Programación I
Grado en Ingeniería Informática
Curso 2018/2019

INTRODUCCIÓN A LA TECNOLOGÍA JAVA
Desarrollo de aplicaciones con la tecnología Java

Tabla de contenidos

1 Introducción

El presente enunciado realiza una presentación de la tecnología Java, de las herramientas de desarrollo disponibles y propone actividades encaminadas a asimilar esta información. Junto al enunciado, se adjuntan una serie de clases Java utilizadas para la correcta realización de las actividades propuestas. La información contenida en este enunciado toma como referencia el sistema operativo Windows.

Antes de continuar con la resolución de los ejercicios se recomienda leer con atención este enunciado, a fin de conocer los objetivos de la práctica, así como el material proporcionado para su correcta realización.

1.1 Objetivos

El objetivo principal de la práctica es conocer el entorno de desarrollo de Sun Microsystems para aplicaciones desarrolladas mediante el lenguaje de programación Java.

Junto al objetivo general, la práctica tiene una serie de objetivos específicos. A saber:

- Herramientas de desarrollo. JDK
  - Conocer y realizar el proceso de instalación del JDK
  - Realizar la correcta configuración del entorno de trabajo.
  - Conocer la estructura del JDK
  - Conocer y manejar el API de Java.
- Aplicaciones Java.
  - Compilación de un fichero fuente.
  - Ejecución de un fichero compilado.
  - Generación de la documentación.

2 Entorno de trabajo

Como paso previo a la realización de los ejercicios, se presenta el concepto de *tecnología Java* y las herramientas de desarrollo que proporciona.

2.1 ¿Qué es la tecnología Java?

El término Java en su sentido más amplio denomina una tecnología encaminada a facilitar la creación de aplicaciones informáticas. La tecnología Java engloba:

- Un lenguaje de programación.
- Un entorno de desarrollo.
- Un entorno de despliegue o ejecución.

Nacida con el objetivo de conseguir la independencia de dispositivo o máquina en que se utilice, la tecnología Java ha tenido una importante repercusión en el desarrollo de aplicaciones durante la pasada década.

2.2 El lenguaje de programación

El lenguaje de programación Java se define por las siguientes características:

- *Orientado a objetos*. Ayuda al programador a visualizar el programa en términos de la vida real. Proporciona: Encapsulación, herencia y polimorfismo.
- *Robusto*. Elimina fallos de otros lenguajes, tales como aritmética de punteros, gestión de memoria o tratamiento de excepciones que afecta la fiabilidad del código.
- *Simple*. Aporta "simplificaciones" respecto a lenguajes similares y añade aspectos muy útiles; como por ejemplo, comunicación con elementos externos.
- *Distribuido*. Lenguaje orientado a la programación en red.
- *Portable*. Permite que el código del usuario sea escrito una vez y ejecutado en distintos sistemas operativos.
- *Seguro*. Proporciona seguridad personalizable, así como mecanismos que verifican y aseguran la consistencia del código.
- *Interpretado*. Provoca que sea más lento pero asegura la portabilidad del código y mejora la velocidad de desarrollo.

2.3 El entorno de desarrollo

Como entorno de desarrollo, la tecnología Java ofrece al usuario una amplia gama de herramientas: un compilador, un intérprete, un generador de documentación, etc.

El JDK (JSE Development Kit) es el entorno ofrecido por Sun Microsystems para el desarrollo de aplicaciones codificadas mediante el lenguaje de programación Java. Aunque existen entornos desarrollados por otras compañías, por ejemplo IBM, el entorno de Sun es el más popular y conocido.

2.4 El entorno de despliegue o ejecución

El lenguaje de programación Java se define como **compilado** respecto a la *máquina virtual* e **interpretado** respecto a la máquina real. Se analizará en detalle está expresión:

- Máquina virtual. Se trata de una máquina imaginaria que está implementada usando un software de emulación sobre una máquina real; es decir, no existe un ordenador como tal sino una serie de programas que simulan su modo de trabajo. Para desplegar, para ejecutar, una aplicación desarrollada mediante Java es IMPRESCIDIBLE disponer de una máquina virtual instalada en nuestra máquina real.
- Al aplicar el compilador sobre código fuente programado en Java se obtiene código compilado comprensible únicamente por la máquina virtual de Java. Este código compilado se denomina *bytecode*.
- Al ejecutar el intérprete de Java sobre el código compilado éste se traduce a código máquina legible por la máquina real.

