PRÁCTICA 8

Programación de aplicaciones de red

1 Objetivos

Esta práctica pretende trabajar los fundamentos para el desarrollo de aplicaciones de red utilizando el API de sockets BSD. Además, se va a introducir el uso de un IDE (Integrated Development Enviroment), llamado Eclipse, que permitirá trabajar de manera más sencilla en el proceso de programación y depuración. Este guión va a cubrir el trabajo para las 5 próximas sesiones de prácticas donde, tras unos programas iniciales y básicos con sockets TCP/UDP, se planteará un proyecto de aplicación de red.

2 Material

- PC con Linux, IDE Eclipse y entorno de programación Java 1.6 SDK

3 IDE Eclipse

Eclipse es un IDE de código libre que puede ser usado para el desarrollo de proyectos de programación de diversas áreas, cómo por ejemplo Android o Java. Dicho de otro modo, es la herramienta que facilita al usuario la creación de proyectos de programación para cualquier lenguaje gracias a la instalación de diversos plugins.

El IDE permite organizar de una forma práctica y visual los proyectos, resalta código, muestra todos los métodos de una clase, o funciones disponibles, permite depuración visual, compilación, generación de documentación, etc.

El entorno y un primer programa, “HolaMundo”

Durante el desarrollo de esta práctica vamos a utilizar el entorno de desarrollo 3.5.2 SDK de Eclipse. Se arranca con el comando:

```
# eclipse
```

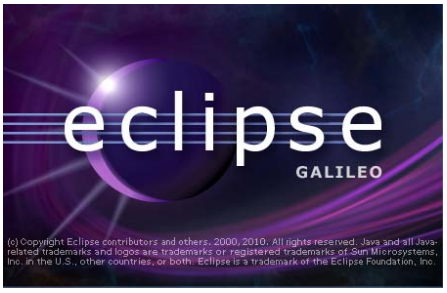


Figura 1: Versión Galileo de Eclipse

La primera ventana que se muestra al ejecutar eclipse es para seleccionar el directorio de trabajo. Los proyectos se guardarán en carpetas dentro de dicho directorio. Se debe de tener en cuenta que el IDE genera sus propios archivos para almacenar la información de los proyectos. Para trabajar con el proyecto creado en otra máquina se deberá copiar el directorio que se haya creado en la ruta del workspace especificada. Si se copiaran sólo las fuentes de los programas se debería de crear nuevamente un proyecto porque Eclipse no lo reconocería. Para el desarrollo de las prácticas, crearemos un directorio nuevo en nuestro $HOME.

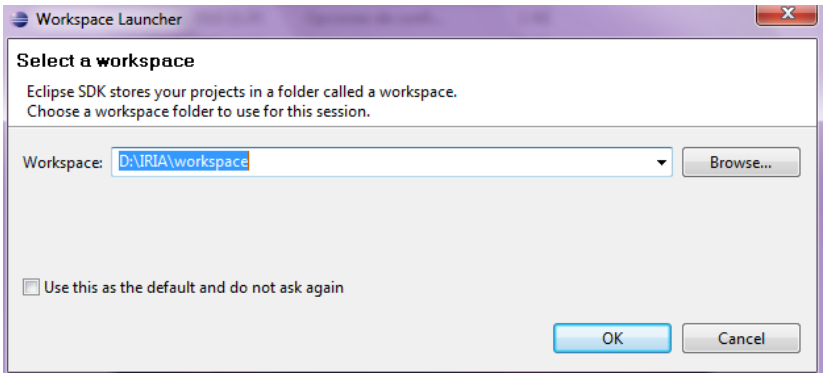


Figura 2: Selección del directorio de trabajo

Una vez seleccionado el directorio de trabajo se muestra la ventana de bienvenida, figura 3.

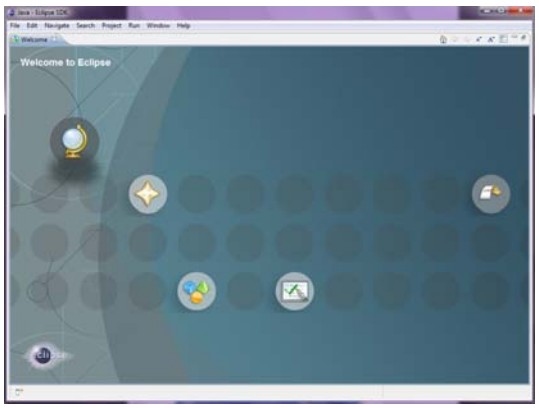


Figura 3: Ventana de bienvenida

Para comenzar a trabajar seleccionaremos “Go to the WorkBench”

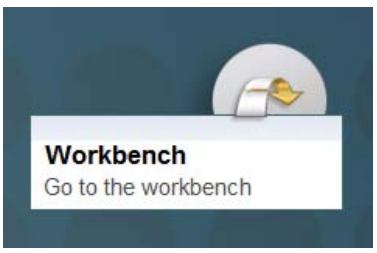


Figura 4: Comenzar a trabajar

La siguiente pantalla que muestra Eclipse es directamente el interfaz de trabajo. Este interfaz se divide en diversas ventanas y perspectivas. Por defecto se nos abre el entorno para comenzar a trabajar en un proyecto en Java, por lo que de momento no nos tenemos que preocupar de cambiar de vista. En las siguientes figuras se muestran las principales funciones y ventanas que serán utilizadas.

Para comprender las ventanas y perspectivas que se muestran, vamos a crear un nuevo proyecto, "holaMundo". Eclipse trabaja por proyectos. Dentro de un proyecto los archivos se organizan en directorios. Por ejemplo, las fuentes de los programas se guardarán en el directorio "src", los ejecutables generados en "out", las librerías importadas en "include"… etc. De esta forma los proyectos están organizados de una forma lógica permitiendo localizar rápidamente cada fichero.

