ACTIVIDAD GUIADA 1
Puesta a punto del entorno

Contenidos

Introducción

El primer paso consiste en descargar el entorno de trabajo necesario para desarrollar interfaces de usuario en Java.

En principio sólo necesitamos una serie de utilidades orientadas al desarrollo de aplicaciones y que incluyen un compilador de Java (**javac**), el intérprete (**java**) o el generador de documentación (**javadoc**) entre otros, y al que nos referimos de manera global como **JDK** (*Java Development Kit*).

Sin embargo, lo habitual es descargar un **IDE** (*Integrated Development Kit*) que nos facilite las tareas habituales en la elaboración de programas como la escritura de código y autocomplección, compilado, prueba, organización de proyectos…, Los dos entornos más comunes para Java son Eclipse (open source) y NetBeans (de Oracle, actual propietario de Java). Si bien no hay grandes diferencias entre ellos, nos inclinaremos por NetBeans.

En este documento se dan las instrucciones para la instalación y puesta a punto del entorno de trabajo. Una opción, para no interferir con el software que se usa en otros módulos (que podrían necesitar otra versión del JDK), es preparar una máquina virtual con tu S.O. preferido y realizar la instalación en la misma.

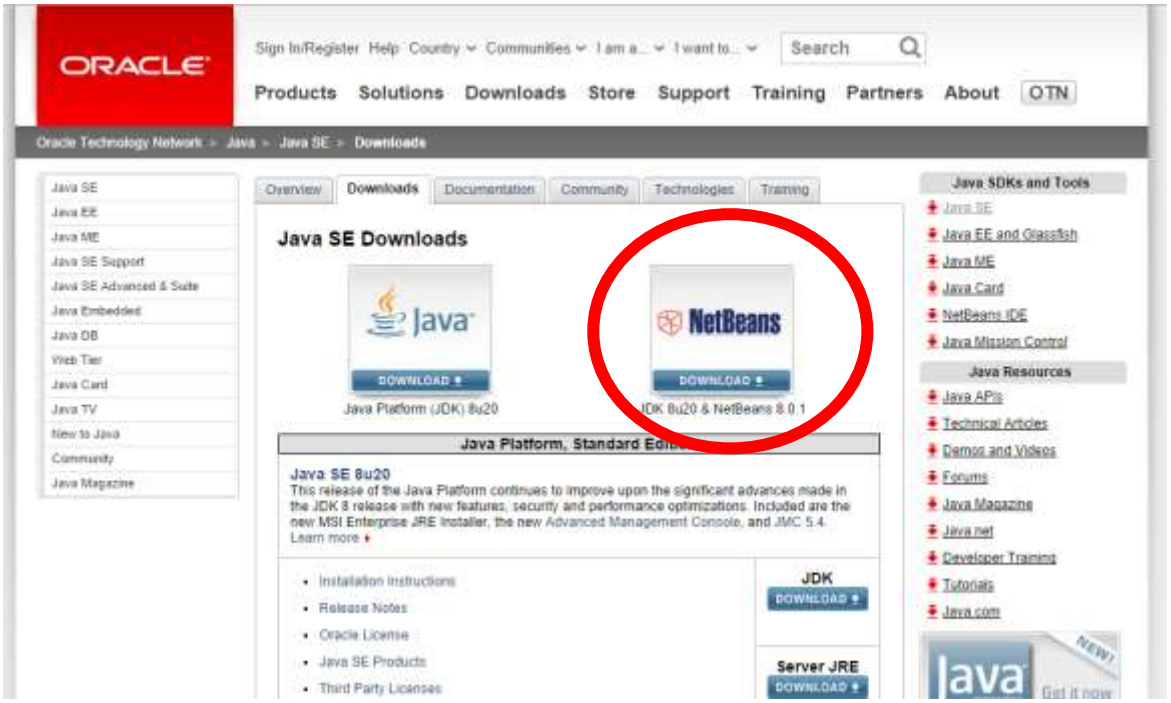
Descarga e instalación de JDK y NetBeans

Vamos a la siguiente dirección de Internet: http://www.oracle.com/java

Escogemos la sección *Downloads* y dentro de ella *Java SE*

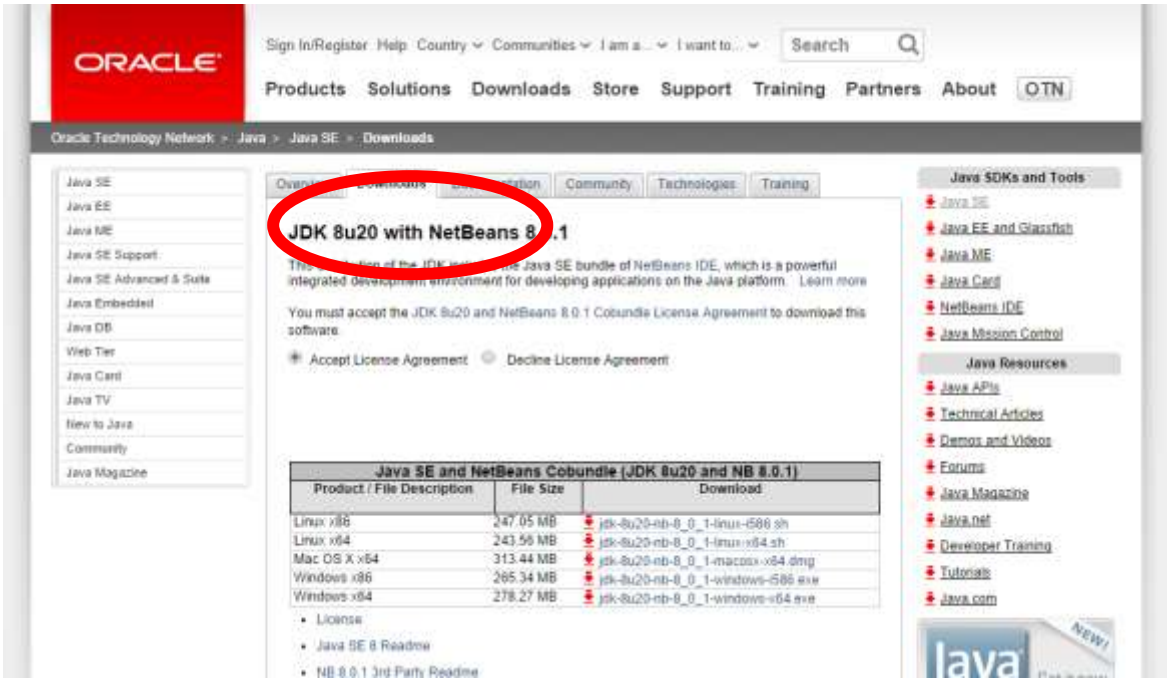
Vemos que tenemos dos opciones:

