TAREA UD 1. INTRODUCCIÓN A LA PROGRAMACIÓN.
ENTORNO DE DESARROLLO JAVA.

CONTENIDO

APARTADO I. INSTALACIÓN DEL ENTORNO DE DESARROLLO JAVA

En esta tarea realizaremos la **configuración del entorno de trabajo** a lo largo del curso. Por el momento**, no instalaremos ningún IDE** para hacernos con el entorno y nos ayude a entender cómo se compila y ejecutan programas en Java, aunque, si lo deseáis, podéis adelantar el trabajo e instalar también el Netbeans [https://netbeans.apache.org/download/nb15/] (los que empleen Linux que lo hagan en su versión comprimida, ZIP, pues el instalador, de momento, da ciertos problemas. Además, está garantizado el funcionamiento con Java a 11 hasta la versión 17) y/o Visual Studio Code [https://code.visualstudio.com/].

Muchas veces se suele partir directamente de un IDE como Netbeans, Visual Studio Code, etc., pero mi reciente experiencia es que muchas personas acaban por no entender en qué consiste el proceso de compilación y creación de una programa en Java.

Para poder realizar la tarea se *precisa* inicialmente instalar el *Java SE Development Kit* (JDK), por lo que el **primer apartado, no evaluable**, es preciso realizarlo.

Si hay algún problema o duda al respecto no dudéis en **preguntar en el foro de la presente unidad**.

- Obviamente, la instalación y **configuración de las variables de entorno dependen del sistema operativo**, pero son los mismos pasos.

- Las **últimas versiones Java sólo dan soporte para 64 bits**. Si alguien tiene un computador (demasiado) antiguo y no quiere o puede actualizarse, puede instalar una versión anterior.

A.1) INSTALA Y CONFIGURA EL ENTORNO DE TRABAJO, ADEMÁS DE REALIZAR LOS PRIMEROS PROGRAMAS JAVA:

1. **Instala Java 17 LTS SDK** (Java SE 17.0.4.1 (LTS)) o la **última versión en la actualidad es la 18. Este mes saldrá ese mes la versión 19. Cualquiera de ellas también es perfecto para este curso**. *En Windows, mi preferencia es instalarlo en la carpeta C:\jdk-<numero-version>\, para así poder acceder fácilmente desde línea de órdenes y tener todas las versiones organizadas (es una opción personal que adapto por motivos históricos, pues se hacía así desde un principio y después cambió realizar la instalación en el directorio de los programas).*

Recuerda que puedes descargar el OracleJDK u OpenJDK, incluso otras versiones menos conocidas. Se recomienda utilizar una versión LTS en explotación, si bien para desarrollo puede instalarse la última sin problemas.

https://www.oracle.com/java/technologies/

Detalles técnicos de instalación en diferentes plataformas:

https://docs.oracle.com/en/java/javase/18/install/overview-jdk-installation.html

Las versiones **LTS**, ***long-term support***, dan soporte/actualizaciones hasta que sale la siguiente versión LTS (dos o tres años). El resto de versiones sólo hasta que sale la siguiente versión, que es cada 6 meses.

En Windows existen tres versiones, dos instalables y otra versión comprimida, que es ideal cuando no tenemos permisos de instalación o queremos portabilidad para llevarlo en una unidad externa:

https://www.oracle.com/java/technologies/downloads/#jdk18-windows

Lo mismo sucede con macOS o Linux, que disponen de versión instalable (rpm, deb, dmg) y comprimida (tar.gz).

2. Descarga e descomprime la **documentación de Java** (API) en el directorio */docs* del JDK (ya viene ese directorio dentro del archivo comprimido):
   https://docs.oracle.com/en/java/javase/18/index.html
   https://www.oracle.com/java/technologies/javase-jdk18-doc-downloads.html

   Pues a lo largo del curso necesitaremos consultar la documentación del API de Java, en la que se pueden consultar las bibliotecas que podremos usar en nuestros programas, así como qué clases, constructores, métodos,… funciones dispone cada una de ellas.