Para crear un nuevo proyecto seleccionaremos desde el menú de opciones: File -> New-> Java Proyect. A este mismo menú se puede acceder desde la ventana de proyectos, figura 5, dándole un clic derecho con el ratón y escogiendo "Nuevo Proyecto".

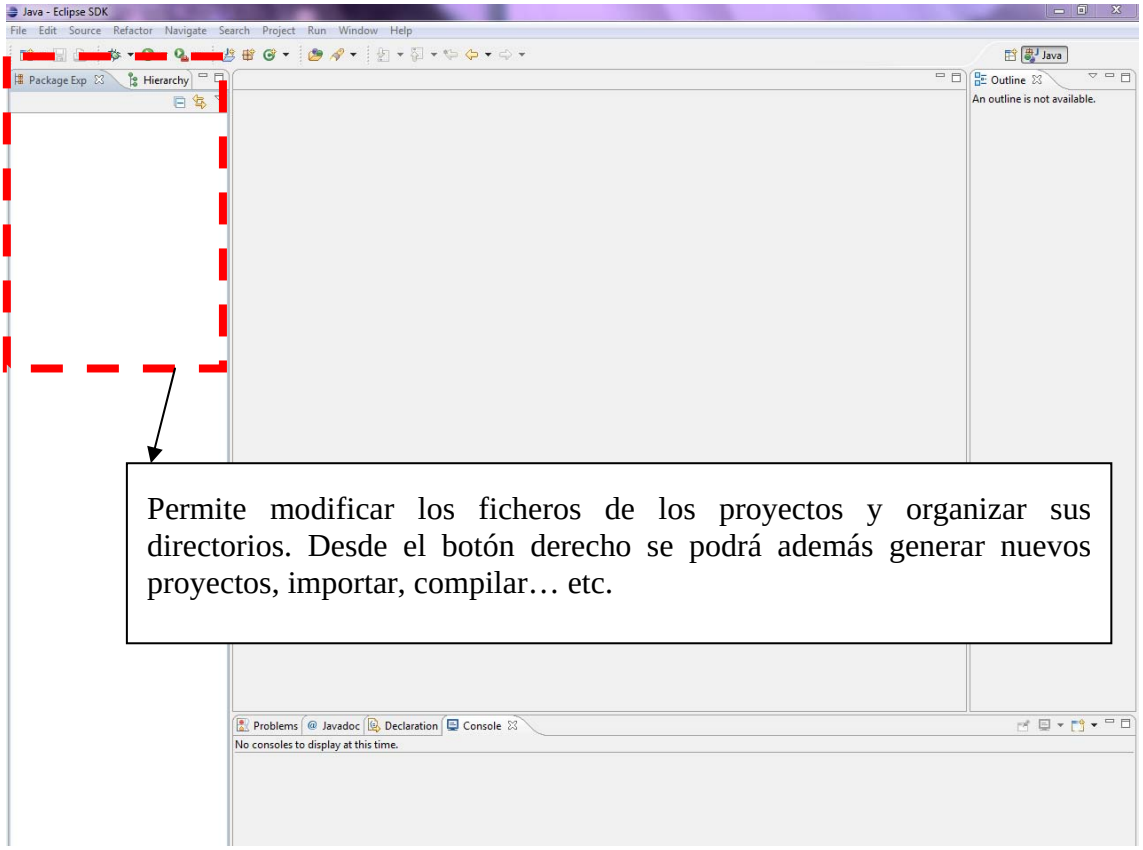


Figura 5: Ventana de edición de proyectos

Desde la opción "Java Project" se nos abrirá un asistente que nos permitirá crear de forma rápida un nuevo proyecto. Siga las instrucciones marcadas en la figura 6 y 7 para la creación del proyecto, "holaMundo".