- Con el enlace de la izquierda podemos bajar la última versión del JDK (Java *Development Kit*) de Java, necesario para desarrollar programas en Java.
- Con el enlace de la derecha podremos bajar tanto el JDK como NetBeans, que es el IDE que vamos a utilizar para desarrollar interfaces con Java. Pulsamos por tanto este enlace.



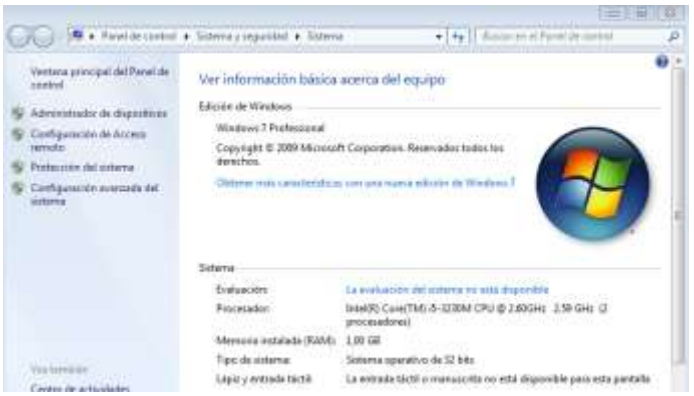
Aunque ya tengamos una versión del JDK instalado, es conveniente descargar la última versión para evitar problemas de compatibilidad con versiones anteriores.

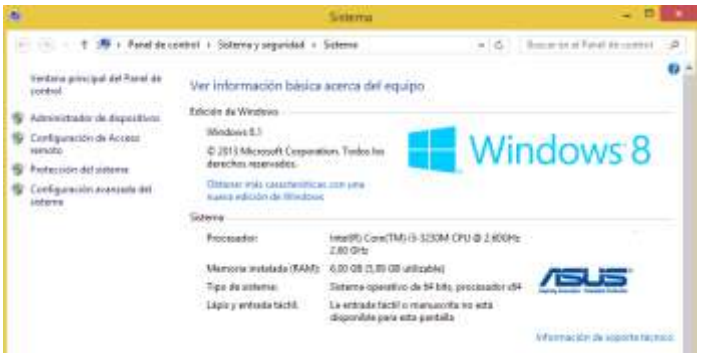
Aceptamos en contrato de licencia y escoger la plataforma en la que vamos a desarrollar. El IDE está disponible en entornos Linux, Mac OS X y Windows, tanto para 32 como para 64 bits.



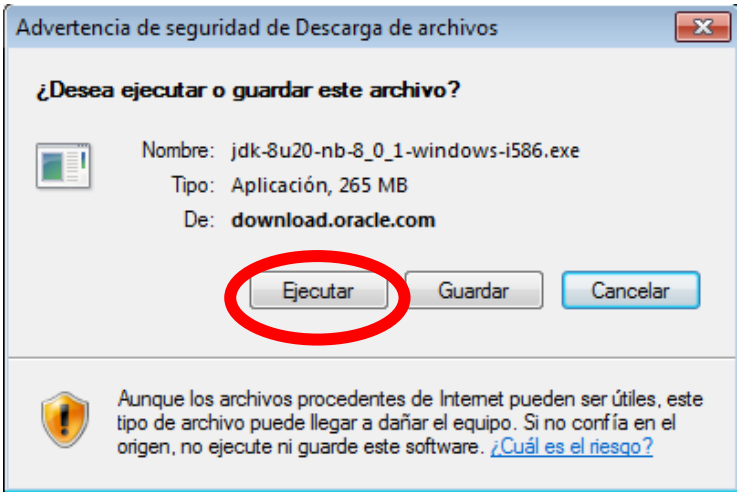
En función del Sistema Operativo que tengamos instalado descargamos el archivo correspondiente. Si disponemos de un S.O. de 64 bits, es más conveniente descargar dicha versión.