La figura 1 muestra de manera gráfica el proceso de compilación/ejecución utilizado por el lenguaje de programación Java.

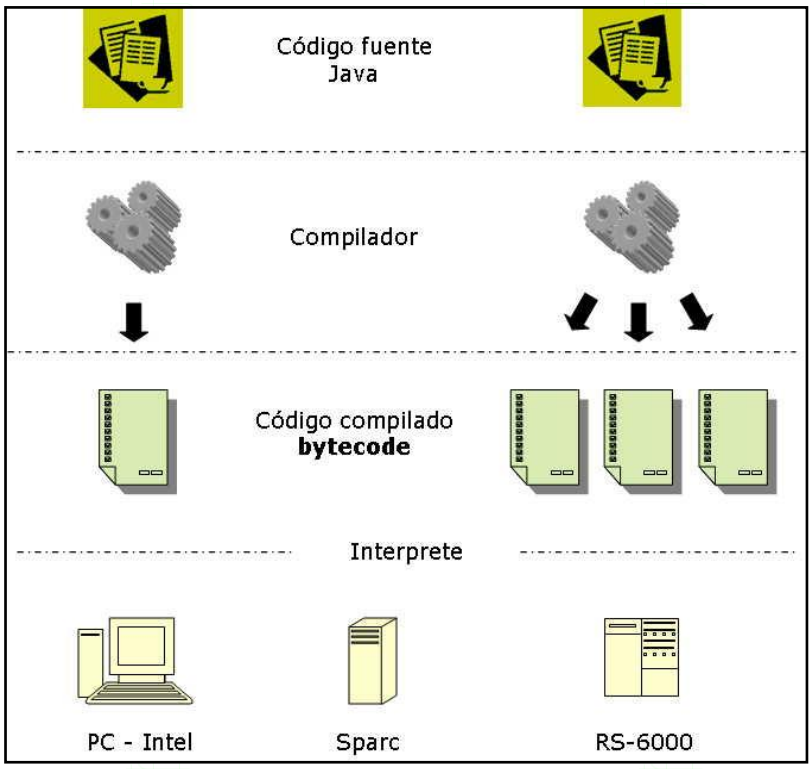

Figura 1. Proceso de compilación/ejecución del lenguaje de programación Java

La característica de portabilidad enumerada previamente viene dada por este proceso de compilación/ejecución: puesto que el *bytecode* es ejecutable en cualquier máquina virtual, el código fuente programado en Java puede ejecutarse en distintas plataformas. Únicamente debe cumplirse una condición, disponer de la máquina virtual correspondiente a la plataforma y sistema deseado.

3 Ejercicios

3.1 Herramientas de desarrollo

La sección muestra el procedimiento de descarga e instalación del entorno de desarrollo de Sun Microsystems.

3.1.1 Instalación

En este apartado se plantearán las actividades que permiten realizar la descarga e instalación del JDK. El alumno deberá realizar la siguiente secuencia de pasos:

Paso 1 – Descarga del JSE Development Kit

La descarga del fichero de instalación se realizará desde la página principal de la tecnología Java de ORACLE (http://www.java.sun.com/)[1].

Paso 2 – Instalación del JDK

El proceso de instalación se llevará a cabo siguiendo las indicaciones que para cada sistema operativo se ofrezcan desde la página de descarga.

Paso 3 – Estructura del JDK

La instalación del JDK dará lugar a un directorio con la siguiente estructura:

- `bin` Contiene los ficheros ejecutables del entorno de desarrollo. Entre los mismos cabe destacar:
  - `javac` Compilador de java.
  - `java` Intérprete de java.
  - `javadoc` Generador de documentación.
  - `jar` Herramienta de compresión en formato JAR.
  - `jdb` Depurador.
- `include` Contiene cabeceras de los métodos programados en código nativo que permiten extender el kernel de Java.
- `jre` Entorno de ejecución. Instancia de la máquina virtual que permite la ejecución de aplicaciones desarrolladas en Java.
- `lib` Contiene los ficheros de clases del kernel de Java. El fichero principal se llama tools.jar

3.1.2 Configuración del entorno

Una vez realizada la instalación del JDK es necesario configurar correctamente el entorno de despliegue de las aplicaciones. Para ello es necesario asignar valores a una serie de variables

[1] Aunque Sun Microsystems fue adquirida por Oracle y las páginas han cambiado, se ha decidido mantener en este enunciado el enlace original de Sun.