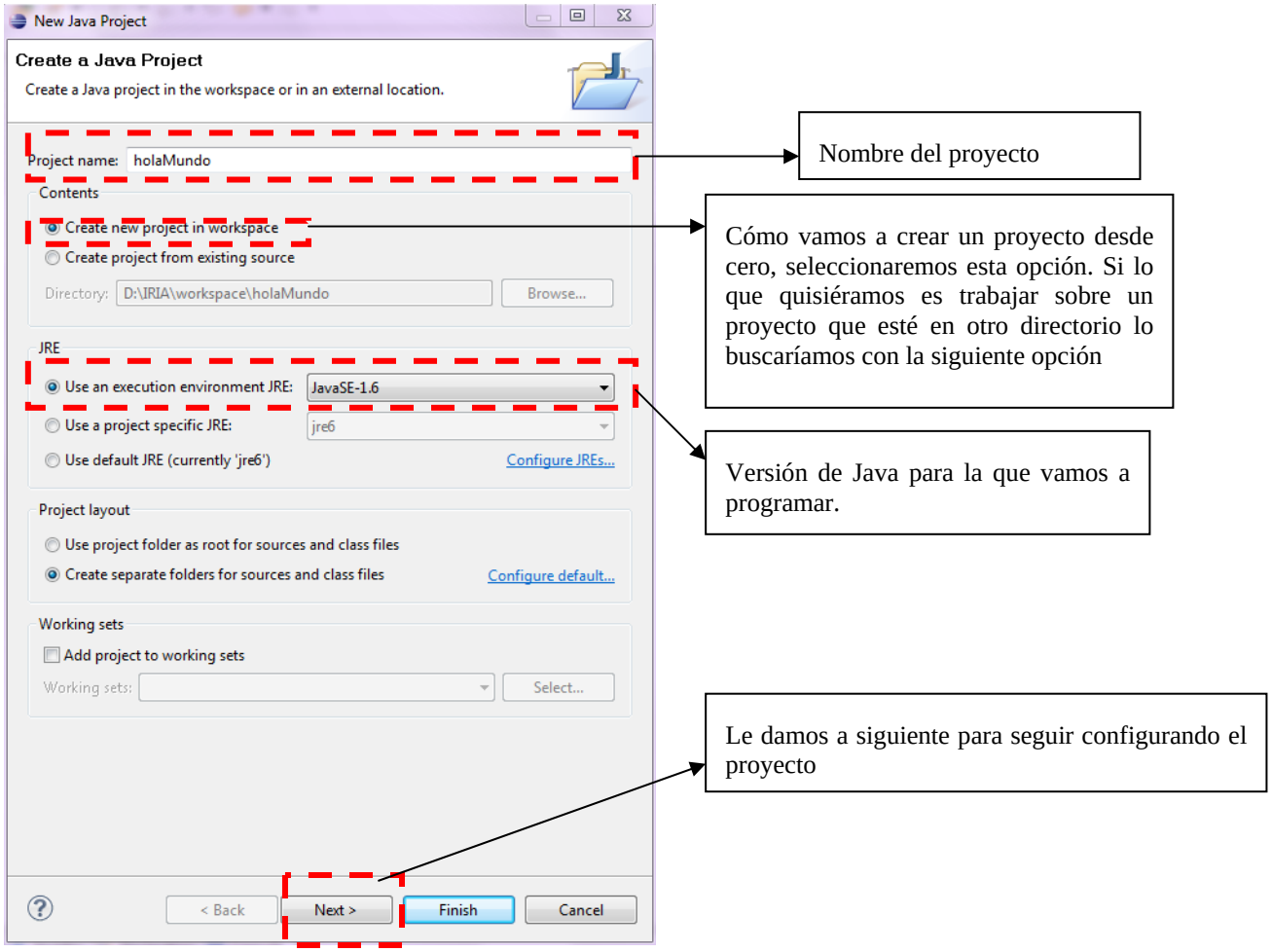
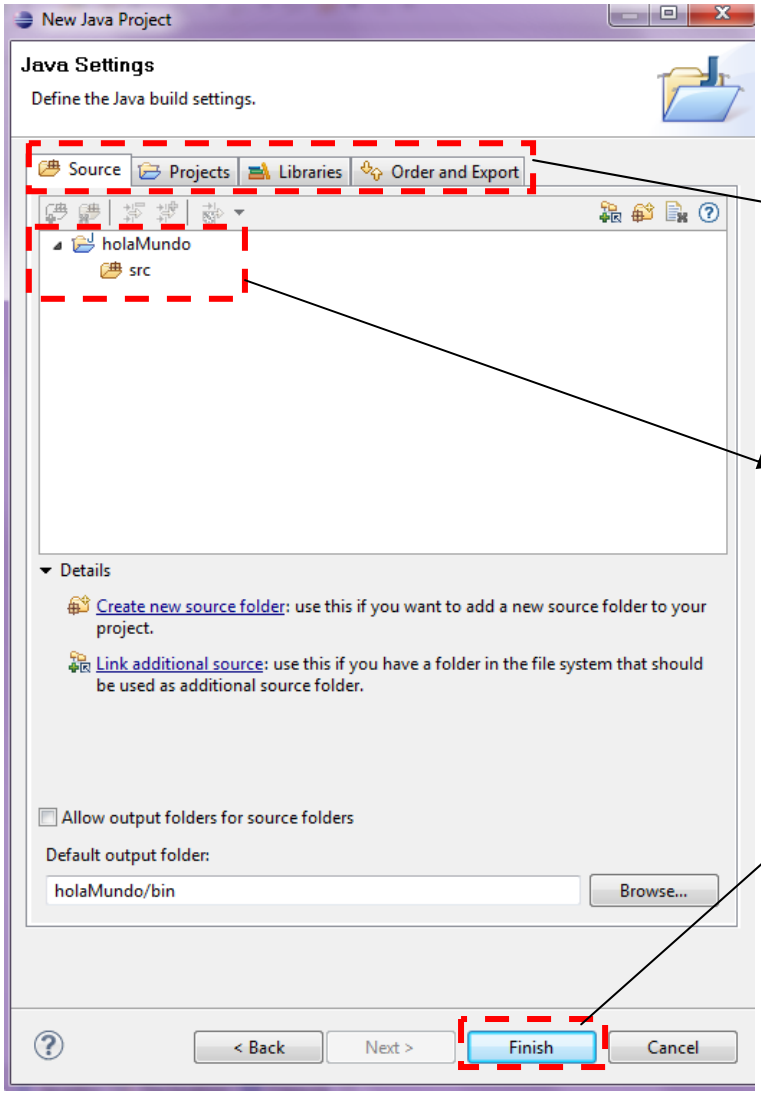


Figura 6: Asistente para la creación de un proyecto (parte 1/2)



Desde estas pestañas podremos configurar diferentes opciones para el "Build" del proyecto. Por ejemplo, librerías externas a utilizar, proyectos a utilizar junto al que estamos creando… etc.

Por defecto crea un directorio dónde programar las fuentes, "src". Se podrían añadir más directorios desde la opción "Create new source folder" que aparece abajo. Sin embargo, una forma de trabajar limpiamente es realizar subdirectorios desde el directorio src.

Terminamos de configurar el proyecto

Figura 7: Asistente para la creación de un proyecto (parte 2/2)

Al finalizar el asistente anterior, desde la ventana de proyectos, ahora veremos nuestro proyecto creado.