Podemos comprobar la versión del S.O. desde *Sistema*. A continuación se muestran un S.O. Windows 7 de 32 bits, y un Windows 8 de 64 bits:



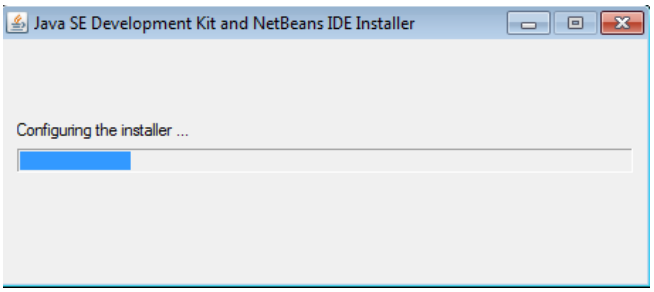


Dependiendo del navegador usado se nos ofrecerán distintas opciones. Podemos guardar el archivo en la ruta que queramos, lo importante es que una vez que se haya descargado pulsemos sobre él para ejecutarlo. El archivo ocupa 256 MB, luego dependiendo de nuestra conexión podría tardar un rato en descargarse:

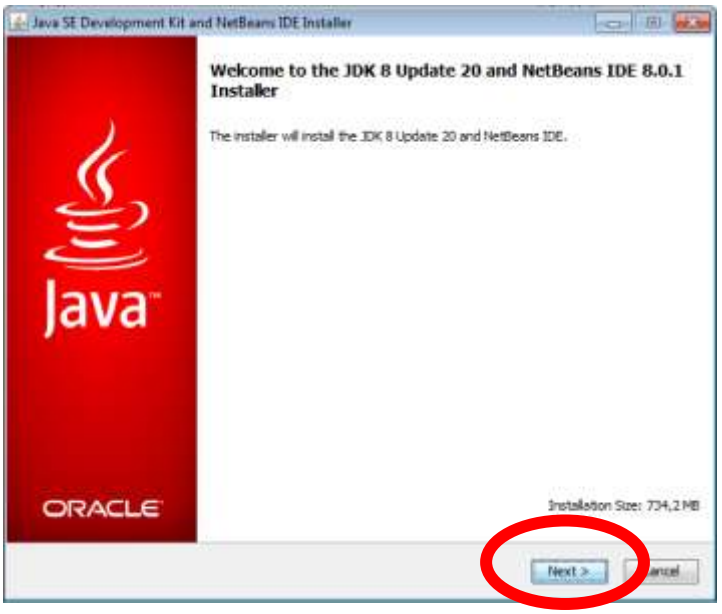


En función de la configuración del S.O. se nos podría hacer una advertencia ¿Desea permitir que el programa realice cambios en el equipo? Y contestaríamos que sí.

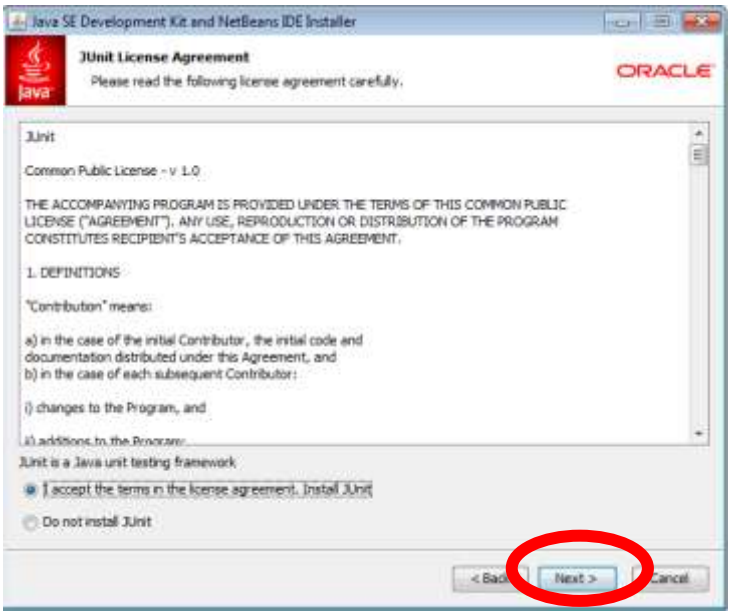
Y aquí comienza la instalación:



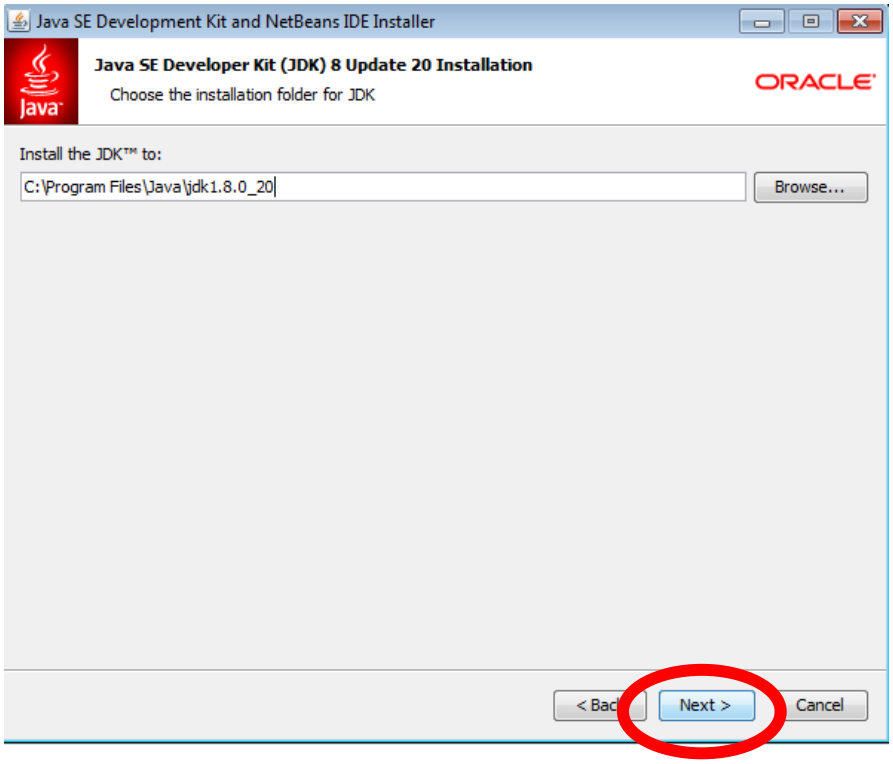
Los pasos necesarios para configurar la instalación son muy sencillos. Comenzamos el proceso:



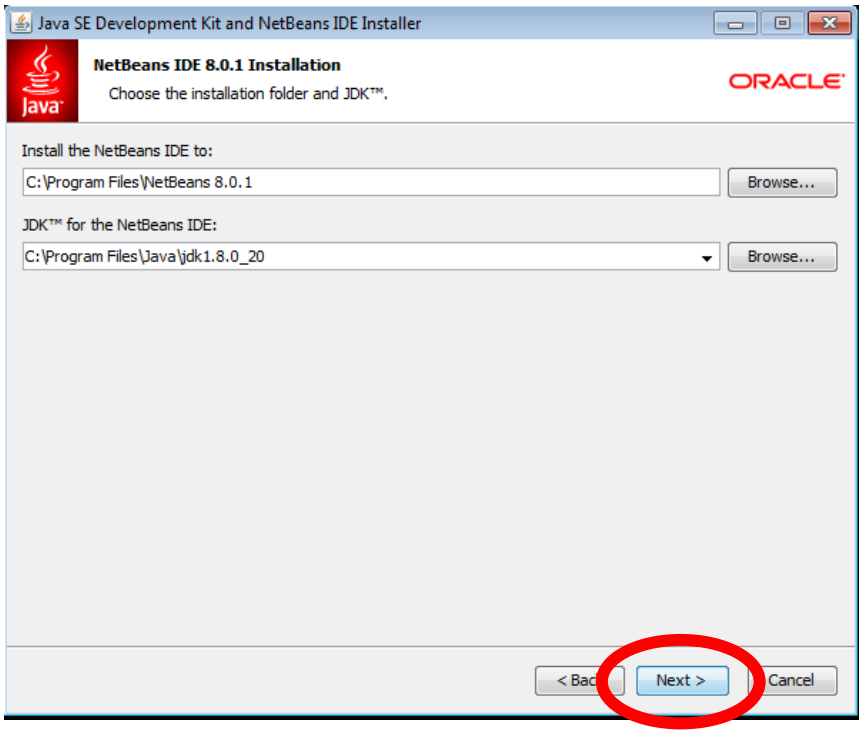
Aceptamos la licencia e instalamos el framework JUnit, que utilizaremos para realización de pruebas unitarias:



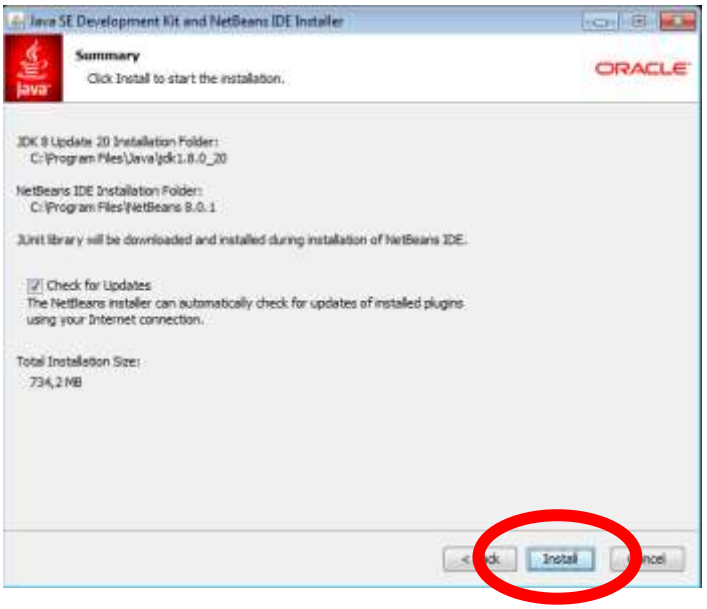
Escogemos la ruta de instalación del JDK (ruta por defecto):



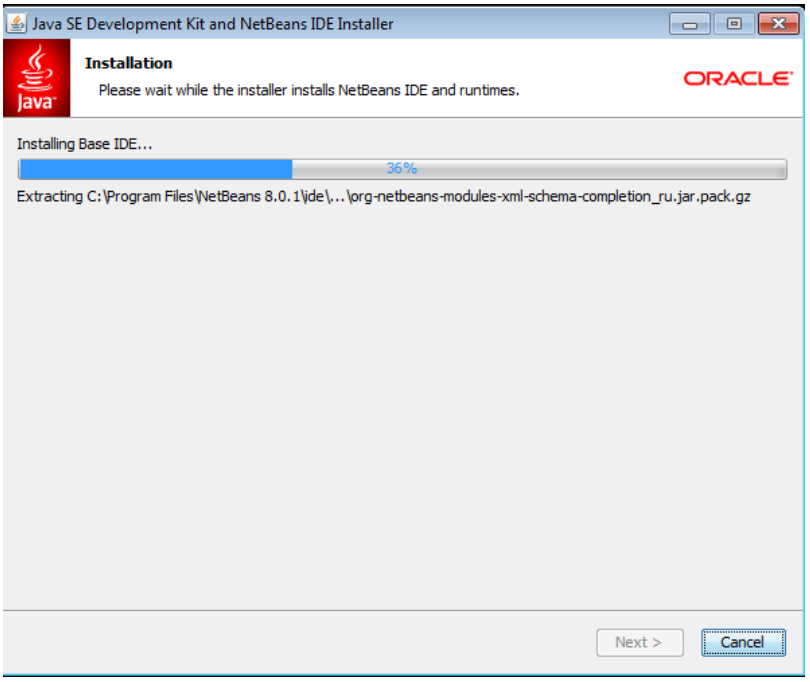
Y la ruta de instalación de NetBeans, que también es la ruta por defecto. Comprobaremos que la ruta del JDK que vamos a usar desde NetBeans es la misma que hemos configurado en el paso anterior:



Y finalizamos el proceso de configuración de la instalación. Se nos avisa de que vamos a necesitar casi 750 MB de disco duro. Por otra parte, el programa buscará de manera automática actualizaciones, para avisarnos cuando salga alguna nueva versión con mejoras, correcciones de errores, etc.



La instalación propiamente dicha puede llevar un rato:



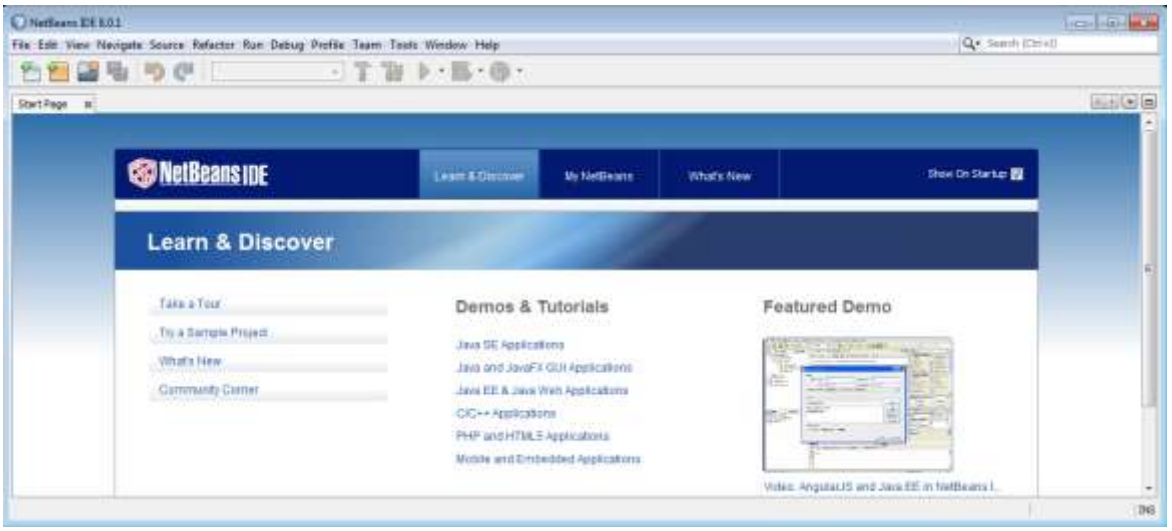
La última pantalla nos indica que ha finalizado la instalación. Si lo deseamos, podemos enviar información anónima para ayudar a mejorar la herramienta.

Creación de un Proyecto

Para comprobar que todo está funcionando bien, vamos a hacer un programa de prueba en Java.

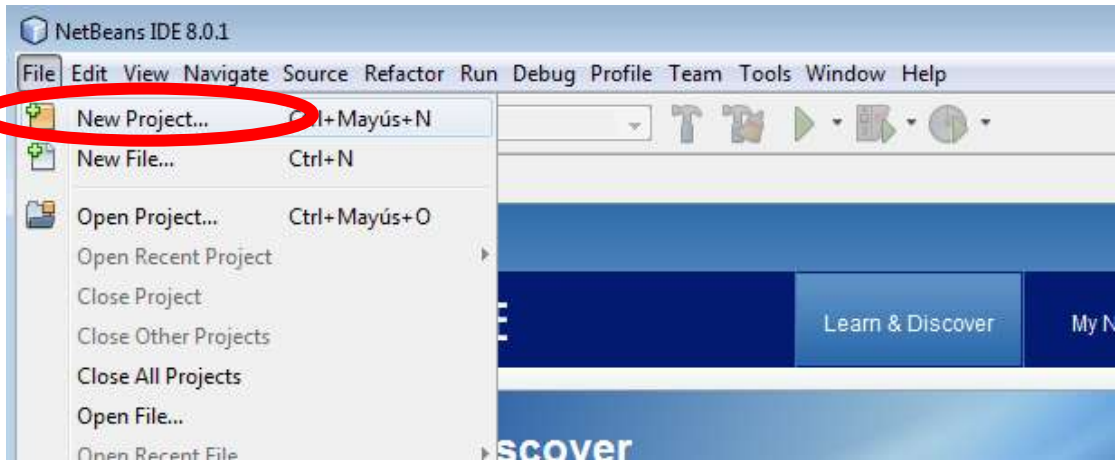
Para acceder al IDE, vemos que se ha creado un atajo en el escritorio:

Y veremos que se abre la interfaz del mismo. Se trata únicamente de una página de inicio donde se nos sugieren aplicaciones, cómo empezar, etc.