3. **Configura las variables de entorno del Sistema** (PATH y JAVA_HOME adecuadamente. Del CLASSPATH ya hablaremos en otro momento). **Si utilizáis el instalador no se precisa configurar el PATH**, pues ya lo hace por defecto. Podéis consultar la guía de instalación en:
   https://docs.oracle.com/en/java/javase/18/install/overview-jdk-installation.html
   https://docs.oracle.com/javase/tutorial/essential/environment/paths.html

4. **Comprueba** a través de consola de órdenes (cmd.exe, terminal, etc.), que has realizado correctamente la instalación y configuración del JDK y JRE. (Orden: *javac –version*)

5. Crea una **carpeta en tu equipo para alojar los ejercicios y programas** que se irán generando en cada una de las actividades que se planteen en las unidades de trabajo. Se recomienda que el nombre de esta carpeta sea sencillo, sin espacios en blanco (puedes eliminarlo y empezar la siguiente palabra con mayúscula), ni caracteres especiales.

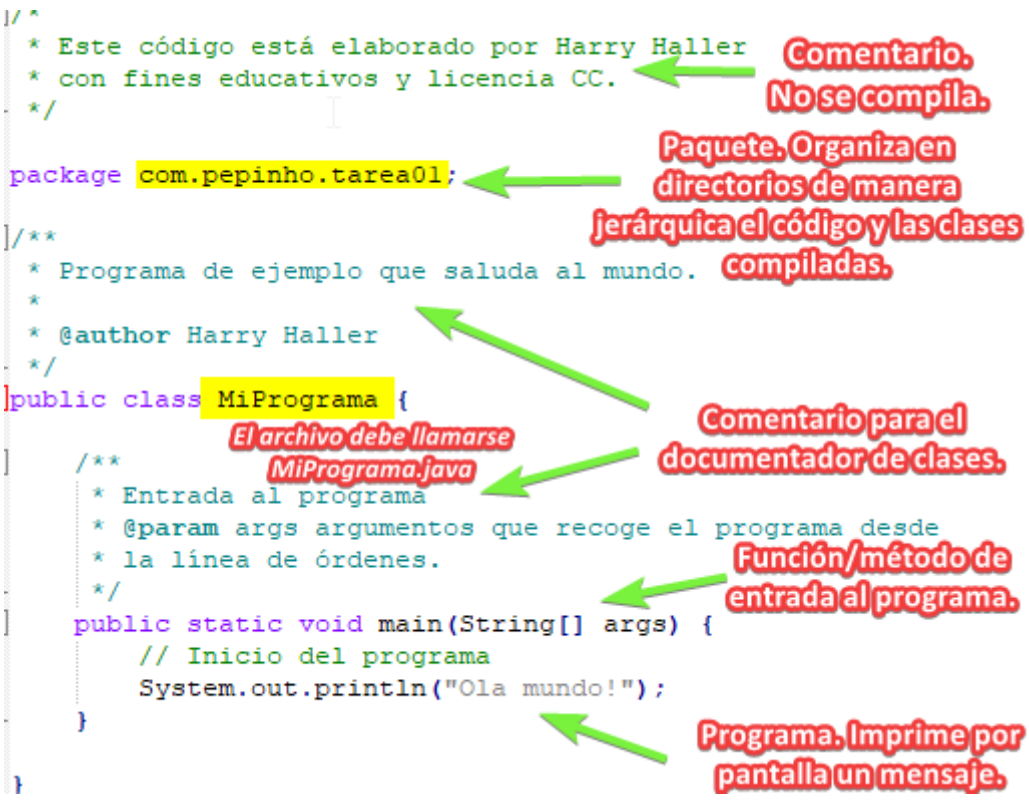
APARTADO II. USO DE LAS HERRAMIENTAS DEL JDK

En este apartado, ya evaluable, realizaremos nuestro primer programa en Java, haciendo uso de las herramientas disponibles en el JDK.

INTRODUCCIÓN Y AYUDA. ESTRUCTURA DE UN PROGRAMA JAVA.

El **archivo del código fuente de los programas Java deben llamarse igual** que el **nombre de la clase (pública) que está definida dentro del archivo** (conservando mayúsculas y minúsculas). Por ejemplo, si la clase se llama ***MiPrimeraApp***, el archivo fuente debe llamarse ***MiPrimeraApp.java***.