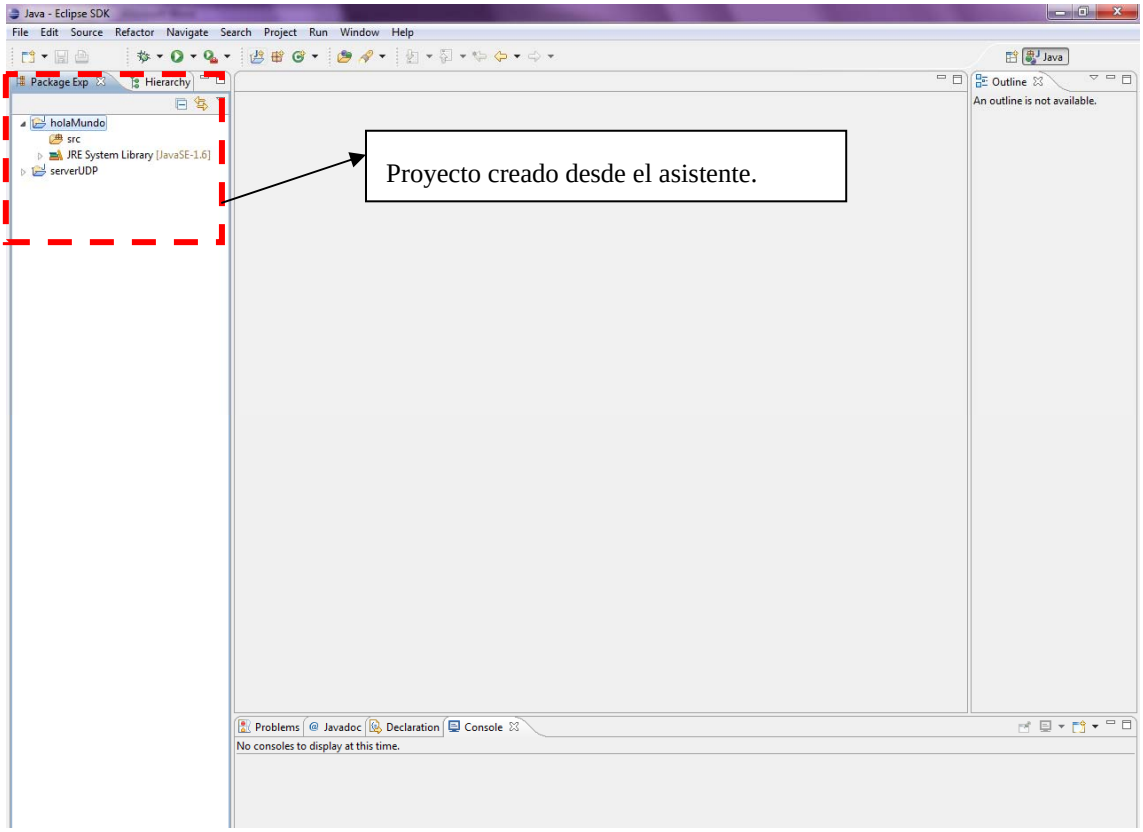
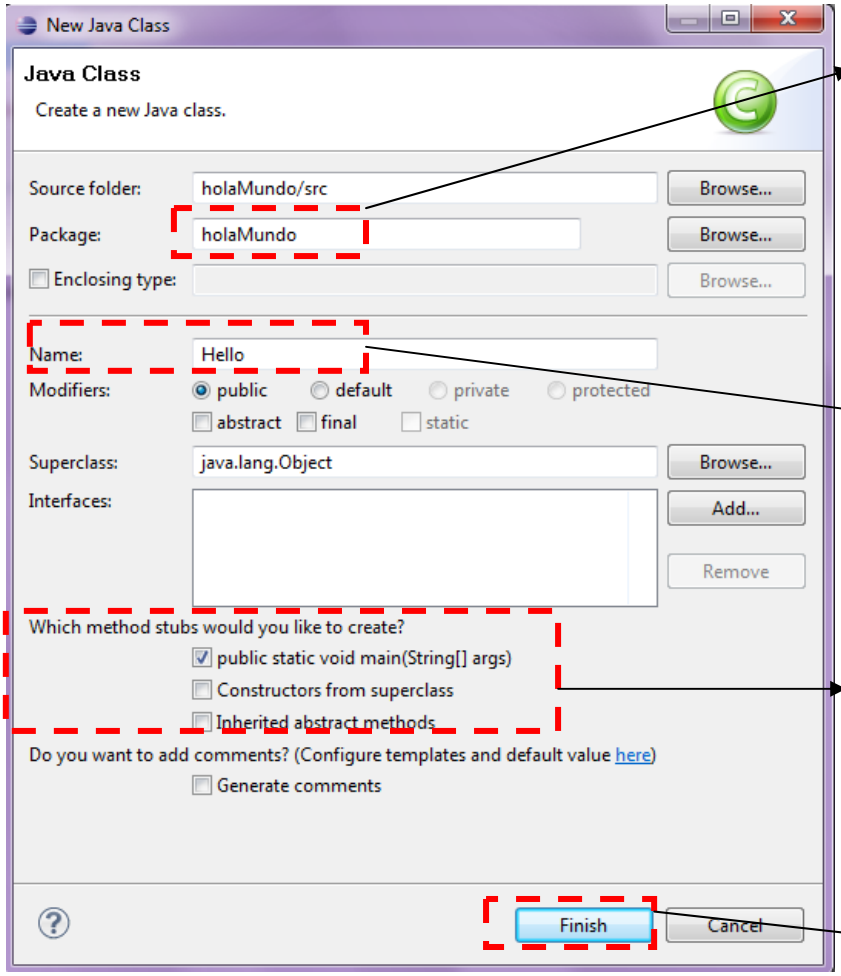


Figura 8: Proyecto creado desde el asistente

La ventana de proyectos, tiene un menú contextual, al cuál se accede con un clic derecho, desde el que se puede crear, importar y exportar proyectos, cambiar las propiedades del proyecto, crear y modificar los directorios, acceder a plantillas para la creación de nuevas clases, crear nuevos proyectos, etc.

Para continuar desarrollando nuestro primer proyecto vamos a crear una clase desde el asistente. Para ello, seleccionamos la carpeta “src” de nuestro proyecto y accedemos a su menú contextual desde el botón derecho del ratón. Desde ahí, igual que anteriormente, seleccionamos New -> Class. Para crear la clase seguimos los pasos que se muestran en la siguiente figura:



Nombre de nuestro nuevo "Package". El Package en Java sirve para agrupar clases afines, es decir, equivale a una librería con sus diferentes métodos de clases y objetos. De esta forma podremos importar nuestras clases a un nuevo proyecto o simplemente mantener modulado el nuestro.