de entorno que intervienen en el proceso de compilación y ejecución. La configuración del entorno de trabajo se realizará respetando la siguiente secuencia de pasos:

Paso 1 – La variable JAVA_HOME

La variable JAVA_HOME es una **variable de entorno** que apunta al directorio base de la instalación de Java; es decir, a la ubicación del JDK. Si se han respetado las pautas indicadas en la actividad previa, el valor de la variable JAVA_HOME deberá contener el valor del directorio en el que se ha instalado el JDK. Por ejemplo, en el caso de **Windows** con la versión 1.8 sería:

```
set JAVA_HOME=C:\Archivos de programa\Java\jdk1.8.0_201
-- -- -- -- -- -- -- -- -- --
set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_201
```

Paso 2 – La variable PATH

La variable PATH es una **variable de entorno** utilizada por el sistema operativo para localizar la ubicación de ficheros ejecutables. Es necesario modificar su valor a fin de referenciar la ubicación de las herramientas de desarrollo aportadas por el JDK. Si se han respetado las pautas indicadas previamente en cuanto a versión y actualización descargada, el valor de la variable PATH será el siguiente:

```
set PATH=%JAVA_HOME%\bin;%PATH%
```

Paso 3 – Descarga del JSE Development Kit

Se comprueba la configuración consultando la versión del JDK de Java utilizada. Para ello, se ejecutará en el intérprete de comandos:

```
java -version
```

Si la configuración es correcta se obtendrá la etiqueta `1.8.0_201`, tal y como se muestra a continuación:

```
Resultado de la ejecución

java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09,
mixed mode)
```

IMPORTANTE

La modificación del valor de las variables de entorno puede realizarse directamente en el intérprete de comandos (*Inicio > Todos los programas > Accesorios > Símbolo del sistema)* En este supuesto, el valor de las variables tendrá carácter temporal, debiéndose actualizar cada vez que se inicie un nuevo intérprete de comandos.

3.1.3 El API de Java

El API de Java proporciona documentación en formato HTML de las clases predefinidas del JDK. A grandes rasgos, la información mostrada por cada clase incluye:

- Paquete al cual pertenece
- Descripción de la clase
- Constantes, si la hubieras, de la clase. Indicando su valor y tipo
- Constructores. Se define su uso y parámetros necesarios
- Métodos. Se describen los métodos, definiendo sus parámetros y tipos de retorno.

En algunos casos, la descripción de los constructores y métodos de la clase vienen precedidos de la palabra ***Deprecated***. Esto significa que el método o constructor ha dejado de ser soportado por el JDK desde la versión indicada. Su uso puede provocar errores de compatibilidades e impedir el correcto funcionamiento del código en versiones posteriores. En este caso, se indica una alternativa a su uso.

El API se encuentra accesible desde la página principal de la tecnología Java de Sun.

Actividad propuesta

Acceded al API de Java y familiarizarse en su manejo:

1. Consultad el método isJavaLetter() de la clase Character para comprobar que está *deprecated*.
2. Buscad las clases que forman parte del paquete java.lang
3. Consultad la clase Runtime para responder a las siguientes cuestiones:

   - ¿Para qué sirve la clase Runtime?
   - ¿Qué devuelve el método freeMemory()?
   - ¿Sería recomendable utilizar el método runFinalizersOnExit() en nuestro código?

3.2 El desarrollo mediante la tecnología Java

La sección instruye al alumno en el proceso de compilación, ejecución y documentación de una aplicación pensada y escrita en el lenguaje de programación Java.

> **IMPORTANTE**
>
> Se define como ***directorio de trabajo*** aquel que actúa como raíz de la estructura de directorios de una aplicación Java.

3.2.1 Primera aplicación Java

En este apartado el alumno acometerá la creación de su primera aplicación Java. Para ello, se respetarán los siguientes pasos:

Paso 1 – El fichero fuente

Se pide editar el fichero `Saludo.java` contenido en el directorio `/src` del material proporcionado, modificando el nombre de los autores.

Clase Saludo.java

```
001 /**
002  * Clase para mostrar por pantalla un mensaje de bienvenida.
003  *
004  * @author  PRG
005  * @version 1.0
006  */
007 class Saludo {
008
009     /**
010      * Metodo que muestra por pantalla un mensaje de bienvenida.
011      *
012      * @param mensaje Mensaje a mostrar por pantalla.
013      */
014     void saludar(String mensaje) {
015         System.out.println(mensaje);
016     }
017
018     /**
019      * Metodo principal de la clase.
020      *
021      * @param args Argumentos de la linea de comandos.
022      */
023     public static void main(String args[]) {
024         Saludo objetoSaludo;
025         objetoSaludo = new Saludo();
026         objetoSaludo.saludar("Bienvenido a PRG!");
027     }
028 }
```

Tabla 1. Contenido del fichero fuente.

