Segundo Ejercicio Práctico

https://ocw.uc3m.es/pluginfile.php/2379/mod_page/content/12/practica_java_basics.pdf

Este ejercicio práctico consta de tres partes:

- La **primera parte es una introducción al entorno de programación con Eclipse**. No se entregará ningún material para esta parte.
- La segunda parte tiene como objetivo introducir la programación en Java. Esta parte incluye una serie de ejercicios básicos que los estudiantes deberán desarrollar, así como un conjunto de preguntas asociadas a cada ejercicio. No se entregará el código de estos ejercicios.
- La tercera parte incluye ejercicios más complejos. El código de estos ejercicios deberá entregarse antes del 08 de mayo de 2026.

1. Primera parte: Introducción al IDE Eclipse

La primera parte del segundo ejercicio práctico es un tutorial que introduce las herramientas utilizadas en esta clase. No se entregará ningún trabajo para esta parte del tutorial.

1.1. Java SE

El Kit de Desarrollo de Java (JDK) es la plataforma estándar para el desarrollo de programas Java de propósito general. El JDK incluye, entre otros:

1. El compilador **javac**. El compilador **verifica la sintaxis** de los archivos fuente Java (*.java*) y los **convierte** en archivos de **bytecode** compilados (*.class*).
2. , La máquina virtual Java (**JVM**). La máquina virtual Java (JVM) **interpreta** los archivos de bytecode (*.class*) y ejecuta el programa.
3. El programa **javadoc**. *javadoc* genera automáticamente la **documentación** del programa.
4. Un catálogo completo de clases con **utilidades** para: gestión de archivos, creación de interfaces gráficas, comunicaciones, etc.

1.1.1. Compilador `javac`

javac es el programa que transforma los archivos fuente de Java en archivos de bytecode. Los archivos de bytecode pueden ser interpretados por la máquina virtual. javac lee un archivo fuente .java y genera un archivo .class correspondiente con el bytecode de cada clase incluida en él.

Por ejemplo, este comando en la ventana de comandos del sistema:

```
> javac HelloWorld.java
```

Crea un archivo **HelloWorld.class** asociado a la clase definida en *HelloWorld.java*[1].

1.1.2. Máquina Virtual de Java (JVM)

La **JVM** es un programa que interpreta los archivos de bytecode. La JVM es una capa abstracta entre los programas Java y el sistema operativo/hardware en el que se ejecutan. De esta forma, los programas Java son portátiles —lo que significa que pueden ejecutarse en diferentes plataformas sin modificar el código fuente—, porque son interpretados por máquinas virtuales Java con exactamente las mismas funcionalidades.

La llamada al programa JVM en la línea de comandos tiene un parámetro: el nombre del archivo de **bytecode** creado a partir del archivo fuente que contiene una clase con un método '`main`'. Todos los archivos '`.class`' mencionados por la clase con '*main*' deben ser accesibles (en una estructura de carpetas adecuada) para poder cargarlos:

Por ejemplo, este comando:

```
> java HelloWorld
```

Ejecuta la clase `HelloWorld` previamente compilada. Para ejecutar una clase, esta debe contener el método '`public static void main (String [] args)`'. Java ejecutará todas las instrucciones de este método secuencialmente.

1.1.3. javadoc

`javadoc` es una herramienta para generar automáticamente la documentación de las clases contenidas en un archivo fuente Java:

Por ejemplo, este comando:

```
> javadoc HelloWorld.java
```

Genera un archivo HTML llamado `HelloWorld.html` con la documentación de esta clase en el formato de la API de Java (Interfaz de Programación de Aplicaciones).

1.1.4. Paquetes de clases del JDK

El **JDK** incluye un repositorio completo de clases para facilitar el desarrollo de programas que acceden al sistema de archivos, crean interfaces gráficas, intercambian datos a través de una red de comunicación, etc. La documentación de las clases de JDK se encuentra en: http://java.sun.com/javase.

[1] Observe que la clase y el archivo .java que contiene la definición de la clase tienen el mismo nombre, incluyendo mayúsculas y minúsculas.

1.1.5. Desarrollo de programas con las herramientas del JDK

Normalmente, los programadores utilizan un entorno de desarrollo integrado (IDE) que facilita la creación, prueba y ejecución del código fuente. Sin embargo, las utilidades del JDK se pueden usar directamente. Este ejercicio explica cómo compilar y ejecutar programas Java sin usar un IDE.

Este ejercicio no debe entregarse.