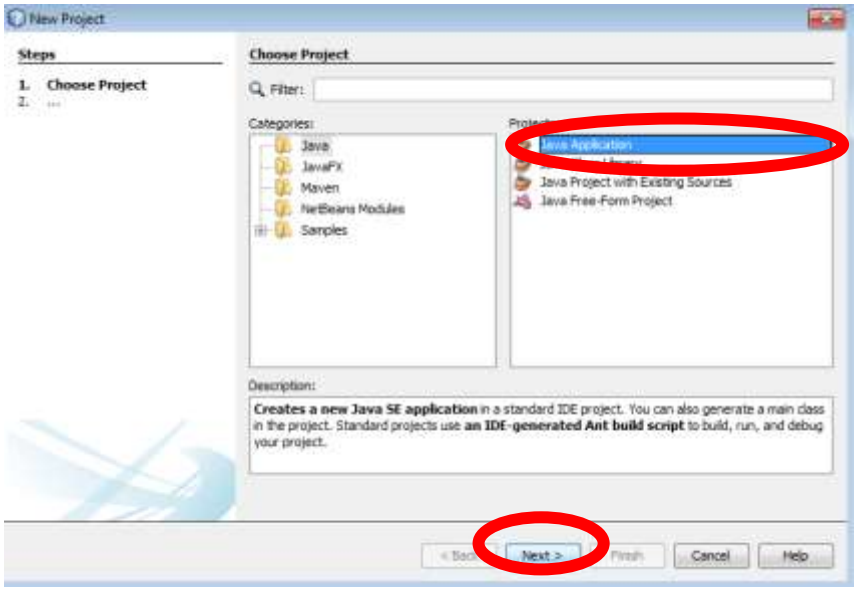


NetBeans maneja el concepto de proyecto, que agrupa todas las clases Java correspondientes a un programa determinado, así como aspectos de su configuración, librerías necesarias, etc.

Vamos a crear un nuevo proyecto en NetBeans para comprobar que todo funciona bien. Escogemos *File > New Project…*

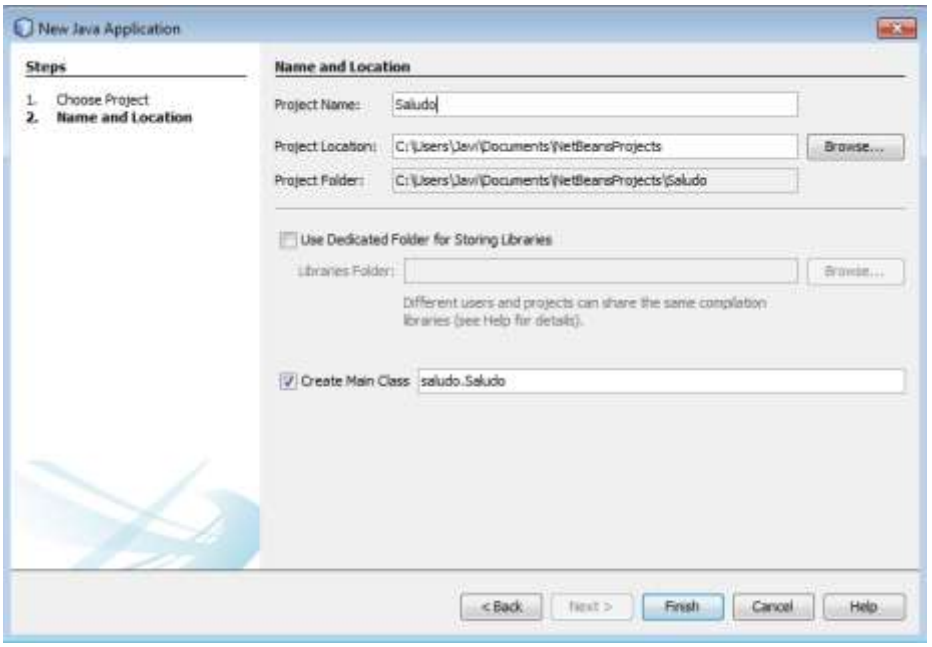


El primer paso es escoger el tipo de proyecto. Vemos que en la parte superior tenemos un filtro para escoger el tipo, pero siempre escogeremos *Java Application*, que nos permite hacer una aplicación Java. A continuación pulsamos *Next*:



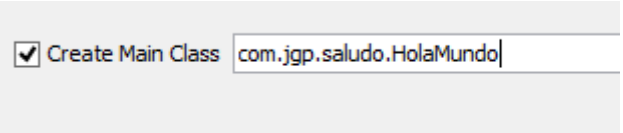
A continuación vamos a introducir el nombre el proyecto. Vamos a llamar al proyecto HolaMundo, vemos que se autocompletan los siguientes elementos:

- La carpeta donde se van a ubicar los archivos del proyecto. Vemos que por defecto es una subcarpeta de *Documents/NetBeansProjects* con el nombre del proyecto. En principio no vamos a necesitar cambiar esta ubicación.
- Se crea una clase principal, por defecto con el propio nombre del proyecto sin espacios (en nuestro caso `Saludo`). Esta clase es la que contendrá el método `main()`, que se invoca al ejecutar el proyecto. Además, la clase se ubica dentro de un paquete con el propio nombre del proyecto (en minúsculas).



