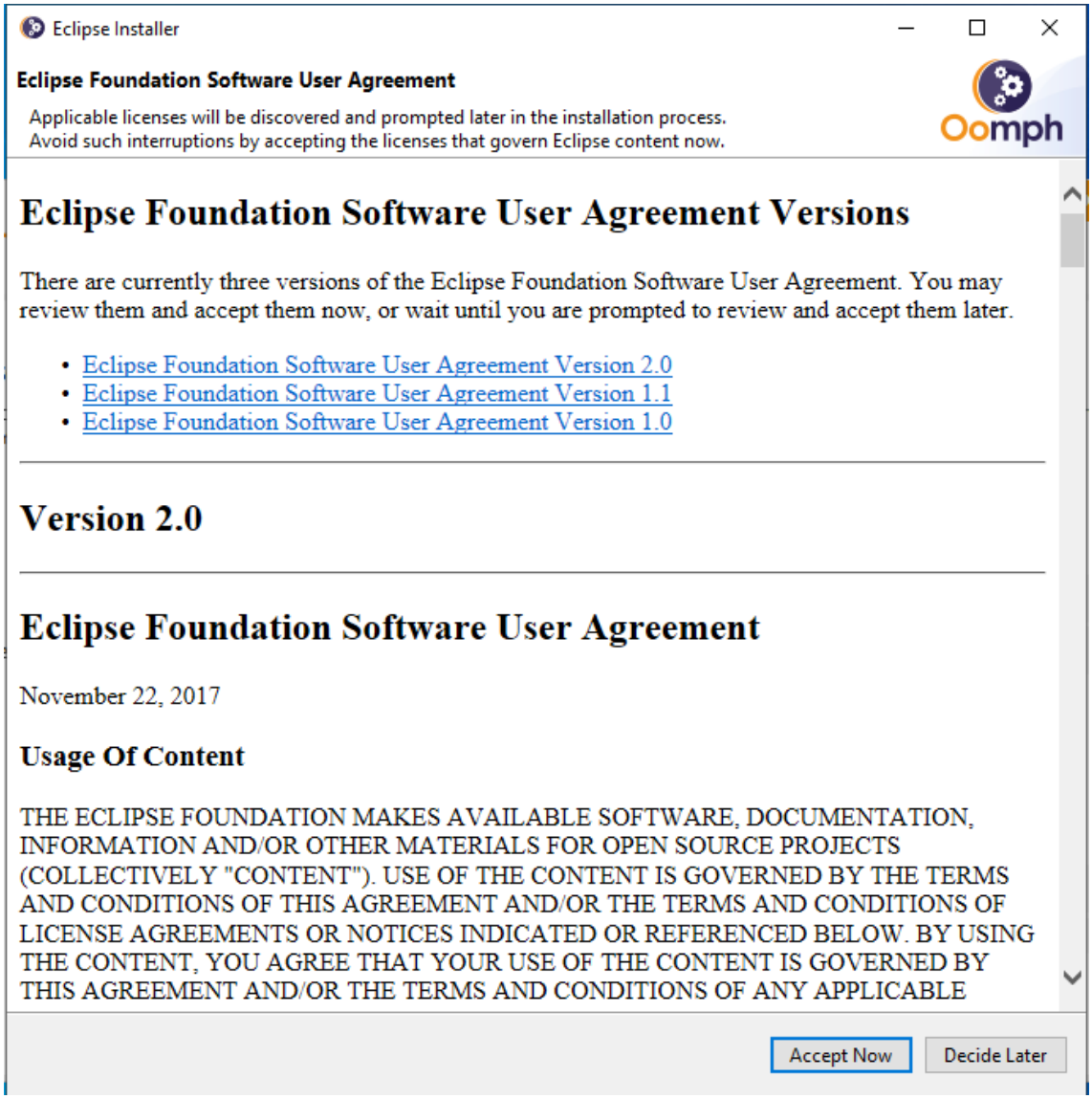
5. Instalación de Eclipse

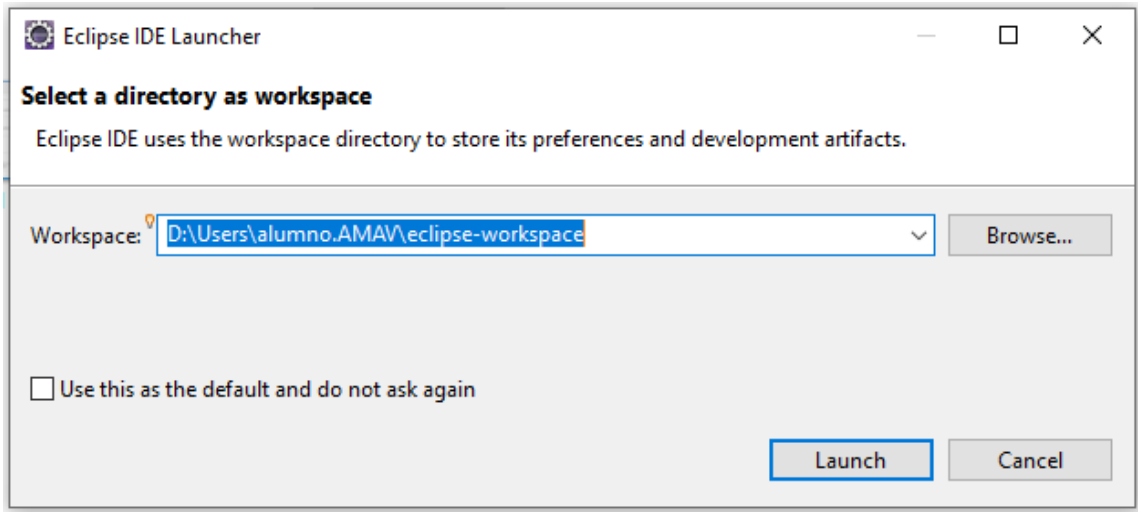
https://www.eclipse.org/downloads/



Cualquiera de las dos primeras opciones es válida aunque la segunda tiene lo mismo que la primera y opciones de desarrollo WEB, con lo cual, la segunda opción sería la mejor.