Nombre para la nueva clase. Automáticamente el fichero fuente se llamará <nombreclase>.java

Cómo vamos a crear una clase con el método main para ejecutarla, seleccionamos la primera opción.

Además como nuestra clase es sencilla, no necesitamos definir métodos abstractos. Por lo que deseleccionamos la última opción, "Inherited abstract methods"

Seleccionamos "Finish" para terminar

Figura 9: Creación de una nueva clase

Una vez creada la clase estamos en disposición de escribir su código. Para ello, vamos a definir un sencillo código que simplemente muestre el mensaje "Hola Mundo" por la consola. En la siguiente figura se muestra la ventana de edición de código.

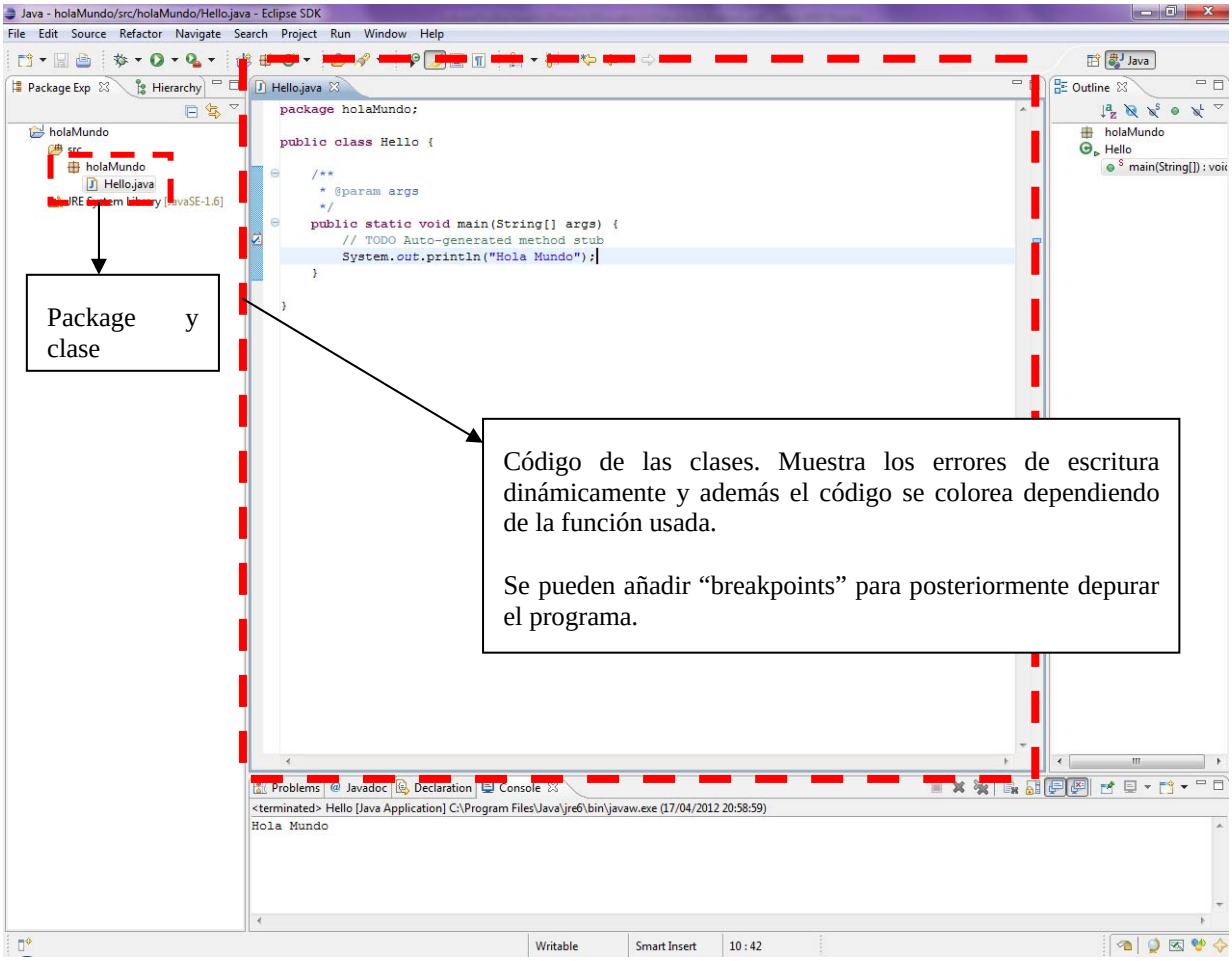


Figura 10: Ventana de edición de código

El código que vamos a definir es el siguiente. Vea que la estructura de la clase la ha creado el propio editor, solamente nos toca escribir una línea que es la que aparece en cursiva:

```
package holaMundo;

public class Hello {

	/**
	 * @param args
	 */
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("Hola Mundo"); // Código a generar, el resto es
generado por el asistente
	}

}
```

Una opción interesante es el auto-completado de código mientras se escribe. Para ello mediante Ctrl+Space se mostrarán los métodos para una clase. Por ejemplo, desde el anterior código escrito, si escribimos "System." y esperamos un poco, o bien pulsamos Ctrl+Space se nos mostrará el método "out".

Una vez definida una clase, en la ventana de la derecha se mostrará los métodos definidos para esa clase.