Paso 2 – Compilación

Se compila el fichero `Saludo.java`, obteniendo el fichero compilado `Saludo.class`. Para ello, tras ejecutar el intérprete de comandos, deberán ejecutarse los comandos necesarios para desplazarse al ***directorio de trabajo***. Una vez situados en el *directorio de trabajo*, se ejecuta la siguiente instrucción:

Instrucción a ejecutar

```
javac –verbose –d ./classes ./src/Saludo.java
```

Comentario sobre la instrucción

El programa javac se corresponde con el compilador de Java. Se trata de una aplicación que recibe los parámetros de ejecución por la línea de comandos, de tal manera que la instrucción ejecutada se corresponde con:

```
javac <opciones><fichero fuente>
```

```
<opciones> =>–verbose –d ./classes
<fichero fuente> => ./src/Saludo.java
```

Dónde las opciones utilizadas se definen como:

```
        -verbose. Muestra los mensajes de salida resultado
de la compilación.
   -d <directorio> Directorio donde se crearán los
ficheroscompilados.
```

Para más información sobre el compilador ejecutar la instrucción:

```
javac –help
```

Como resultado de la ejecución se obtendrá el archivo `Saludo.class` en el directorio `/classes`, y una traza similar a la siguiente:

Resultado de la ejecución

```
[parsing started ./src/Saludo.java]
[parsing completed 12ms]
[search path for source files: .]
[search path for class files:……]
[loading java/lang/Object.class(java/lang:Object.class)]
[loading java/lang/String.class(java/lang:String.class)]
[checking Saludo]
[loading java/lang/System.class(java/lang:System.class)]
[loading java/io/PrintStream.class(java/io:PrintStream.class)]
[loading
java/io/FilterOutputStream.class(java/io:FilterOutputStream.clas
s)]
[loading java/io/OutputStream.class(java/io:OutputStream.class)]
[wrote ./classes/Saludo.class]
[total 571ms]
```

Paso 3 – Ejecución

Una vez realizada la compilación, desde el intérprete de comandos, y situados en el ***directorio de trabajo,*** se puede ejecutar el fichero compilado mediante la siguiente instrucción:

Instrucción a ejecutar

```
java –classpath ./classes Saludo
```

Comentario sobre la instrucción

El programa java se corresponde con el intérprete de Java. Se trata de una aplicación que recibe los parámetros de ejecución por la línea de comandos, de tal manera que la instrucción ejecutada se corresponde con:

```
java <opciones><fichero principal>
```

```
<opciones> => –classpath./classes
<fichero principal> => Saludo
```

Dónde las opciones utilizadas se definen como:

```
-classpath <path> Ubicación de los compilados.
```

Para más información sobre el interprete ejecutar la instrucción:

```
java –help
```

Como resultado de la ejecución se obtendrá:

Resultado de la ejecución

```
Bienvenido a PRG!
```

Paso 4 – Documentación del código

A partir del fichero fuente, y mediante la herramienta javadoc, se obtendrá la documentación en forma de páginas HTML. Para ello, posicionándose en el ***directorio de trabajo***, el alumno ejecutará desde el intérprete de comandos la siguiente instrucción:

Instrucción a ejecutar

```
javadoc -d ./doc -author -version -private
      -linksource ./src/Saludo.java
```

Comentario sobre la instrucción

El programa javadoc se corresponde con la herramienta de documentación de Java. Se trata de una aplicación que recibe los parámetros de ejecución por la línea de comandos, de tal manera que la instrucción ejecutada se corresponde con:

```
javadoc <opciones><ficheros a documentar>
```

```
<opciones> => -d ./doc -author -version
                      -private -linksource
<fichero compilado> =>./src/*.java
```

Dónde las opciones utilizadas se definen como:

```
-d <directorio> Directorio dónde se creará
       la documentación.
-author Se incluye la información de la
       etiqueta @author.
-version Se incluye la información de la
       etiqueta @version.
-linksource Genera páginas HTML con el código fuente.
```

Para más información sobre el interprete ejecutar la instrucción:

```
javadoc –help
```

Como resultado, se obtendrán una serie de ficheros con extensión `.html` en el directorio `/doc`. Para comprobar que la documentación ha sido correctamente generada deberá cargarse en el navegador el fichero `/doc/index.html` obteniéndose páginas similares a la mostrada en la figura 2 y 3.