Ejercicio 1. Uso directo de J2SE

1. Cree un archivo llamado `HelloWorld.java` con el contenido descrito en el Apéndice I. Use cualquier editor de texto (por ejemplo, el Bloc de notas de Windows) y guárdelo en el Escritorio. Asegúrese de que el archivo tenga la extensión **.java**, y no **.java.txt**.
2. Abra la consola del sistema. En Windows 11, haga clic en **Inicio** y escriba **cmd** en el cuadro de búsqueda. Otra opción es hacer clic en **Inicio** ->**Todos los programas** ->**Accesorios** ->**Símbolo del sistema**.
3. En la línea de comandos, escriba **cd Escritorio**.
4. Compile el archivo **Java** para transformarlo en **código de bytes** con **javac**. Asegúrese de que se haya creado un archivo **.class**[2].
5. Interprete el código de bytes con el programa **Java**.

1.2. Eclipse

Eclipse es un IDE gráfico (entorno de desarrollo integrado) basado en Java de código abierto (código abierto significa que el código fuente del programa está disponible y, bajo ciertas condiciones, puede modificarse y distribuirse. Más información sobre software de código abierto: http://www.opensource.org/docs/definition.php). El IDE de Eclipse está organizado en perspectivas. Cada perspectiva muestra las funcionalidades del IDE útiles para una tarea específica. Por ejemplo, la perspectiva Java predeterminada está diseñada para escribir código fuente Java y ejecutar programas Java, mientras que la perspectiva Debug está diseñada para depurar código. La perspectiva actual de Eclipse se puede cambiar seleccionando Ventana ->Abrir perspectiva en el menú, o haciendo clic en el botón situado en la esquina superior derecha de la pantalla. Como se mencionó, utilizaremos la perspectiva Java para la programación en Java.

1.2.1. Ejecución de Eclipse

La primera vez que se inicia Eclipse (y posteriormente, si se desea), se le solicita al usuario la ubicación del espacio de trabajo. El espacio de trabajo es el directorio donde Eclipse guarda los programas creados por el usuario (archivos .java) y los códigos de bytes compilados correspondientes (archivos .class), así como la información relacionada con los proyectos de

[2] Si Windows no encuentra el programa **javac**, antes de ejecutarlo, debe ejecutar el siguiente comando: ruta = "ruta del archivo javac"

software. Se recomienda a los estudiantes usar una memoria USB y seleccionar como espacio de trabajo un directorio del sistema de archivos USB. Para ello, se debe crear un directorio en la memoria USB (por ejemplo, llamado Java) y designarlo como espacio de trabajo al iniciar Eclipse.

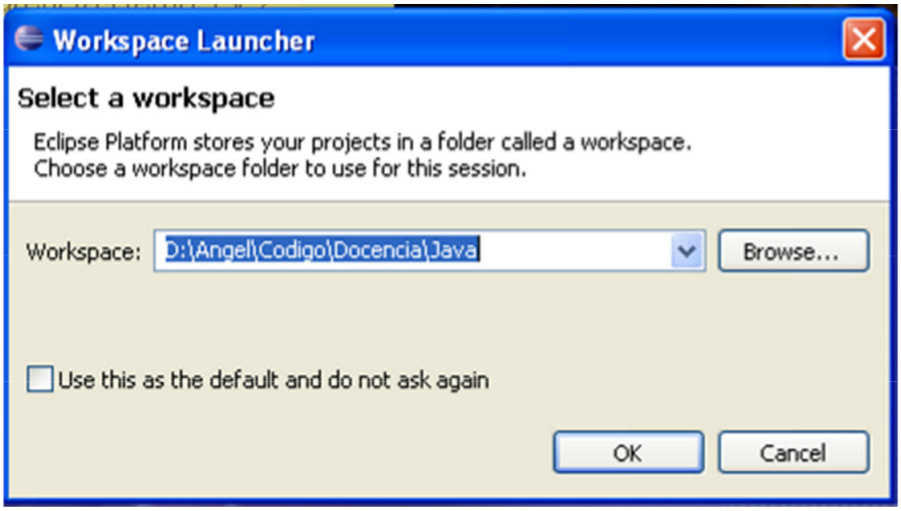

Figura 1: Selección del espacio de trabajo

1.2.2. Creación de proyectos

Eclipse gestiona el desarrollo de Java mediante proyectos; esto significa que para crear un programa Java en Eclipse, es necesario crear un proyecto previamente. Para crear un proyecto, siga estos pasos:

1. Seleccione **Archivo** ->**Nuevo** ->**Proyecto Java** en el menú.
2. Complete los datos del proyecto. Para proyectos sencillos, como los que crearemos en esta lección, solo es necesario especificar el nombre del proyecto. Cree un proyecto llamado **Práctica**, que se utilizará en este ejercicio y en el siguiente.
3. Una vez introducidos los datos del proyecto, haga clic en el botón **Finalizar**.