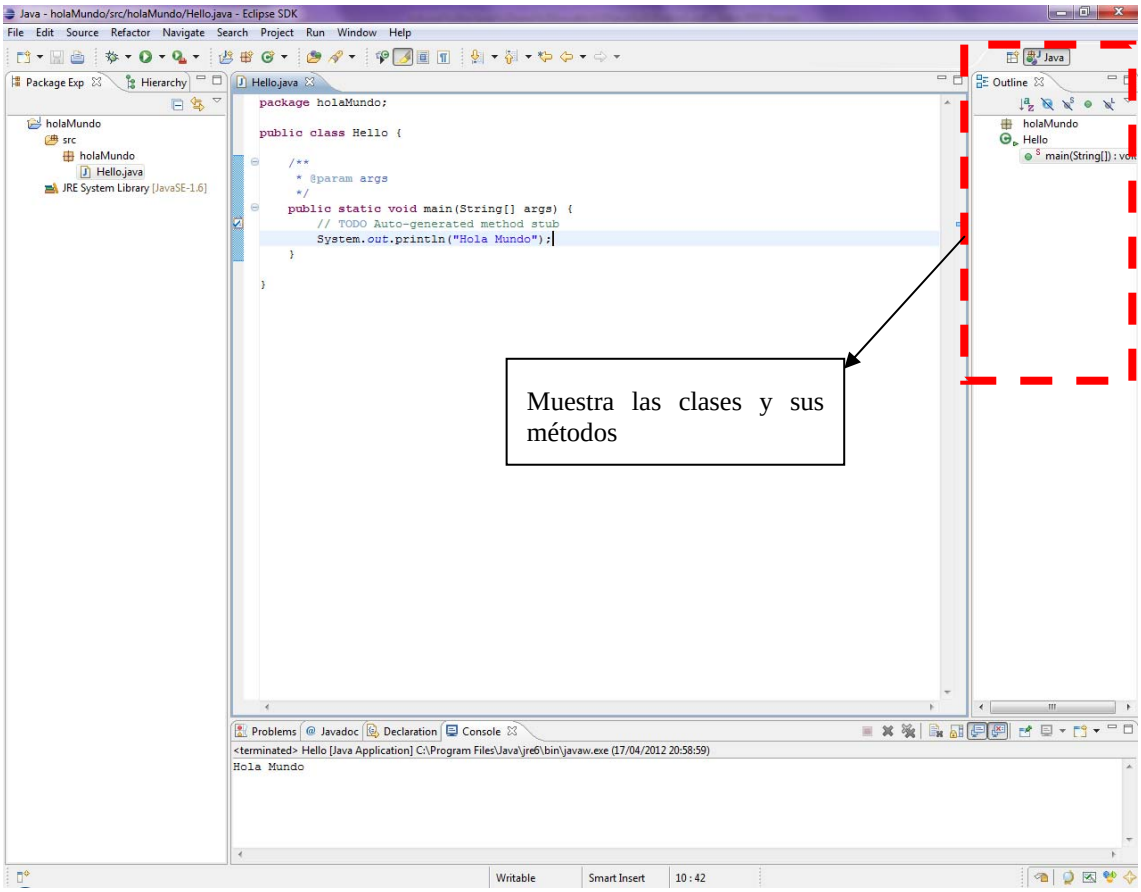
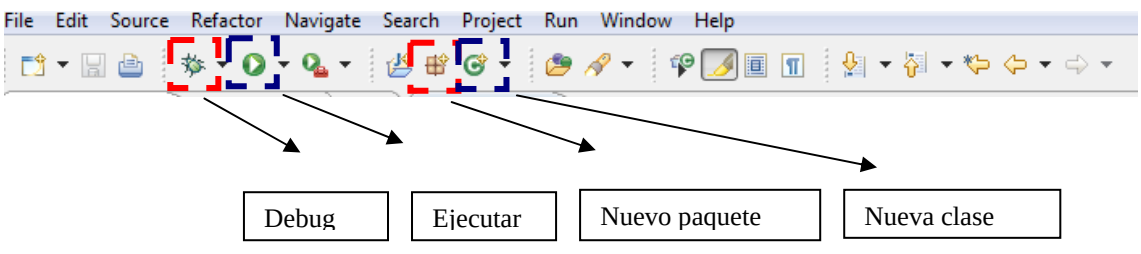


Figura 11: Ventana de métodos de las clases

Ya hemos definido la clase, ahora para compilarla y ejecutarla se van a explicar las principales opciones accesibles desde la barra de herramientas:



Desde este menú de momento vamos a utilizar la opción de ejecutar. Al seleccionar esta opción nuestro proyecto se compilará automáticamente. Una forma rápida para compilar, sin ejecutar, es mediante las teclas **Ctrl+B**. Igualmente se puede compilar desde el menú Project>Build Working Set (aunque por defecto ya está activada esta opción para hacerlo automáticamente) Otra forma es desde la ventana de proyecto, sobre el proyecto pulsar botón derecho y escoger la opción "Clean". Mediante esta opción las clases viejas son borradas y el proyecto entero es recompilado, si se tienen muchas clases el proceso puede ser un poco costoso.

Al ejecutar el proyecto se abrirá una ventana de consola en la parte inferior mostrándonos el mensaje "Hola Mundo", mostrando la salida que obtendríamos si ejecutáramos el programa desde el terminal con el comando java Hello. Esta ventana puede ser previamente abierta desde el menú Window ->Show View -> Console. En la siguiente figura se muestra dónde se abre por defecto la vista de consola.