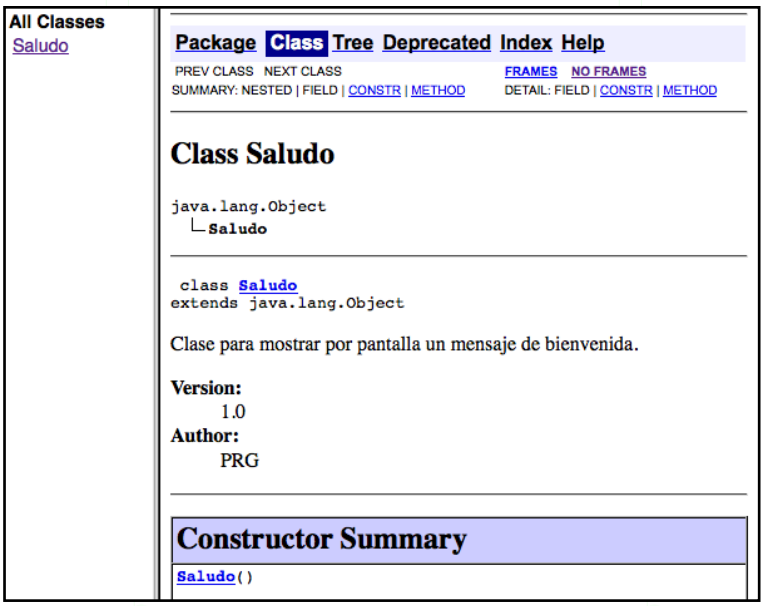

Figura 2. Extracto de la documentación javadoc de la clase Saludo

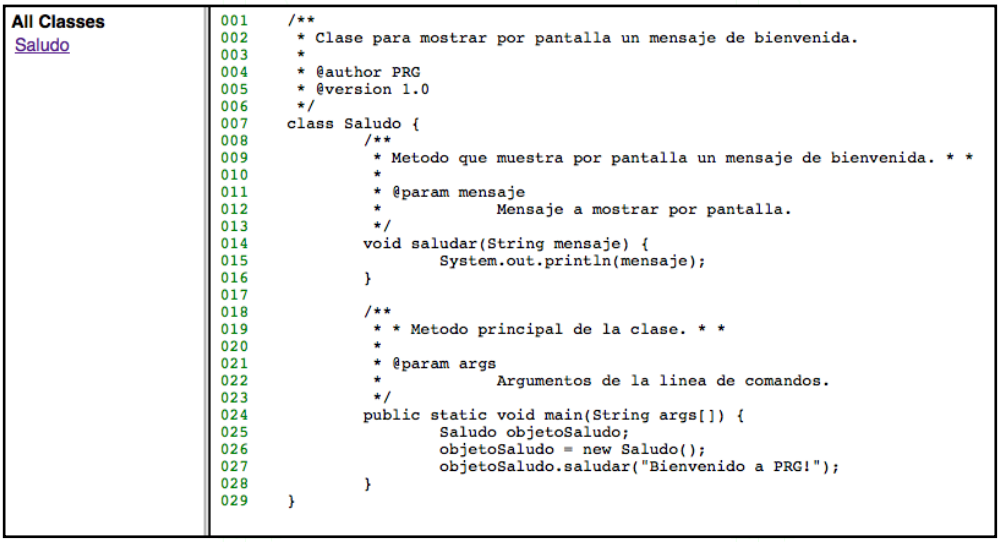

Figura 3. Código mostrado al navegar por el enlace “Saludo”

La obtención de la documentación del código depende de la labor del programador; es decir, Java proporciona la herramienta que permite generar la documentación pero es responsabilidad del programador el comentar correctamente su código. Java proporciona tres tipos de comentarios:

– Comentarios de línea. Implica una única línea de código.

```
01 // Esto es un comentario de línea
```

– Comentarios de bloque. Implica varias líneas de código.

```
01 /*
02    Esto es un comentario de bloque.
03    Ocupa varías línea de código
04  */
```

– Comentarios javadoc. Sirven como entrada para la herramienta javadoc para generación de documentación en página HTML.

```
01 /**
02  * En los comentarios javadoc se incluyen
03  * las descripciones de clases, atributos
04  * y metodos.
05  */
```

Dentro de los comentarios javadoc se incluye lo que se denomina `tags` y que sirven para producir una salida en las páginas HTML en un determinado formato; por ejemplo, la etiqueta `@author` muestra el nombre del autor tabulado y precedido por la palabra **Author** en negrita. Los `tags` que utilizaremos a lo largo de las prácticas de la asignatura son:

`@author` Seguido del nombre del programador de la clase.