eclipseinstaller by Oomph
Eclipse IDE for Enterprise Java and Web Developers
details
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data Tools, Maven and Gradle, Git, and more.
Java 17+ VM
JRE 17.0.4.1 - https://download.eclipse.org/justj/jres/17/updates/release/latest
Installation Folder
D:\Users\alumno.AMAV\eclipse\jee-2022-09
create start menu entry
create desktop shortcut
INSTALL
BACK

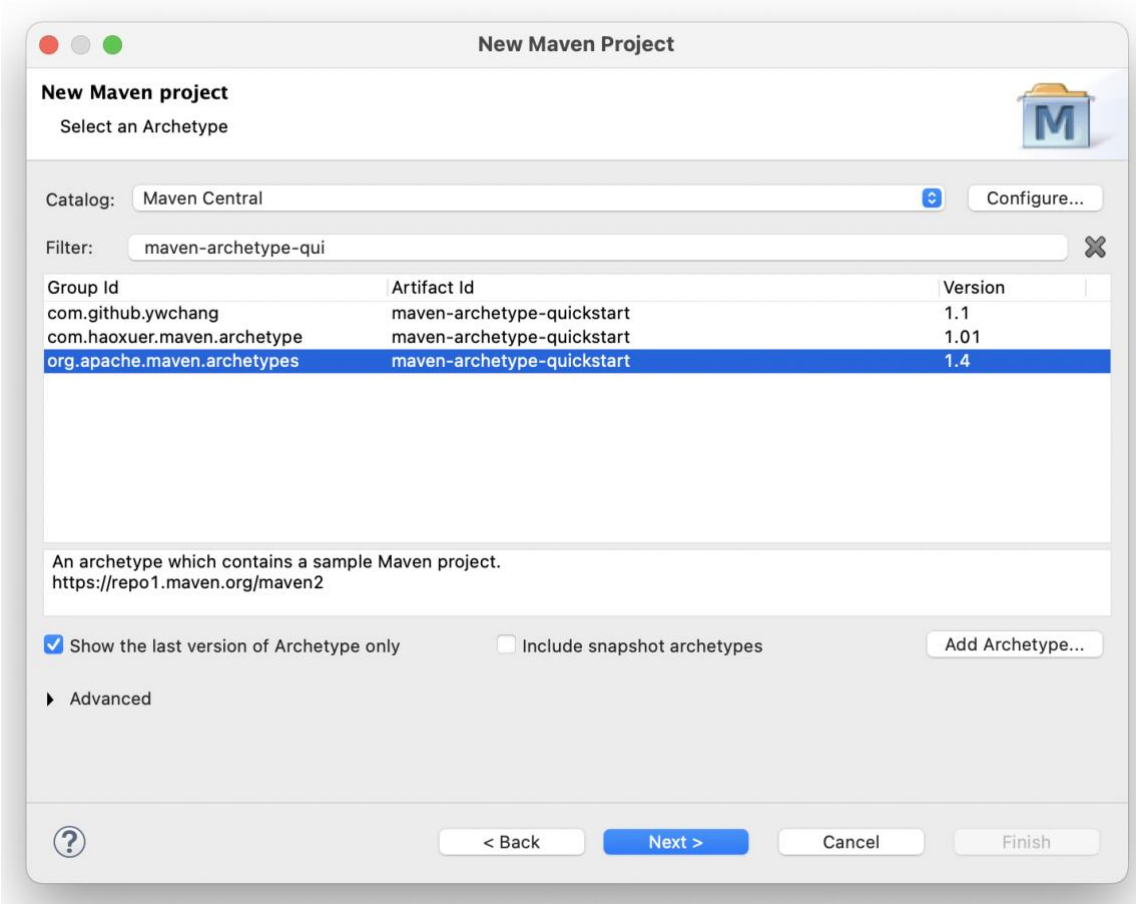

Eclipse Installer
Eclipse Foundation Software User Agreement
Applicable licenses will be discovered and prompted later in the installation process.
Avoid such interruptions by accepting the licenses that govern Eclipse content now.
Oomph
Eclipse Foundation Software User Agreement Versions
There are currently three versions of the Eclipse Foundation Software User Agreement. You may review them and accept them now, or wait until you are prompted to review and accept them later.
Eclipse Foundation Software User Agreement Version 2.0
Eclipse Foundation Software User Agreement Version 1.1
Eclipse Foundation Software User Agreement Version 1.0
Version 2.0
Eclipse Foundation Software User Agreement
November 22, 2017
Usage Of Content
THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE
Accept Now
Decide Later