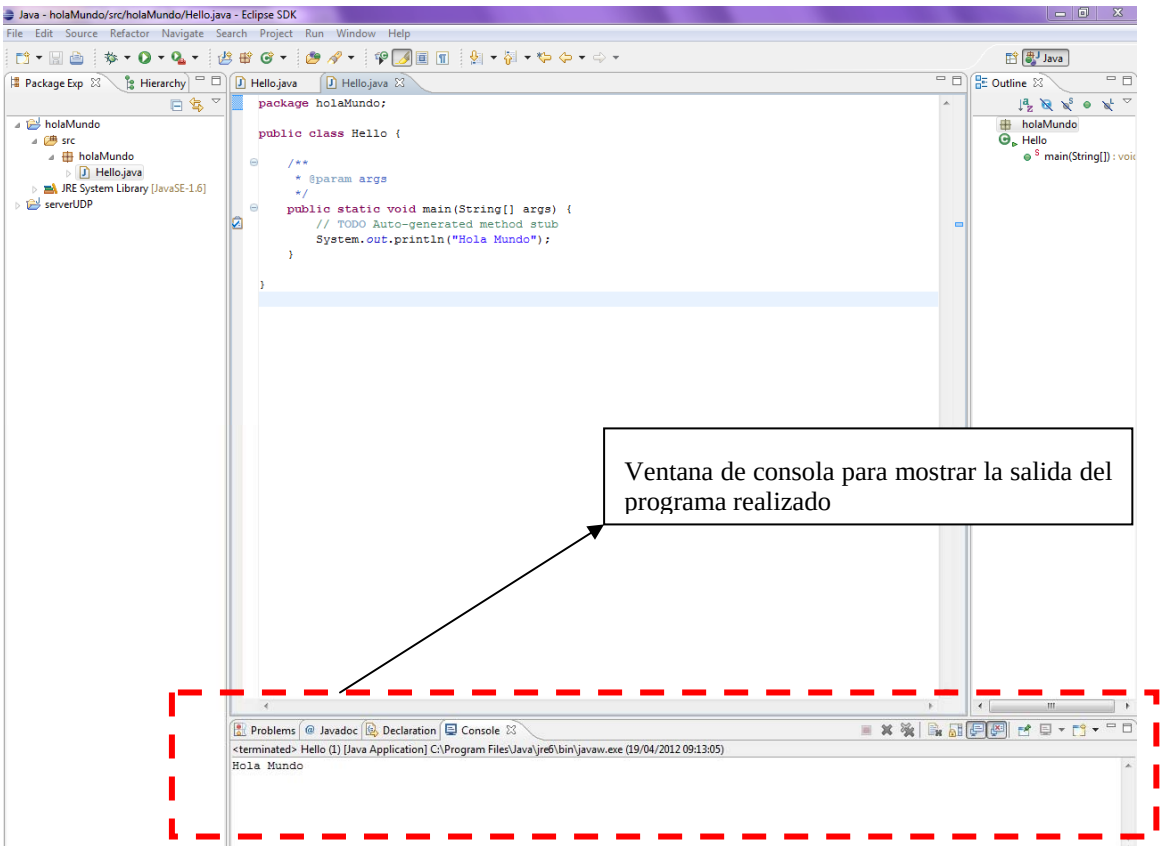


Figura 12: Vista de consola

Con los pasos realizados anteriores habríamos terminado nuestro primer programa. Una opción interesante a tener en cuenta es la exportación del proyecto para ejecutarlo en otros sistemas. Para ello, desde el menú contextual de la ventana de proyectos, se seleccionará la opción "Export". Mediante esta opción el proyecto puede ser exportado en un .jar que incluya todas las librerías y pueda ser ejecutado desde otras máquinas sin tener que utilizar ningún IDE y ahorrándonos el esfuerzo de crear un Makefile.

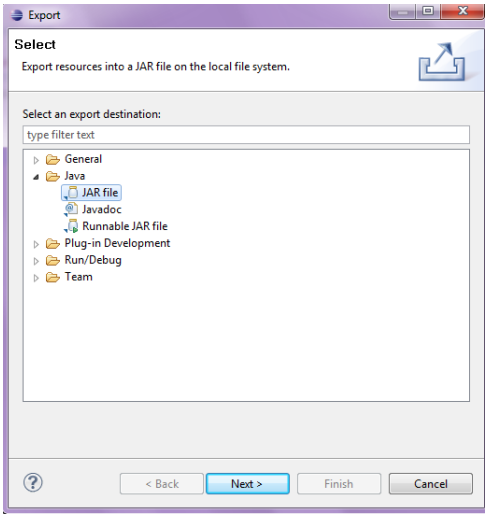


Figura 13: Exportar nuestro proyecto para ejecutarlo en otra máquina

4 Cliente TCP

Implemente una aplicación cliente TCP que tenga como parámetros de línea de comandos 2 argumentos, el nombre del servidor al que conectarse y su puerto. Una vez conectado al servidor, debe mandarle las líneas de texto introducidas por el usuario por teclado con cada pulsación de tecla INTRO. Por cada línea de texto enviada, el cliente leerá una cadena de texto que le mande el servidor, y la mostrará en pantalla. El proceso se repite hasta que el usuario teclee el carácter "." únicamente en una línea, en cuyo caso cerrará la conexión y terminará el programa.

En el IDE eclipse puede introducir los parámetros de línea de comandos desde el menú "Run> Run Configurations" en la pestaña "Arguments" donde dice "Program Arguments".

Verifique el funcionamiento de su cliente usando como servidor TCP el netcat (nc). Pruébelo inicialmente teniendo servidor y cliente en la misma máquina. Luego ponga el servidor en otra máquina (haga ssh a cualquier otra máquina del laboratorio para lanzar el servidor).

Checkpoint D: Avisar al responsable de prácticas cuando haya completado las prácticas hasta este apartado. No se quede bloqueado, mientras tanto avance con las siguientes secciones.

5 Servidor TCP

Implemente una aplicación servidora TCP que tenga como parámetro de línea de comandos un argumento, el número de puerto en el que debe escuchar. El servidor va a recibir una conexión de un cliente y otro cliente no se podrá conectar hasta que finalice el servicio con el primer cliente. El servidor va a ser capaz de dialogar con el cliente desarrollado en la sección anterior: va a recibir una línea de texto y según el contenido de esa línea va a devolver otra línea de texto, repitiendo el proceso mientras el cliente siga conectado. Una vez desconectado el cliente, podrá recibir la conexión de otro cliente.