`@version` Seguido de la versión del programa.

`@param` Seguido del nombre del parámetro y su descripción.

`@return` Seguido de la descripción del valor devuelto por el método.

Una práctica errónea pasa por comentar profusamente el código. Lo recomendable es incluir comentarios javadoc que aclaren el sentido y estructura de los métodos del programa, incluyendo comentarios adicionales (ya sea de línea o de bloque) en el caso de fragmentos de código especialmente complicados.

3.3 Organización en paquetes

Las clases se organizan en Java mediante paquetes. El paquete al que pertenece una clase viene especificado en la primera línea del fichero que contiene dicha clase. Por ejemplo, para que la clase `Saludo` pertenezca al paquete`es.unileon.prg1.holaMundo`, habría que incluir en la primera línea del fichero:

```
package es.unileon.prg1.holaMundo;
```

Para el nombrado de paquetes se sigue como convenio el inverso a la forma de nombrar sitios web.

Además de indicarle a la clase el paquete en el que se encuentra, el fichero `Saludo.java`tiene que estar incluido en el directorio `es/unileon/prg1/holaMundo`

El nombre de la clase ha cambiado. Si la clase pertenece a un paquete, para referirse a ella es necesario nombrar también el paquete al que pertenece. De esta forma, los comandos de compilación, ejecución y generación de la documentación se ven afectados. Partiendo de una estructura de directorios como la siguiente:

```
.
|__ workingDirectory
    |__src
       |__es
          |__unileon
             |__prg1
                |__holaMundo
                   |__Saludo.java
    |__classes
    |__doc
```

El comando de compilación es:

```
javac –d ./classes/ ./src/es/unileon/prg1/holaMundo/Saludo.java
```

El comando de ejecución es:

```
java –classpath ./classes es.unileon.prg1.holaMundo.Saludo
```

El comando de generación de documentación es:

```
javadoc -d ./doc -author -version -private –linksource
./src/es/unileon/prg1/holaMundo/Saludo.java
```

3.3.1 Ejercicio tic-tac-toe

A partir del código del TicTacToe, se pide:

1. Compilar el código
2. Generar la documentación
3. Ejecutar la clase `MainTicTacToe`
4. Incluir todas las clases en el paquete `es.unileon.prg1.ticTacToe`
5. Compilar desde el directorio de trabajo
6. Generar la documentación desde el directorio de trabajo
7. Ejecutar desde el directorio de trabajo
8. Incluir las imágenes dentro del directorio `etc/images` y volver a ejecutar la aplicación.
9. Crear un fichero llamado `leeme.txt` en el directorio de trabajo que incluya los comandos utilizados para cada una de las tareas anteriores: compilar, generar la documentación y ejecutar la aplicación.
10. Comprimir el directorio de trabajo y entregarlo a través de moodle. El archivo comprimido deberá tener el siguiente nombre:

    `ticTacToe_studentName.zip`

4 Material proporcionado

Para la realización de la práctica se proporciona, junto a este enunciado, dos ficheros comprimido con el material necesario para su correcta realización.

- `p-jdk.zip` - Una vez descomprimido el fichero, se creará un directorio `/p-jdk`con la siguiente estructura de trabajo:
  - **src.** Ficheros fuente de la aplicación. Contiene el programa principal de prueba.
  - **classes.**Ficheros compilados.Contendrá los *bytecode* de los ficheros fuente. Directorio vacío.
  - **doc.** Documentación javadoc. Directorio vacío.
- `ticTacToe.zip` - Una vez descomprimido el fichero, se creará un directorio `/ticTacToe`con la siguiente estructura de trabajo:
  - **src.** Ficheros fuente de la aplicación. Contiene las clases que componen la aplicación y las imágenes que se utilizan como fichas.
  - **classes.** Ficheros compilados. Contendrá los *bytecode* de los ficheros fuente. Directorio vacío.
  - **doc.** Documentación javadoc. Directorio vacío.

IMPORTANTE

Se aconseja la creación de un ***directorio de trabajo*** con nombre /PRG1 dónde descomprimir los ficheros proporcionados.