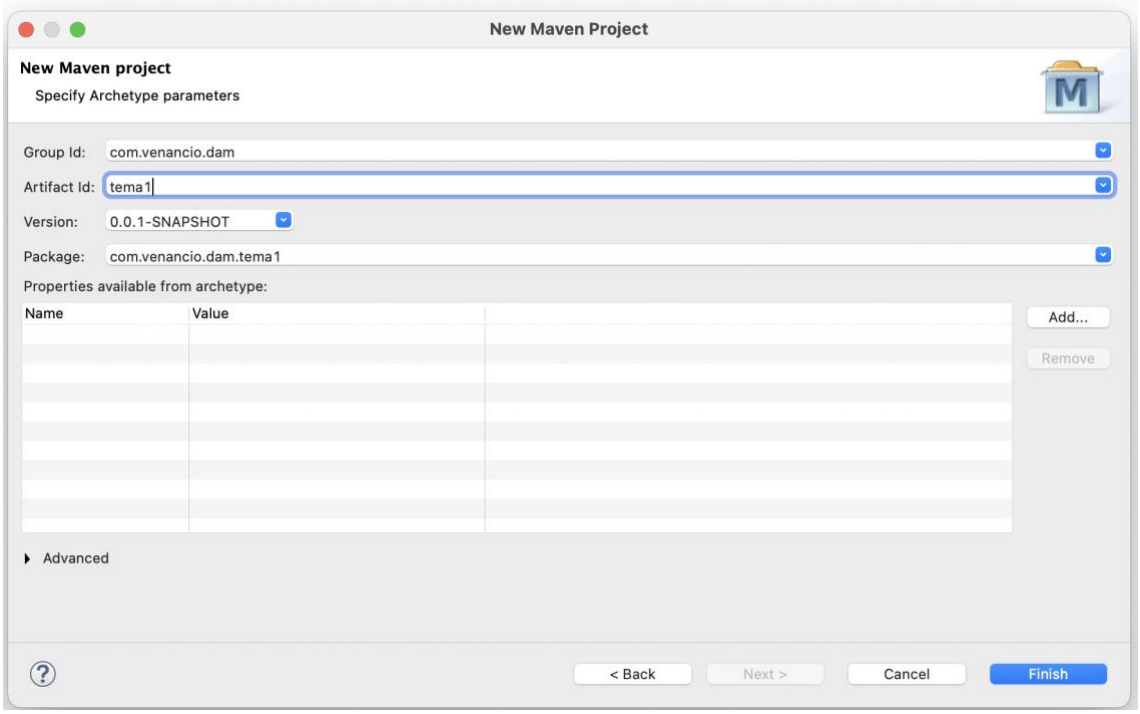


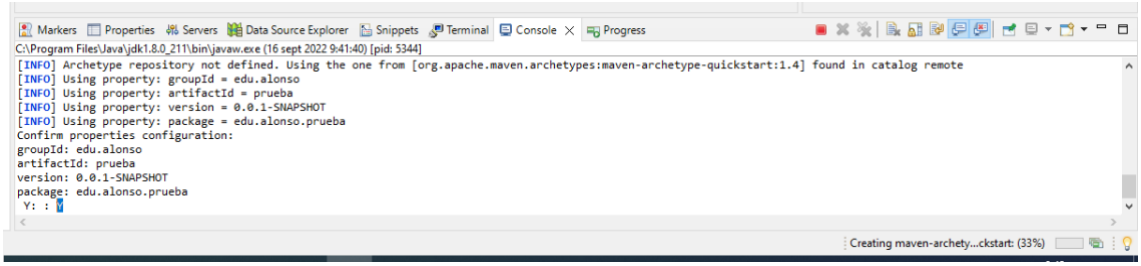
Ojo, **NUNCA** seleccionar la opción "*Use this as the default and do not ask again*". Al no seleccionarlo, cada vez que se abra Eclipse, nos dará la opción de seleccionar el *workspace* que consideremos oportuno.

5.1. Proyecto Maven

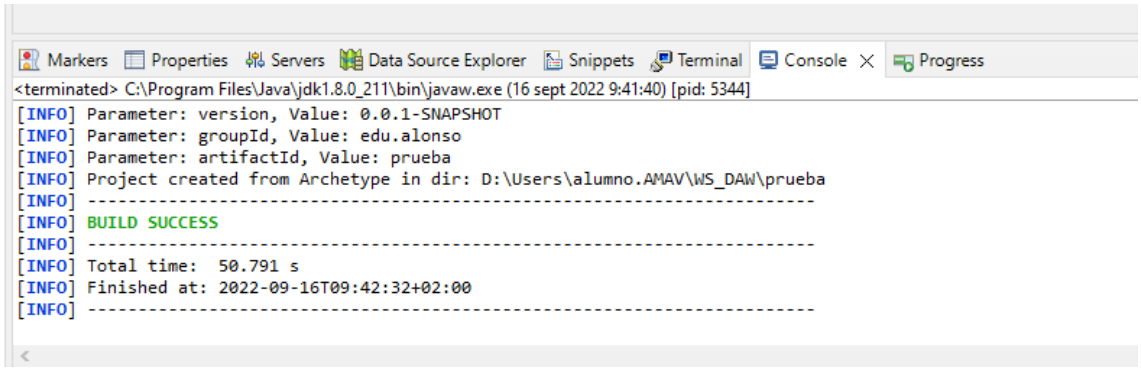
Para crear un proyecto desde Eclipse vamos a File->New->Maven Project. Podríamos indicar donde se guarda y definimos el tipo básico de proyecto:





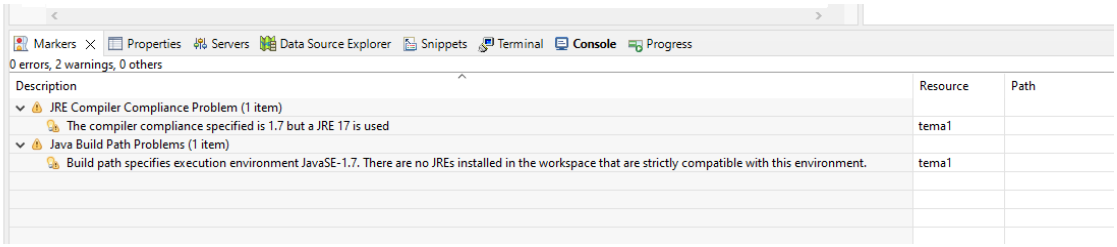


Hay que indicarle que continúe con el proceso (“Y”).