A los mensajes de texto que manda el cliente, el servidor va a responder según el contenido del mensaje de entre los siguientes:

- "hola", el servidor contesta con "buenos días"
- "hora", el servidor contesta con la fecha y hora actual
- "ip cliente", el servidor contesta con la dirección IP del cliente
- "ip servidor", el servidor contesta con la dirección IP del servidor
- Para el resto de mensajes del cliente, el servidor devuelve el número de caracteres del mensaje que envió el cliente.

Verifique el funcionamiento con cliente y servidor en la misma y diferentes máquinas.

Utilice el tcpdump para verificar el funcionamiento de la conexión entre cliente y servidor. Identifique cómo se encapsulan los mensajes de texto en los paquetes TCP y quien manda el primer paquete de SYN y FIN.

Checkpoint E: Avisar al responsable de prácticas cuando haya completado las prácticas hasta este apartado. No se quede bloqueado, mientras tanto avance con las siguientes secciones.

6 Cliente UDP

Implemente una aplicación cliente UDP que lea líneas de texto introducidas por el usuario por teclado con cada pulsación de tecla INTRO. En esas líneas el usuario introducirá una dirección IP y puerto del servidor, con la forma por ejemplo:

```
10.1.1.1 9999
```

Tras ello el cliente mandará un número entero de 4 bytes (no cadena de texto) al servidor con esa dirección IP y puerto, un entero que corresponderá al número de línea introducida por el usuario: 1 para la primera línea, 2 para la segunda línea, etc. El número entero que se envía irá en un datagrama UDP hacia el servidor. El servidor retornará una línea de texto con el mismo número enviado por el cliente, que entonces el cliente imprimirá en pantalla. Tenga en cuenta que los paquetes se pueden perder. Cuando el usuario introduzca por teclado una línea en blanco se finalizará el programa.

Verifique el funcionamiento de su cliente usando como servidor UDP el netcat (nc) en la misma y diferentes máquinas que el cliente, así como con el tcpdump.

Para realizar el envío de los datos como entero a través del socket UDP creado se realizará a través de la clase ByteBuffer. Esta clase permite mediante el método allocate dimensionar el espacio en bytes que va a tener el buffer. Mediante el método putInt permite la introducción de una variable de tipo Int que se guardará en el buffer y mediante el método array permite transformar la cadena de bytes en un array de bytes. La forma de utilizarla para transformar una variable Integer llamada entero en un array de bytes de 4 bytes es la siguiente

```
ByteBuffer.allocate(4).putInt(entero).array()
```

Para realizar la operación inversa se utiliza la misma clase ByteBuffer que transforma el array de bytes utilizando el método wrap de la clase ByteBuffer y también se realiza una transformación a Integer mediante el método getInt. La forma de hacerlo es la siguiente considerando en el ejemplo que la variable bytesrecibidos se ha leído mediante el DatagramPacket:

```
ByteBuffer.wrap(bytesrecibidos).getInt();
```

Checkpoint F: Avisar al responsable de prácticas cuando haya completado las prácticas hasta este apartado. No se quede bloqueado, mientras tanto avance con las siguientes secciones.

7 Servidor UDP

Implemente una aplicación servidora UDP que tenga como parámetro de línea de comandos un argumento, el número de puerto en el que debe escuchar. El servidor va a recibir mensajes del cliente desarrollado en la sección anterior, es decir, recibir un número entero de 4 bytes y devolver un mensaje de texto con ese número.

Verifique el funcionamiento con cliente y servidor en la misma y diferentes máquinas. Utilice el tcpdump para verificar el funcionamiento y entender cómo se encapsulan los mensajes en datagramas UDP. ¿Puede saber el servidor si el programa cliente ha parado?

Checkpoint G: Avisar al responsable de prácticas cuando haya completado las prácticas hasta este apartado. No se quede bloqueado, mientras tanto avance con las siguientes secciones.

8 Proyecto de aplicación de red

Según el tiempo que haya dedicado a las secciones anteriores vamos a plantearnos un proyecto de programación de una aplicación de red original más o menos ambicioso. Se trata de que vosotros mismos hagáis una propuesta de qué tipo de aplicación os interesaría, fijar los requisitos de la misma e implementarla para la última de las sesiones de prácticas reservadas para este guión.

Algunas ideas de aplicaciones a desarrollar podrían ser:

- Un cliente-servidor de chat que sea capaz de recibir conexiones de múltiples conexiones de clientes simultáneas, y que cada mensaje que mande un cliente lo envíe a todos los demás clientes.

- Un scanner de puertos TCP/UDP que dada una subred IP, para cada máquina de esa subred escanee un rango de puertos en la misma en busca de aplicaciones servidores TCP/UDP escuchando en esos puertos.

- Una aplicación cliente-servidor capaz de medir el ancho de banda disponible entre los dos.

- Una aplicación cliente-servidor capaz de medir el retardo, jitter y pérdidas en el camino entre ambos.

- Un servidor bot, que sea capaz de mantener una "conversación" con un cliente que le mande diálogos en texto.

- Una aplicación que chequee cambios en un conjunto de páginas web predefinidas y avise cuando se ha producido algún cambio en el contenido de las mismas.

- Un servidor de ficheros, que permita solicitar un fichero y se reciba el contenido del mismo a la mayor velocidad posible.

- Un programa que detecte las máquinas encendidas dentro de una subred IP.

Seleccione y evalúe la dificultad de la aplicación que desee según el tiempo que tenga disponible. Para la aplicación seleccionada, especificar brevemente:

- Aplicaciones necesarias: cliente, servidor, ambas, mixta

- Parámetros de entrada para cada aplicación

- Resultados de salida de cada aplicación

- Concurrencia de la aplicación

Checkpoint pre-H: Antes de ponerse a trabajar sobre la aplicación elegida, avisar al responsable de prácticas para determinar el alcance de la misma. No se quede bloqueado, mientras tanto avance en su aplicación.

Checkpoint H: Avisar al responsable de prácticas cuando haya completado las prácticas hasta este apartado. No se quede bloqueado, mientras tanto avance con las siguientes secciones.