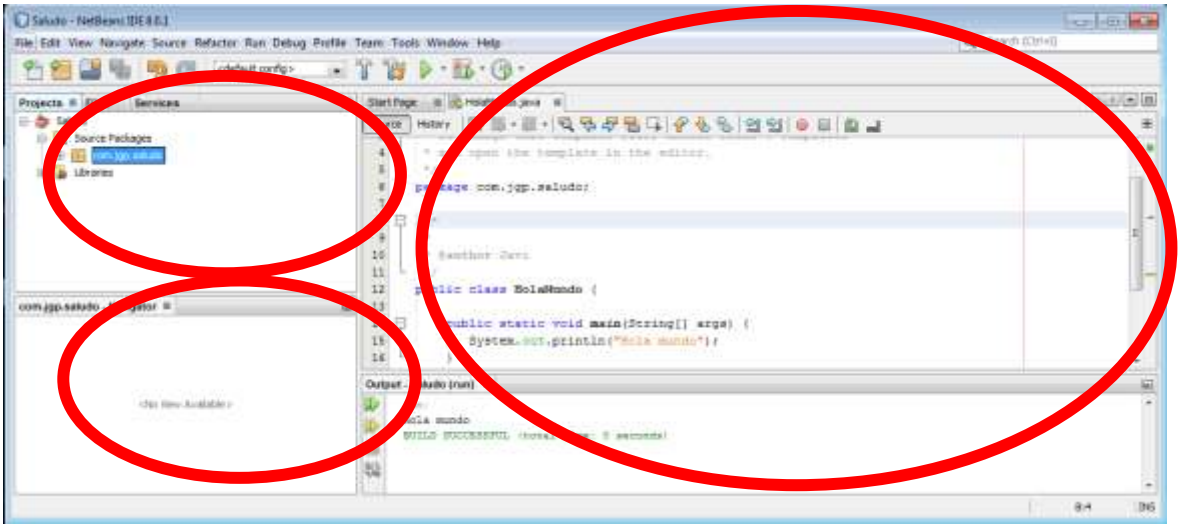
Podemos renombrar la clase o el paquete sin problema, no tienen por qué tener ninguna relación con el proyecto

En nuestro caso, vamos a renombrar el nombre de la clase principal para que denomine **`HolaMundo`**. Una práctica habitual es renombrar los paquetes para que sigan la estructura **`com.inicialesdetunombre.proyecto`** Por ejemplo:



A continuación pulsamos *Finish.* Veremos que se crea el proyecto y se cargan los paneles asociados:

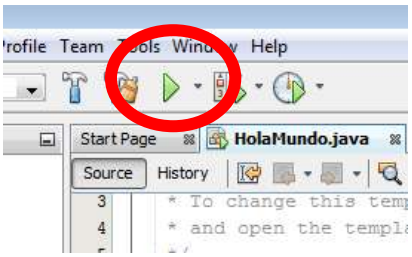
- En la parte derecha de la pantalla tenemos el panel de código. Se muestra el código de la clase que queramos visualizar y editar. Tenemos varias pestañas para alternar la edición de distintas clases.
- En la parte superior izquierda tenemos el panel de proyecto, donde podemos consultar los archivos, clases y paquetes que forman parte del mismo.
- En la parte inferior izquierda vemos el panel de navegador, donde se muestran las propiedades y métodos que tenemos en nuestra clase. Cuando nuestras clases crezcan en tamaño, es importante utilizar este panel para acceder de manera rápida al código de las mismas.



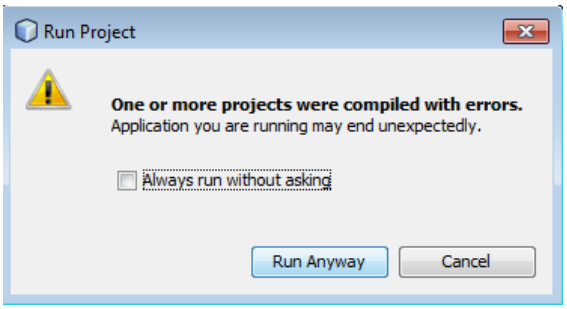
Para comprobar que todo funciona bien, vamos a meter algo de código en la clase de ejemplo. Añadimos al `main()` algo de código destinado a mostrar un mensaje por pantalla:

```
public class HolaMundo {

    public static void main(String[] args) {
        System.out.println("Hola mundo");
    }

}
```

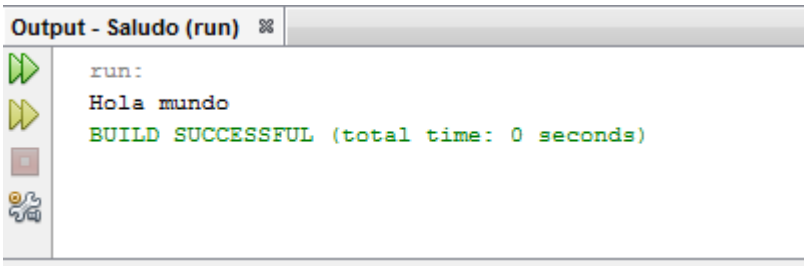
Para compilar y ejecutar la clase tenemos varias opciones, pero la más rápida es pulsar *Run Project* (atajo F6). Es el botón con un icono de *Play*:



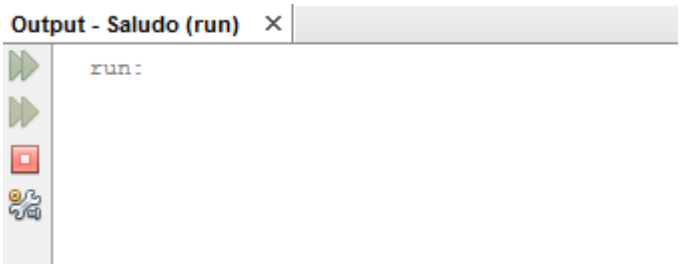
Si el código tiene errores debido a la sintaxis (lo que no debería suceder en este programa tan simple), veremos que se nos avisa de que hay algún error de compilación). Podemos ejecutarlo ("*Run Anyway*") pero no es lo recomendable, deberíamos volver al panel de código y comprobar qué errores se van produciendo.