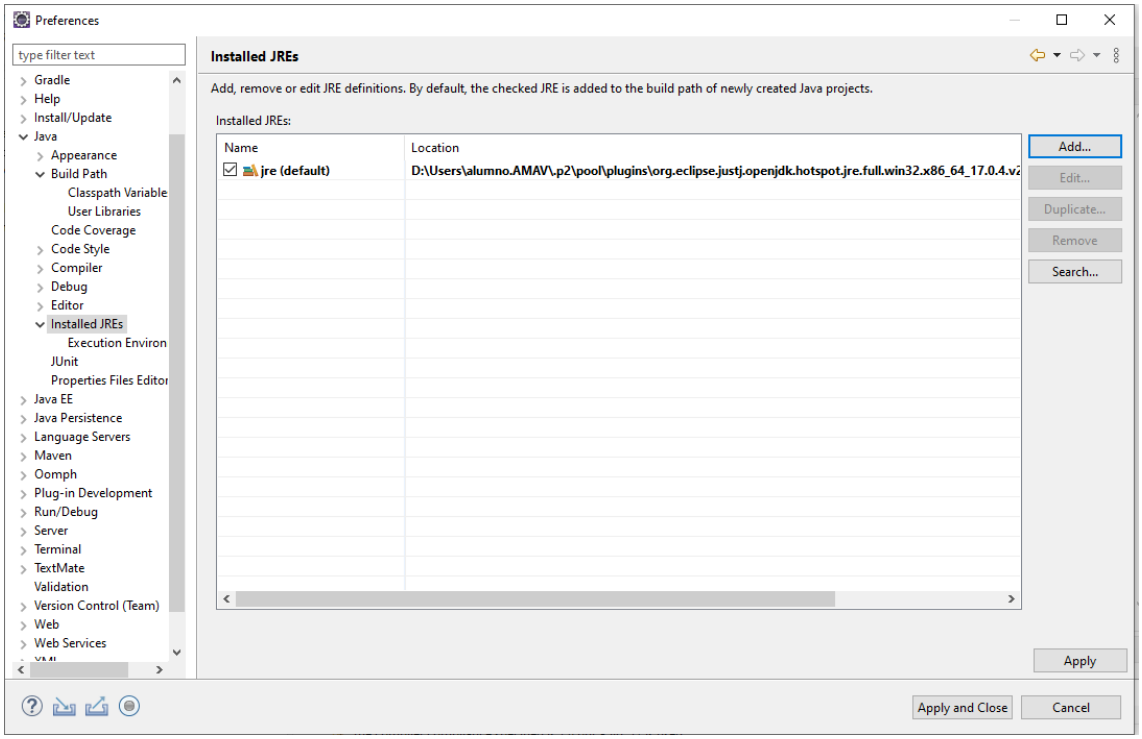


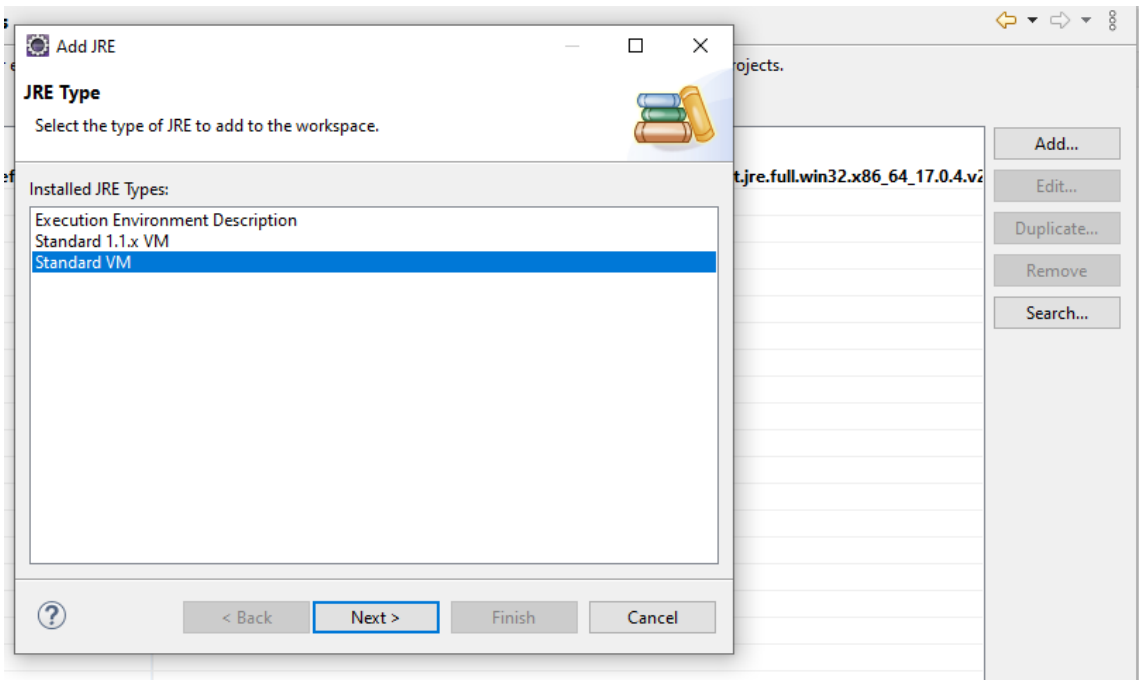
5.2. Errores comunes de Eclipse

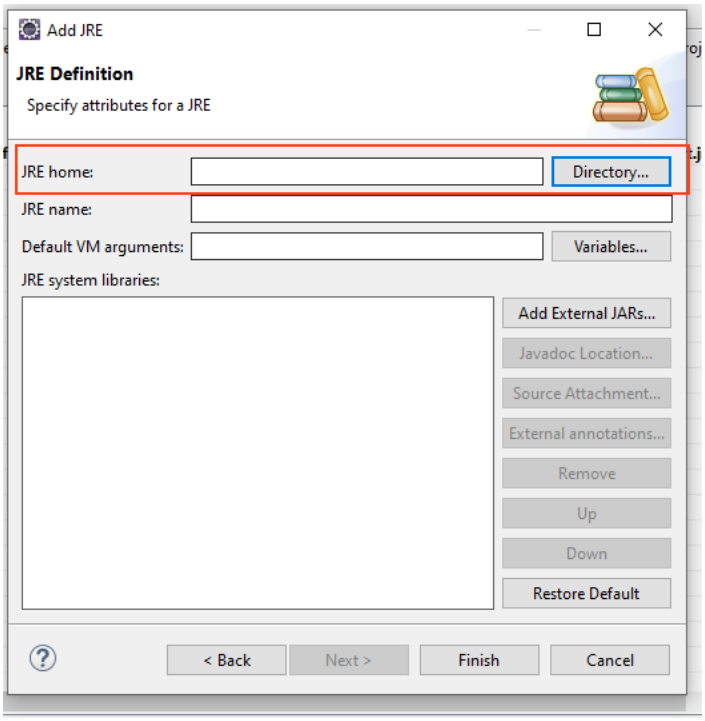
Uno de los problemas más comunes es la incompatibilidad de la versión de compilación de java.