Por ejemplo, la estructura general (el archivo debe llamarse *MiPrograma.java*, igual que la clase) del código fuente de un programa Java:



- El **nombre del archivo debe coincidir con el nombre de la clase** y con extensión *.java, por lo que en el ejemplo debe llamarse ***MiPrograma.java***.

- La línea "***package***" indica el paquete, el directorio en el que debe estar la clase y el prefijo de la clase del programa, que en este caso es: ***com.pepinho.tarea01***, que debe ser *./com/pepinho/tarea01/MiPrograma.java*. No es necesario ponerlo, en cuyo caso el programa debe estar en el directorio actual y paquete por defecto.

- Aparecen dos tipos de **comentarios**. Los que empiezan con dos asteriscos son para el documentador de las clases. En cualquier caso, dichos comentarios no son compilados y se emplear para informar y documentar el código. Para comentar una línea se hace con //, el resto van entre /*…*/ o /**…*/.

- El programa debe tener un método "***main***" con la sintaxis indicada, que es el **punto en el que empieza a ejecutarse el programa**.

- Se recomienda, para mí obligado, que el nombre del programa (de las clases) siga la sintaxis "***CamelCase***". Esto es, **la primera letra y la primera de cada nueva palabra en mayúsculas**. Ejemplos: *CalculoRaiz*, *VentanaUsuario*, *NumeroCuenta*, etc.

```
/*
 * Autor: Pepe Calo
 * Realizado con fines educativos.
 * Puede modificarlo siempre que no lo haga con fines comerciales.
 */

package com.pepinho.programacion.boletin02;

/**
 * Este programa saluda a todo el mundo, o casi.
 * @author pepecalo
 */
public class MiPrograma {

    public static void main(String[] args) {
        System.out.println("Ola Otto!");
    }

}
```

Para ejecutar un programa, la orden ***java*** inicia una aplicación Java. Para ello, (1) inicia el *Java Runtime Environment (JRE)*, (2) carga en memoria la clase especificada y (3) llama al método main() de esa clase. El método debe declararse **public** y **static**, no debe devolver ningún valor (**void**). Además, debe aceptar un **array de String** como parámetro. La declaración del método tiene la siguiente forma:

public static void main(String[] args)

También puede ejecutarse el código fuente con la orden java (uso del modo de archivo fuente para iniciar programas).

TAREA

Haz una captura de pantalla de cada uno de los pasos. Ayuda:
https://docs.oracle.com/en/java/javase/11/tools/tools-and-command-reference.html

COMPILACIÓN Y EJECUCIÓN.

6. En la carpeta de trabajo, utilizando un editor de texto *(para Windows, Notepad++ es un buen editor que colorea el código: https://notepad-plus-plus.org/)*, crea un **archivo con extensión** ".java" al que debes llamar **OlaWittgenstein.java**. En su interior inserta las líneas necesarias de código Java para obtener por pantalla el siguiente resultado, en el paquete por defecto (no pongas la línea package):

    - ***¡No juegues con las profundidades de otro!***
    - ***Revolucionario será aquel que pueda revolucionarse a sí mismo.***
    - ***Una proposición sólo puede decir cómo es una cosa, pero no qué es ella.***
    - ***Si la gente no hiciera tonterías de vez en cuando, nunca se haría nada inteligente.***

7. Una vez creado el código fuente, **guarda el archivo y, mediante línea de comandos, realiza la compilación del mismo**. Consulta los mensajes de error, si se producen, antes de preguntar dudas. **El compilador indica la línea en la que se produce el error y el tipo de error**.

8. Comprueba lo que ha ocurrido en la carpeta donde está el archivo ".java" que acabas de compilar. Lista el contenido del directorio.

9. Realiza la **ejecución** del programa creado.
   https://docs.oracle.com/en/java/javase/11/tools/java.html

10. **Visualiza en pantalla los resultados**.

11. Si no se ajustan al ejemplo, realiza las modificaciones necesarias sobre el archivo fuente, vuelve a compilarlo y lanza su ejecución otra vez.