1.2.3. Creación de paquetes

Un paquete es una carpeta dentro del proyecto de Eclipse que se utiliza para almacenar archivos fuente de Java. Por defecto, Eclipse guarda los archivos fuente de las clases en un paquete predeterminado. Se recomienda crear al menos un paquete en un proyecto. Para crear un paquete:

1. **Archivo** ->**Nuevo** ->**Paquete** en el menú.
2. Asigne un nombre al paquete en el cuadro de diálogo de creación de paquetes (*practicalex2*) (La convención es nombrar los paquetes comenzando con una letra minúscula).

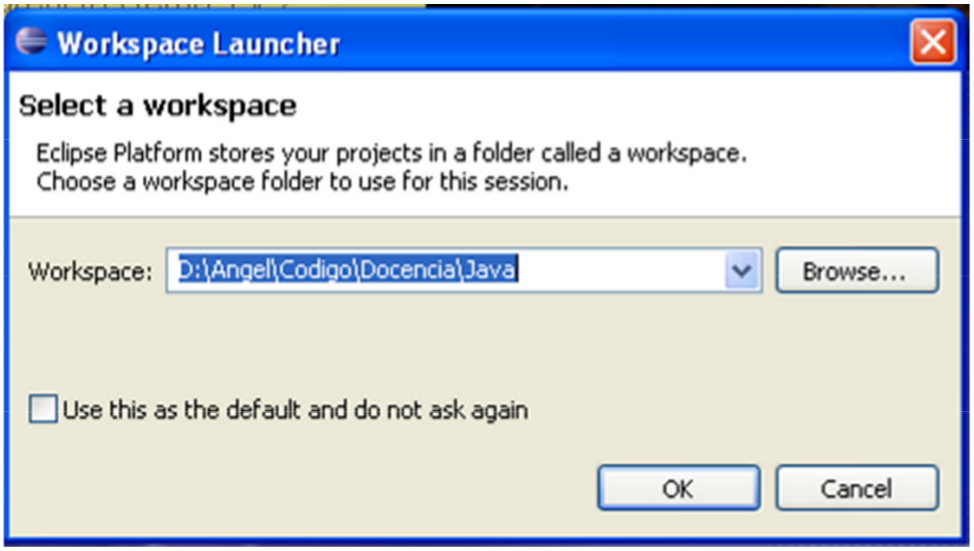

Figura 2: Creación del proyecto

Como resultado, Eclipse crea una carpeta para el proyecto en el espacio de trabajo con dos archivos: `.project` y `.classpath`. Estos archivos no contienen código Java, ya que son utilizados internamente por el IDE. El código fuente Java se guarda en la subcarpeta **src**. El código Java compilado se guarda en la subcarpeta **bin**. El espacio de trabajo activo se puede cambiar seleccionando **Archivo** ->**Cambiar espacio de trabajo** en el menú.

1.2.4. Creación de clases

Para crear una clase Java dentro de un proyecto, se deben seguir los siguientes pasos:

1. Seleccione **Archivo** ->**Nuevo** ->**Clase** en el menú.
2. Seleccione el paquete de la clase (`practicalex2`).
3. Escriba un nombre para la clase en el cuadro de diálogo de creación de clases que aparece a continuación (por ejemplo: *HelloWorld*). La convención es nombrar las clases Java comenzando con una letra mayúscula.
4. Especifique si se debe crear un método main para esta clase marcando la casilla correspondiente. (Esto no es necesario en nuestro ejemplo, ya que modificaremos todo el contenido del archivo con el código del Apéndice I).
5. Haga clic en el botón **Finalizar**.