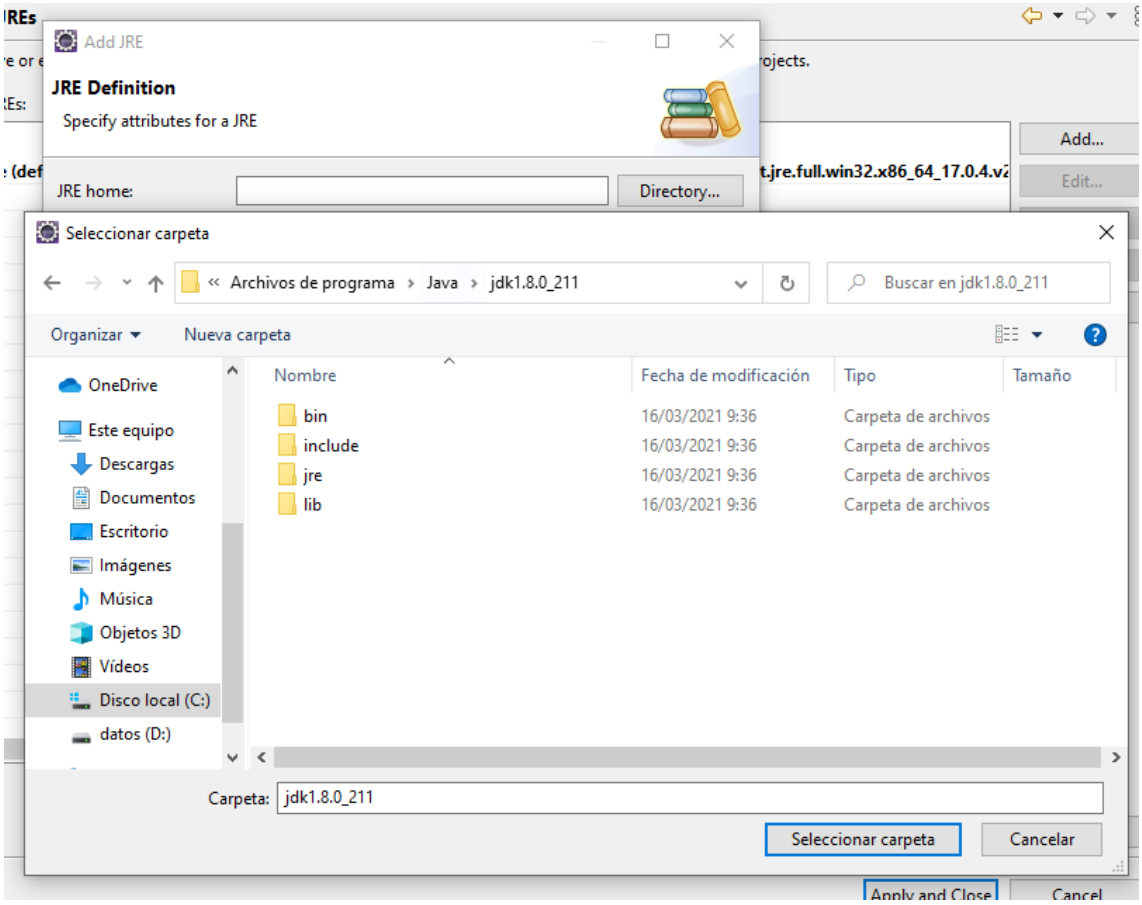
Hay que cambiar la versión del compilador que viene instalada por defecto.