COMPILACIÓN Y EJECUCIÓN EN PAQUETES

12. Añade la línea del paquete donde corresponde:
    ***package** local.<tunombresinespacios>.tarea01;*

13. Crea un directorio llamado "***build***" (**no que preciso, pues lo hace la orden de compilación**) que será en dónde se pondrán los archivos compilados. Busca en la documentación la opción de compilación para **crear la jerarquía de**

directorios dentro de "build". Escribe la orden y una captura de pantalla de la jerarquía de directorios creados. Pista: opción -d.

https://docs.oracle.com/en/java/javase/11/tools/javac.html

Recuerda que una vez compilado, el programa debe estar en la jerarquía de directorio que corresponda: "*Debe organizar los archivos de origen en un árbol de directorios que refleje su árbol de paquetes*".

14. Entra en el ***directorio build y ejecuta el programa.*** Recuerda que el programa ahora debe precederse del nombre del paquete (el nombre del paquete forma parte del nombre de la clase):

```
java nombrePaquete.NombrePrograma
```

Que debe ser ejecutado desde el directorio padre, build en este caso, pues, si no se indica lo contrario, por defecto el directorio actual es en el que busca las clases compiladas (CLASSPATH).

CÓDIGO DESAPROBADO

15. Añade las siguientes líneas dentro del "***main***" pero al final, justo después de imprimir por pantalla, poniendo tu edad:

```
Integer idade = new Integer("40");
System.out.println("idade = " + idade);
```

Escribe los mensajes y **vuelve a compilar el programa de modo que te informe del aviso que se ha producido** (es código desaprobado).

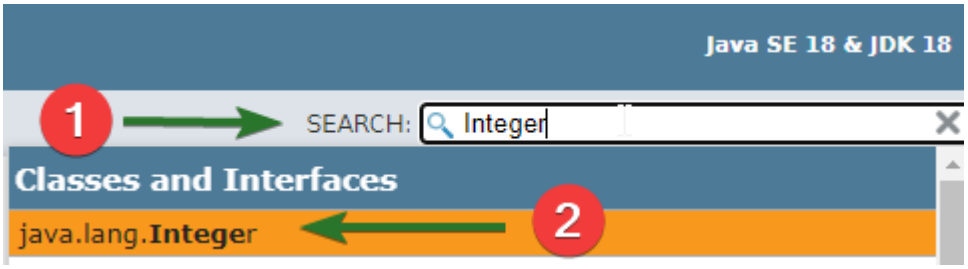
Nota: consulta en la siguiente página cómo se compila para ver los mensajes por el uso de código desaprobado (deprecated):

https://docs.oracle.com/en/java/javase/11/tools/javac.html

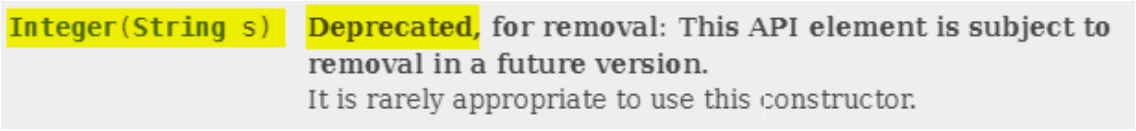
16. Consulta en la API de Java:

https://docs.oracle.com/en/java/javase/18/docs/api/overview-summary.html

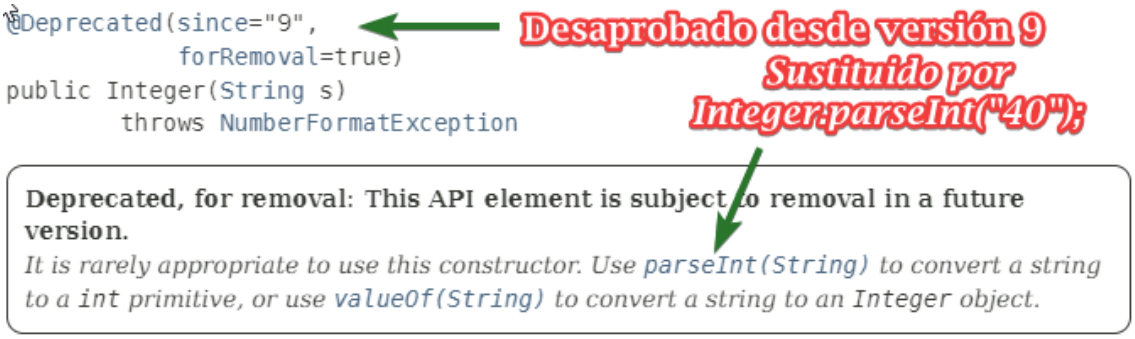
La clase *Integer* e intenta corregir el *"error"/warning* que se ha producido.