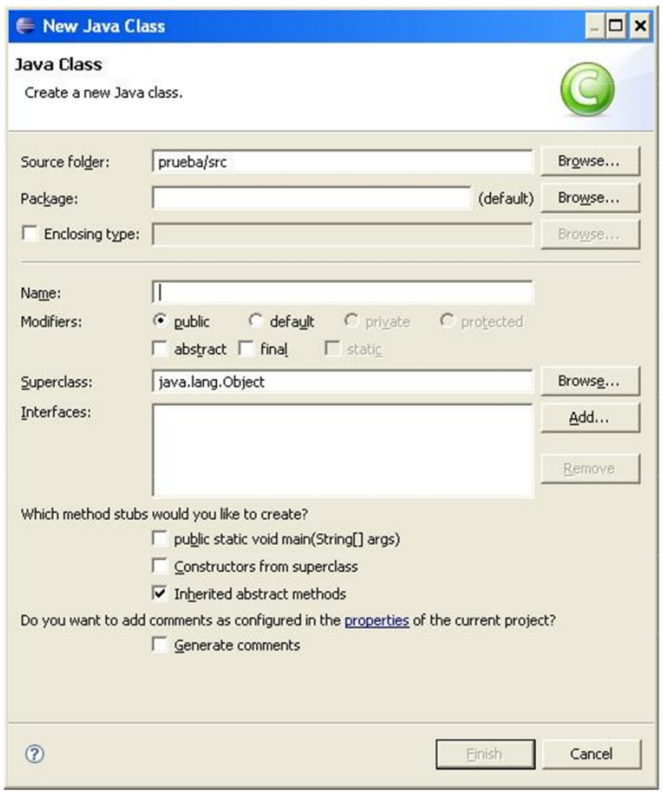

Figura 3: Diálogo de creación de clase

Eclipse crea un archivo llamado <*nombre de la clase*>.*java* con el código fuente correspondiente a la estructura básica de la clase. A continuación, se debe escribir el código que implementa la clase (dentro de la clase principal). Como se mencionó, modificaremos todo el contenido del archivo excepto la primera línea: el paquete de la clase.

1.2.5. Ejecución del programa

Los programas se pueden ejecutar en la perspectiva Java haciendo clic en **Ejecutar** ->**Ejecutar** en el menú, o bien, pulsando **Control** + **F11** en el teclado. Al ejecutar el programa, Eclipse muestra el resultado de la ejecución en la consola.

1.2.6. Ejecución del programa con argumentos

Se recomienda leer esta sección al resolver el ejercicio 28 de la segunda parte de la práctica.

Al igual que al invocar un programa Java desde la línea de comandos, Eclipse permite modificar los argumentos (también llamados parámetros del programa o simplemente parámetros) que se pasan al método `main` de la clase. Para modificar estos valores, seleccione **Ejecutar** ->**Abrir diálogo de ejecución** en el menú. En la pestaña *Argumentos*, podemos especificar una lista adecuada de argumentos, uno por línea. Cada argumento se almacenará en posiciones sucesivas del array *args*, es decir, el primero será *args[0]*, el segundo *args[1]*, etc.

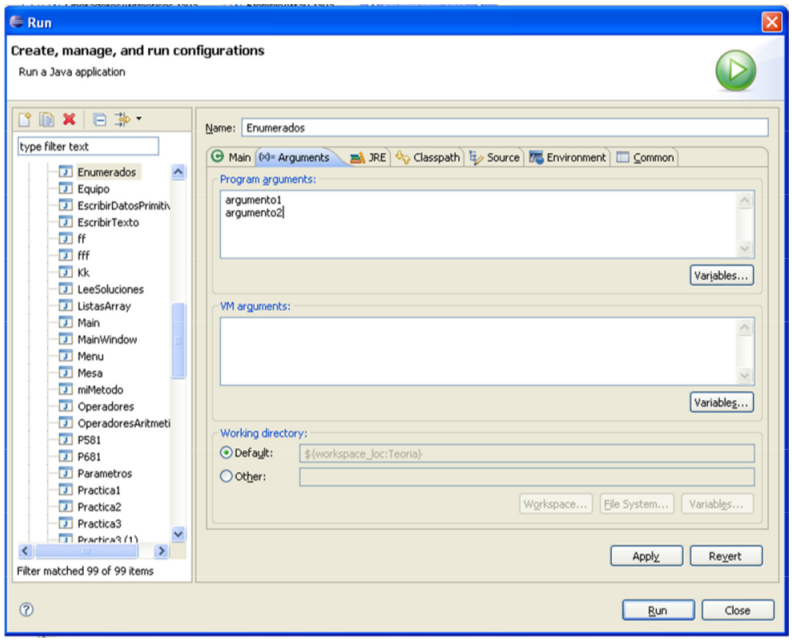

Figura 4: Argumentos en la ejecución de un programa Java

1.2.7. Depuración de programas

La perspectiva de depuración de Eclipse está diseñada para detectar y corregir errores de ejecución de programas, lo que se conoce como depuración. Esta perspectiva se activa seleccionando **Windows** ->**Abrir perspectiva** ->**Depurar** en el menú. La ejecución de programas con la perspectiva de depuración se puede analizar de las siguientes maneras:

1. Pausando la ejecución del programa al alcanzar un punto de interrupción. La opción de menú **Ejecutar** ->**Activar punto de interrupción** permite especificar la línea de código donde se pausa la ejecución. Tras definir un punto de interrupción, el programa se puede iniciar en modo de depuración seleccionando **Ejecutar** ->Depurar. La ejecución se pausará en el primer punto de interrupción.
2. Ejecutando el programa paso a paso (pausando la ejecución después de cada instrucción). Las opciones de menú **Ejecutar** ->**Avanzar paso a paso** y **Ejecutar** ->**Saltar** línea permiten ejecutar el programa línea por línea.

La perspectiva de depuración también permite a los programadores comprobar el valor de las variables durante la ejecución del programa en modo de depuración. La pestaña **Variables** muestra las variables activas en el ámbito actual y sus valores. También es posible añadir nuevas variables de observación para comprobar el valor de otras variables o expresiones. Para ello, hay que seleccionar la expresión que se desea observar en el editor de código, hacer clic con el botón derecho y seleccionar **Observaciones**/*Watches*. La expresión y su valor se mostrarán en la pestaña **Observaciones**. Las expresiones observadas se pueden editar y eliminar.

Se recomienda encarecidamente usar el depurador para detectar y corregir errores en tiempo de ejecución. El uso de la instrucción println suele generar nuevos errores y dificulta la detección de fallos en el programa.

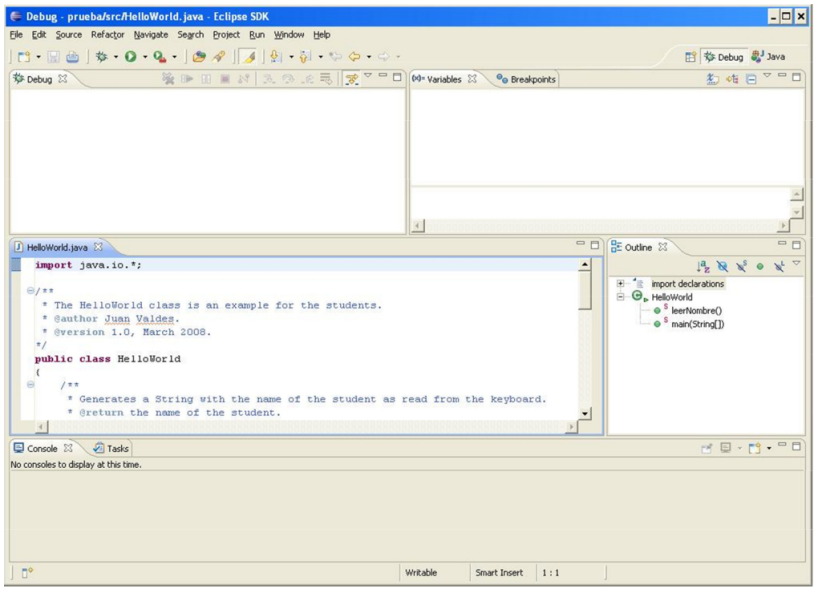

Figura 5: Perspectiva de depuración en Eclipse

Ejercicio 2. Introducción a Eclipse

1. Crea un proyecto Java que incluya la clase *HelloWorld* descrita en el Apéndice I.
2. Ejecuta el método *main* de la clase *HelloWorld*.
3. Ejecuta el método *main* de la clase *HelloWorld* paso a paso (modo depuración).
4. Observa el valor de la variable '*name*'.

Apéndice I: HelloWorld.java

```
/**
 * The HelloWorld class is an example for the students.
 * @author Juan Valdes.
 * @version 1.0, March 2008.
 */

import java.util.Scanner;

public class HelloWorld {
    /**
     * Genera una cadena con el nombre del estudiante
     * leído desde el teclado.
     * @return el nombre del estudiante.
     */
    public static String readName(){
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Ingrese su nombre y pulse Enter: ");
        String name = sc.nextLine();

        return name;
    }

    /**
    * Dice Hola al estudiante.
    * @param args el conjunto de argumentos de la línea de comandos.
    * @return .
    * @exception .
    */
    public static void main(String[] args) {
        if(args.length > 0){
            System.out.println("Hola " + args[0]);
        } else{
            System.out.println("Hola "+ readName());
        }
    }
}
```