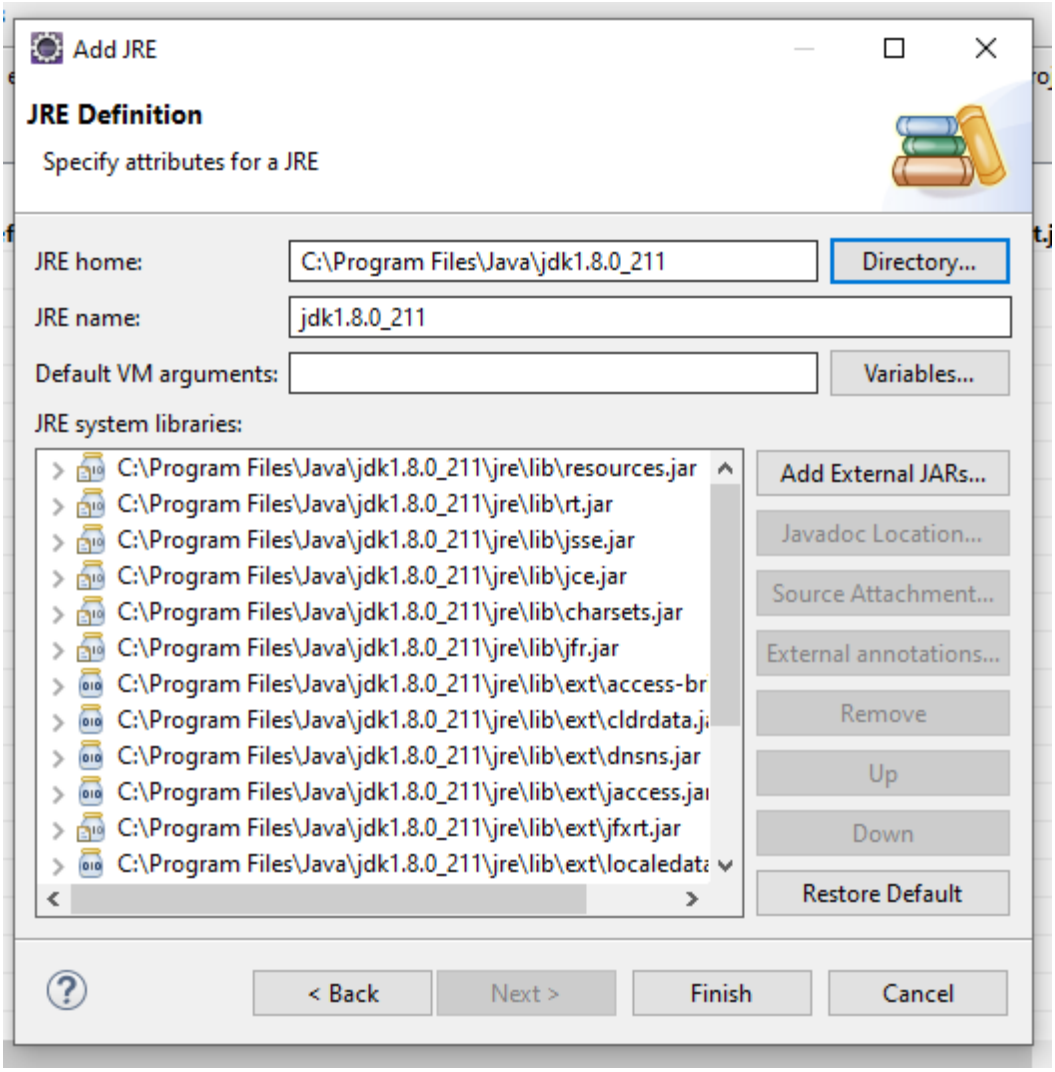


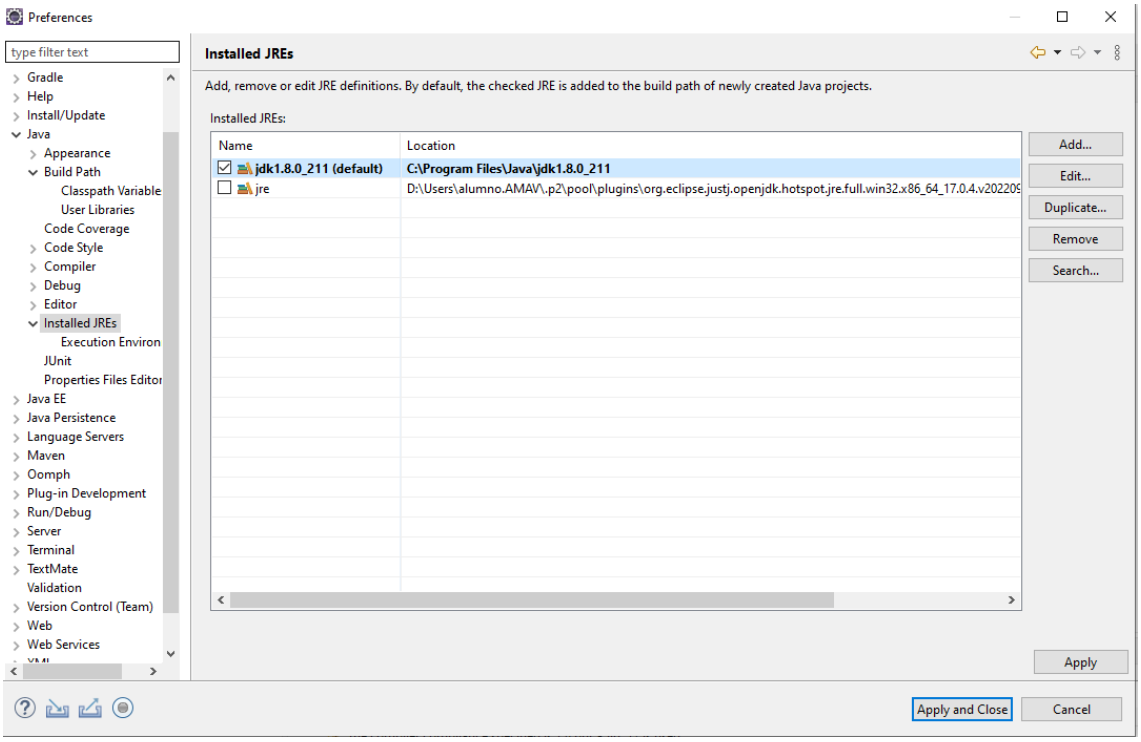




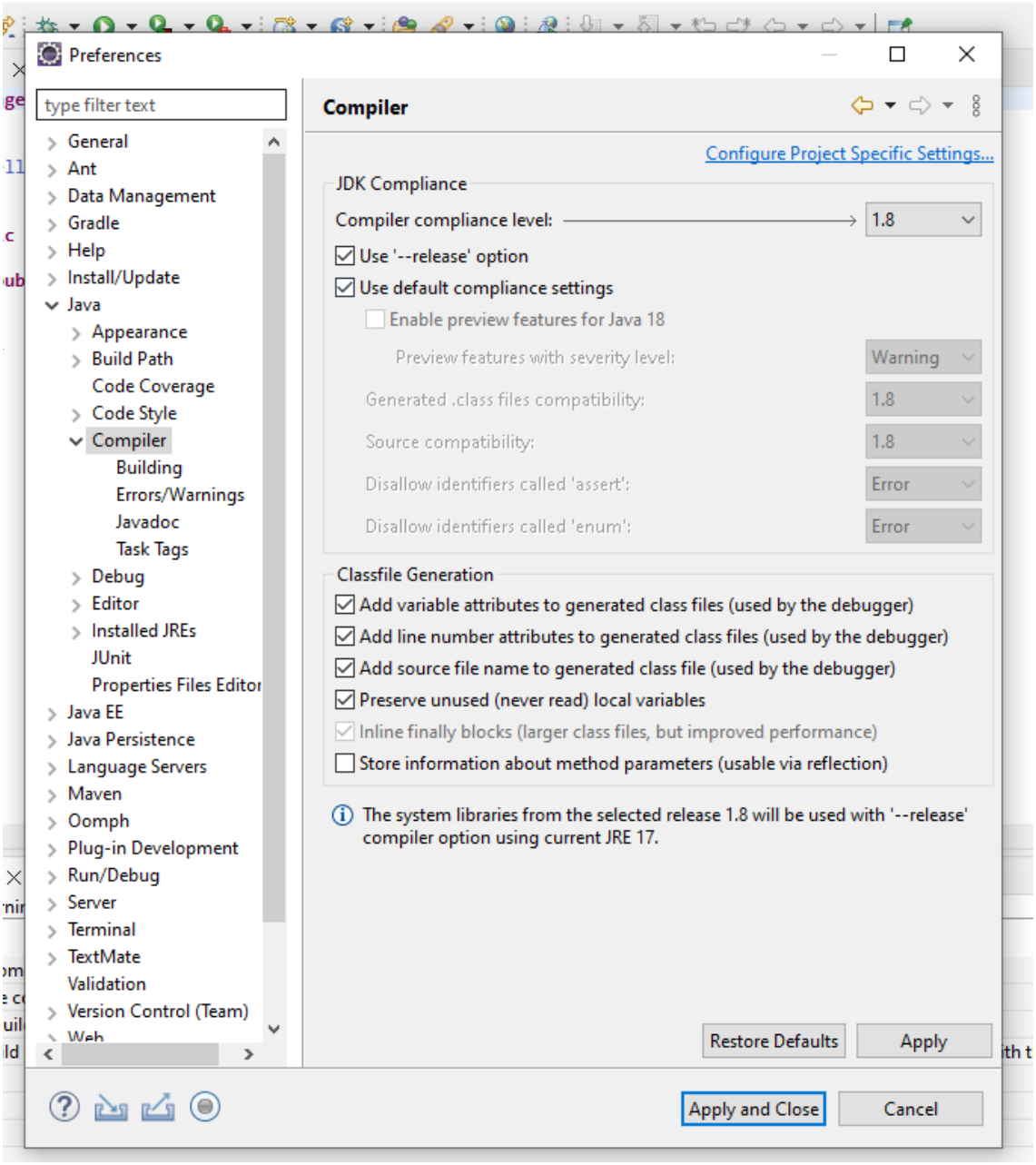
Es mejor indicar la JDK a la JRE, puesto que es más completa.




Add JRE
JRE Definition
Specify attributes for a JRE
JRE home:
C:\Program Files\Java\jdk1.8.0_211
Directory...
JRE name:
jdk1.8.0_211
Default VM arguments:
Variables...
JRE system libraries:
C:\Program Files\Java\jdk1.8.0_211\jre\lib\resources.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\rt.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\jce.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\access-br
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\cldrdata.j
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\jaccess.ja
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\jfxrt.jar
C:\Program Files\Java\jdk1.8.0_211\jre\lib\ext\localedata
Add External JARs...
Javadoc Location...
Source Attachment...
External annotations...
Remove
Up
Down
Restore Default
< Back
Next >
Finish
Cancel


Preferences
type filter text
Gradle
Help
Install/Update
Java
Appearance
Build Path
Classpath Variable
User Libraries
Code Coverage
Code Style
Compiler
Debug
Editor
Installed JREs
Execution Environ
JUnit
Properties Files Editor
Java EE
Java Persistence
Language Servers
Maven
Oomph
Plug-in Development
Run/Debug
Server
Terminal
TextMate
Validation
Version Control (Team)
Web
Web Services
Installed JREs
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
jdk1.8.0_211 (default)
C:\Program Files\Java\jdk1.8.0_211
jre
D:\Users\alumno.AMAV\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v202209
Add...
Edit...
Duplicate...
Remove
Search...
Apply
Apply and Close
Cancel

Además, hay que indicar la versión del compilador:



6. Programación básica

6.1. Programa principal