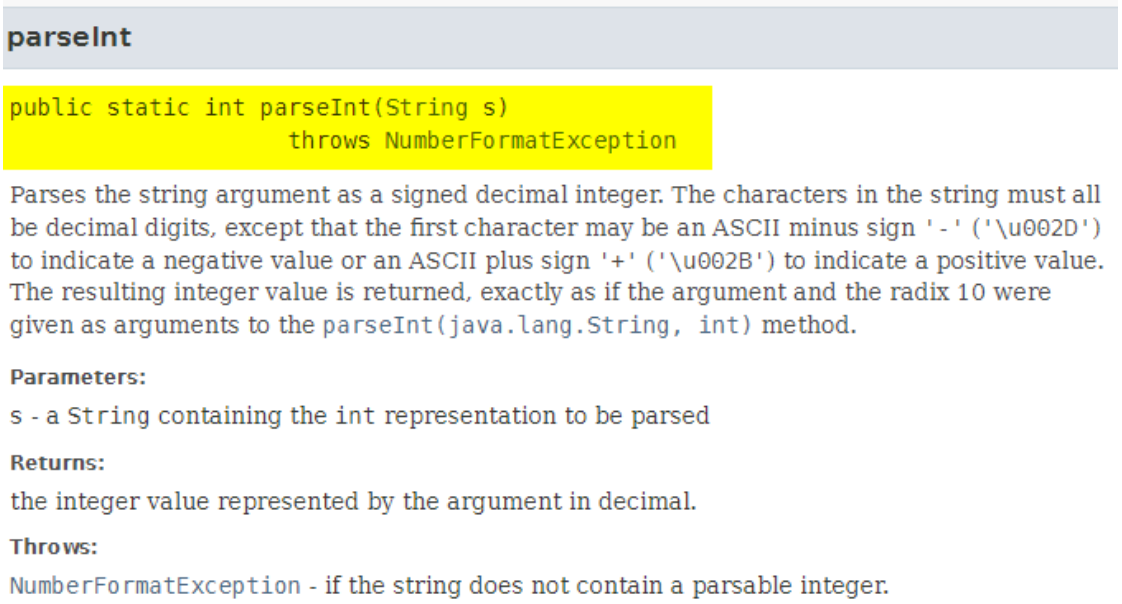
Podéis ver que el constructor está desaprobado:



Si enlazáis os lleva a la descripción del constructor que recoge una cadena:



Detalle del método estático *parseInt*:



Podéis comprobar que recoge una cadena de texto, como “40”, por ejemplo, y es estático. Esto es, se invoca precedido del nombre de la clase (busca en Internet cómo hacerlo).

Compila y ejecuta con ese cambio.

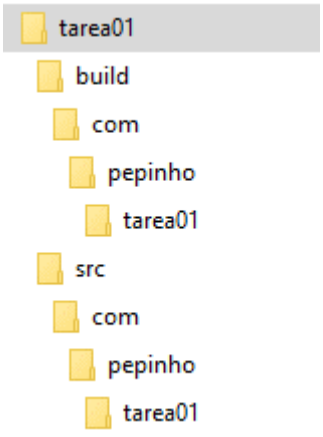
17. Crea una nueva clase, ***Alumno.java***, con el siguiente código:

```
package local.<tunombre>.tarea01;

public class Alumno {
   String nome;
   int idade = 18;

   public Alumno(String n) {
      nome = n;
   }

   public Alumno(String n, int i) {
      nome = n;
      idade = i;
   }
}
```

Modifica el programa, añadiendo las siguientes líneas:

```
package local.<tunombre>.tarea01;

public class MiPrograma {
    public static void main(String[  ] args) {
          // aquí va lo anterior…
          Alumno alumno = new Alumno("Xan", 20);
          System.out.println(alumno.nome + " de "
                  + alumno.idade);
    }
}
```

18. Crea un directorio "*src*" y, dentro de él, toda la jerarquía de directorios del código fuente del paquete (con el nombre que corresponda, en mi caso he puesto la clase en otro paquete diferente):



Copia los archivos fuente en ese directorio.

Compila desde el directorio raíz de trabajo (en mi caso, tarea01) en una única orden todas las clases.

19. **Ejecuta el programa desde el directorio raíz de trabajo**. Debes indicar la ruta en la que están las clases (***classpath***), *build* en nuestro caso.

USO DE LA HERRAMIENTA DE COMPRESIÓN Y EMPEQUETAMIENTO

INTRODUCCIÓN

La orden ***jar*** es una **herramienta de compresión y archivado** de propósito general, basada en los formatos de compresión **ZIP** y ZLIB. Inicialmente, el la orden fue diseñada para empaquetar applets de Java (no compatibles desde JDK 11) o aplicaciones; sin embargo, a partir de JDK 9, se puede usar para crear archivos JAR modulares:

https://docs.oracle.com/en/java/javase/11/tools/jar.html

Un archivo JAR puede contener una o más clases principales (con ***main***). **Cada clase principal es el punto de entrada de una aplicación. U**n archivo JAR teóricamente puede contener más de una aplicación, pero **debe contener al menos una clase principal (con main) para poder ejecutarse**.

Un archivo JAR puede tener un punto de entrada establecido en su archivo de manifiesto *(MANIFEST.MF).*

Al usar la orden ***jar***, se **debe especificar la operación** que debe realizar por los los argumentos de operación que recibe. Se puede mezclar un argumento de operación con otras opciones de una letra.

Generalmente, el argumento de la operación es el primer argumento especificado en la línea de órdenes:

- **-c** o **--create**: crea el archivo.
- **-i=*ARCHIVO*** o **--generate-index=*ARCHIVO***: muestra información del archivo JAR especificado.
- **-t** o **–list**: muestra la tabla de contenido del archivo.
- **-u** o **--update**: actualiza un archivo JAR existente.
- **-x** o **–extract**: extrae los archivos nombrados (o todos) del archivo.
- **-d** o **--describe-module**: imprime el descriptor del módulo o el nombre del módulo automático.

Ejemplo de creación:

```
jar cf miprograma.jar com/pepinho/programación/tarefa01/*.class
```

(faltaría especificar la clase principal)

20. Crea un archivo jar desde el directorio de trabajo, indicando cuál es la clase principal del programa.

21. Ejecuta el programa.

Por suerte, todo este proceso tedioso lo hacen por nosotros los IDE de programación, que emplean herramientas como Maven o Ant. Un lío, verdad. Pues para eso están los IDE, para facilitarnos este trabajo.

GENERACIÓN DE DOCUMENTACIÓN

Ayuda:
https://docs.oracle.com/en/java/javase/11/tools/javadoc.html
https://www.baeldung.com/javadoc

22. Por último, se trata de generar la documentación de las dos clases. Para ello crea un directorio "*doc*" en el directorio de trabajo y escribe la orden que permita crea la documentación de las clases del código fuente en dicho directorio.

*Muestra captura de **pantalla de cada uno de los pasos del ejercicio**. No del código fuente, sólo de la interface de órdenes con los "comandos" y la salida en cada caso.*

ENTREGA

El *pdf* **debe mostrar capturas de cada uno de los pasos (no de la instalación)**.

La **entrega se realizará en un único archivo PDF** que se nombrará siguiendo las siguientes pautas:

Apellido1Apellido2NombreTarefa01.pdf

Sin espacios ni caracteres espaciales como ñ, acentos, etc. Por ejemplo, *Ludwig Josef Johann Wittgenstein* enviaría un archivo con el nombre:

LudwigJosefJohannWittgenstein.pdf

EVALUACIÓN

Habrá varios niveles de evaluación:

NO APTO: cuando no se ha realizado el 40% de la tarea.
APROBADO (6): cuando se ha realizado correctamente entre el 40 y 70%.
NOTABLE (8): cuando se ha realizado entre un 70 y 90% de la tarea.
SOBRESALIENTE (10): cuando se han realizado casi toda la tarea (>90%).

Cualquier captura de pantalla sospechosa o que aparezca en varios estudiantes será considerada como una copia y evaluará la tarea con un 0.