El punto de acceso a cualquier programa es la función *main*:

```
package miprimerprograma;

public class MiPrimerPrograma {

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

6.2. Palabras reservadas

En Java existen una serie de palabras con un significado especial. Algunas de ellas son:

```
abstract   class      final        int          public       this
assert     continue   finally      interface    return       throw
boolean    default    float        long         short        throws
break      do         for          native       static       transient
byte       double     if           new          strictfp     try
case       else       implements   package      super        void
catch      enum       import       private      switch       volatile
char       extends    instanceof   protected    synchronized while
```

6.3. Variables

Son objetos cuyo valor puede ser modificado a lo largo de la ejecución de un programa.

En la declaración de una variable se debe indicar el tipo, el nombre de la misma y opcionalmente el valor.

Los tipos primitivos son los siguientes:

Tipo	Uso	Tamaño	Rango
`byte`	entero corto	8 bits	de –128 a 127
`short`	entero	16 bits	de –32 768 a 32 767
`int`	entero	32 bits	de –2 147 483 648 a 2 147 483 647
`long`	entero largo	64 bits	± 9 223 372 036 854 775 808
`float`	real precisión sencilla	32 bits	de -10^{32} a 10^{32}
`double`	real precisión doble	64 bits	de -10^{300} a 10^{300}
`boolean`	lógico	1 bit	*true* o *false*
`char`	texto	16 bits	cualquier carácter

Un ejemplo:

```
int edad;
edad = 18;

System.out.println("La edad del alumno es de: "+ edad + " años");
```

6.4. Constantes

Las constantes son datos que permanecen sin cambios durante el desarrollo del programa, es decir, el programador no puede modificar su valor una vez inicializado.

```
final double PI = 3.141592;

System.out.println(PI);
```

Las constantes siempre van en mayusculas
Si se ponen fuera se ponen entre public class y public static void
Si se ponen fuera tenemos que poner static final

6.5. Literales

Los valores literales son aquellos que podemos asignar a las variables.

Hay varios tipos de literales:

- **Literales numéricos**: permiten representar distintos valores numéricos:
  - long: l ó L.
  - float: f ó F
  - double: d ó D.

```
float realPrecisionSimple = 5.2f;
float prueba;
prueba = 59.9;
```

Add cast to 'float'
Change type of 'prueba' to 'double'

```
...
float prueba;
prueba = (float) 59.9;
...
```

SOLUCIÓN:

```
float realPrecisionSimple = 5.2f;
float prueba;
prueba = 59.9f;
```

- **Binario y hexadecimal:** Para hexadecimal se añade al principio “0x”;

```
int variableBinaria = 011010;
int variableHexadecimal = 0x1a;
```

- **Booleanos**: verdadero o falso.

```
boolean varVerdadera = true;
boolean varFalsa = false;
```

- **Caracteres**:

- \n: Salto de línea
- \t: Tabulador
- \b: *Backspace*. Retorno
- \r: Retorno de carro
- \uxxxx: donde xxxx es el código Unicode del carácter. Por ejemplo \u0027 es el apóstrofe (')

6.6. Comentarios

Permiten al programador hacer anotaciones en el código para facilitar su comprensión:

```
int edad; //Comentario particular
edad = 18;

/*
 * Comentarios generales
 * Podría continuar...
 */
```

6.7. Operaciones básicas

Java dispone de multitud de operadores con los que se pueden crear expresiones utilizando como operandos variables, constantes, números y otras expresiones.

- **Asignación**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
=	Operador asignación	n = 4	n vale 4

- **Aritméticos**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	-4	-4
+	Suma	2.5 + 7.1	9.6
-	Resta	235.6 - 103.5	132.1
*	Producto	1.2 * 1.1	1.32
/	División (tanto entera como real)	0.050 / 0.2 7 / 2	0.25 3
%	Resto de la división entera	20 % 7	6

- **Incremento/Decremento**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

++	Incremento
	i++ primero se utiliza la variable y luego se incrementa su valor
	++i primero se incrementa el valor de la variable y luego se utiliza
		4++
		a=5;
		b=a++;
		a=5;
		b=++a;
			5
			
			a vale 6 y b vale 5
			
			a vale 6 y b vale 6

--	decremento	4--	3

- **Aritméticos combinados**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
+=	Suma combinada	a+=b	a=a+b
-=	Resta combinada	a-=b	a=a-b
*=	Producto combinado	a*=b	a=a*b
/=	División combinada	a/=b	a=a/b
%=	Resto combinado	a%=b	a=a%b

- **Relacionales**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	7 == 38	false
!=	distinto que	'a' != 'k'	true
<	menor que	'G' < 'B'	false
>	mayor que	'b' > 'a'	true
<=	menor o igual que	7.5 <= 7.38	false
>=	mayor o igual que	38 >= 7	true

- **Lógicos o booleanos**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	!false !(5==5)	true false
|	Suma lógica – OR (binario)	true | false (5==5)|(5<4)	true true
^	Suma lógica exclusiva – XOR (binario)	true ^ false (5==5)|(5<4)	true true
&	Producto lógico – AND (binario)	true & false (5==5)&(5<4)	false false
||	Suma lógica con cortocircuito: si el primer operando es true entonces el segundo se salta y el resultado es true	true || false (5==5)||(5<4)	true true
&&	Producto lógico con cortocircuito: si el primer operando es false entonces el segundo se salta y el resultado es false	false && true (5==5)&&(5<4)	false false

- **Condicional o ternario**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

?:	operador condicional	a = 4;
		b = a == 4 ? a+5 : 6-a;
		b = a > 4 ? a*7 : a+8;
			b vale 9
			b vale 12

```
int valor1 = 2;
int valor2 = 3;

boolean esMayor = valor1<valor2 ? true: false;
```

- **Concatenación**:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
+	Operador concatenación	"Hola" + "Juan"	"HolaJuan"

En caso de que una expresión tenga varios operadores, habrá una serie de prioridades: primero se evalúan los paréntesis, luego se sigue el orden de preferencias en función del tipo de operador y por último, en caso de igualdad, se evalúa de izquierda a derecha. La siguiente tabla indica la prioridad de ellos:

Descripción	Operador
Postfijos	expr++ expr--
Unarios prefijos	++expr --expr +expr -expr !expr
Aritméticos	* / %
Aritméticos	+ -
Relacionales	< <= > >=
Comparación	== !=
AND lógico	&&
OR lógico	||
Ternario	? :
Asignación	= += -= *= /= %= &= ^=

6.8. Conversión de tipos

Otro aspecto relevante del uso de expresiones con operadores es la conversión de tipos de datos. Hay ocasiones en las que una operación con dos tipos de datos da lugar a otro diferente, como podría ser el caso de dos enteros que dan lugar a un número real, en cuyo caso, la conversión es automática. En otras situaciones, el propio programador requiere hacer conversiones forzosas de datos, para lo que se utiliza el "*cast*" o "*casting*":

```
int a = 3;
int b = 2;

float c = a/b;

float d = (float)a/b;
```

7. Librerías. Entrada y salida

Java dispone de varias librerías que permiten realizar tareas comunes. Es común referirse a estas librerías como el API (*Application Programming Interfaces*) de Java.

Un API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

Una de las clases más utilizadas es la de *System*, que permite la salida de datos por pantalla:

```
System.out.printf("Mensaje");

System.out.println("Mensaje con retorno de carro");
```

Para la entrada de datos, se utiliza la clase Scanner:

```
Scanner sc = new Scanner(System.in);

System.out.print("Introduzca el valor 1:");
int v1 = sc.nextInt();
System.out.print("Introduzca el valor 2:");
int v2 = sc.nextInt();
```