Si el código no tiene errores, veremos que bajo el panel de código aparece un nuevo panel donde se nos informa del resultado de ejecutar el método `main()` de la clase `HolaMundo`, tanto en lo referente a la compilación (BUILD SUCESSFUL) como en la ejecución, donde vemos el texto que imprimimos por pantalla. En este caso el programa ya ha finalizado, pues es muy simple



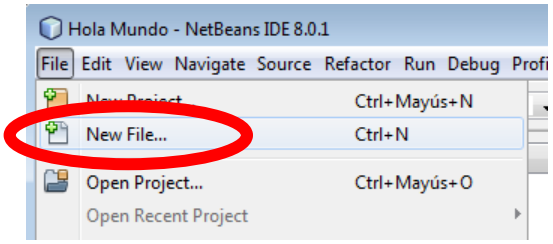
Cuando desarrollemos interfaces gráficas de usuario, se abrirá una ventana emergente, que si hemos programado correctamente debería cerrarse al pulsar el botón superior derecho (X). Si algo va mal, podemos cerrar la ventana (y finalizar el programa) pulsando el botón de *Stop*.



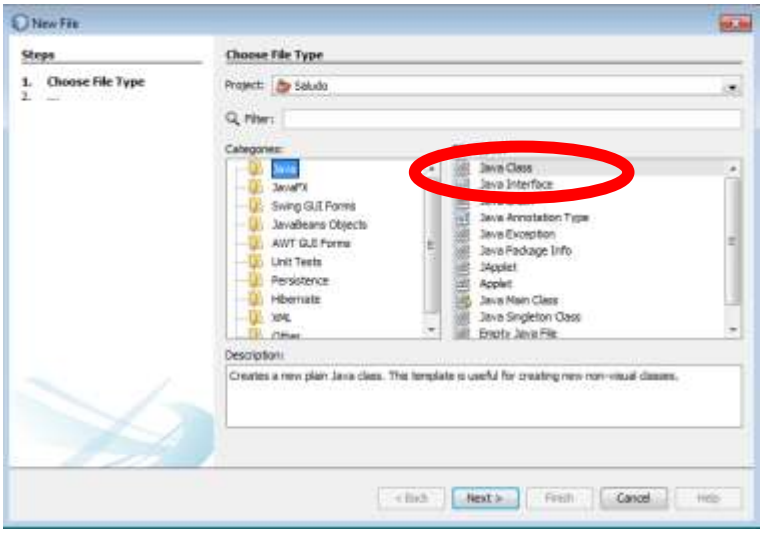
En los siguientes capítulos iremos probando sucesivos programas en Java que tendremos que ejecutar. Para ello tendremos dos opciones:

1. *Crear un único proyecto (o un proyecto por capítulo) e ir añadiendo clases Java al mismo.*

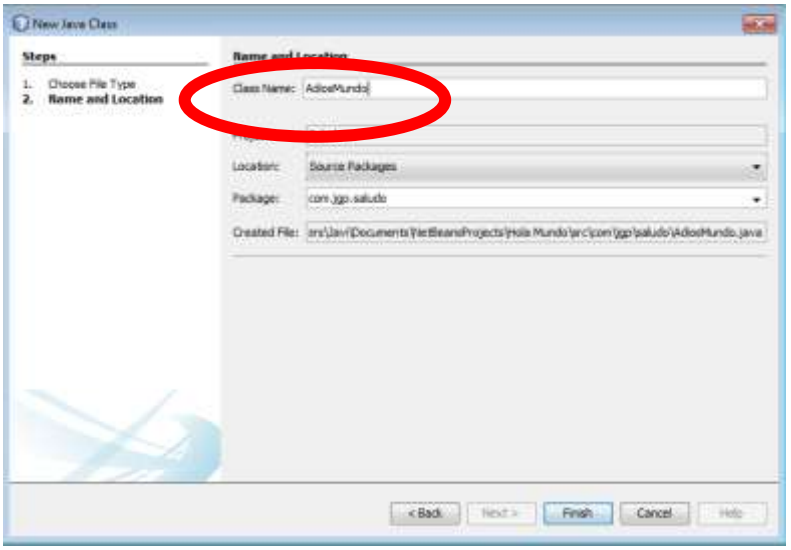
   En nuestro ejemplo, supongamos que deseamos crear una clase llamada `AdiosMundo` que queremos que se ejecute. Para crearla escogeríamos *File> New File*:



   A continuación escogemos el tipo de archivo que queremos crear, que es una clase Java (*Java Class*)



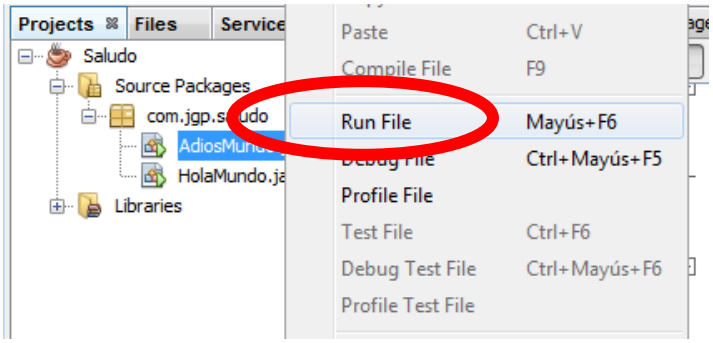
   Daremos el nombre a la clase (vemos que se respeta el nombre de proyecto que dimos en el paso anterior):



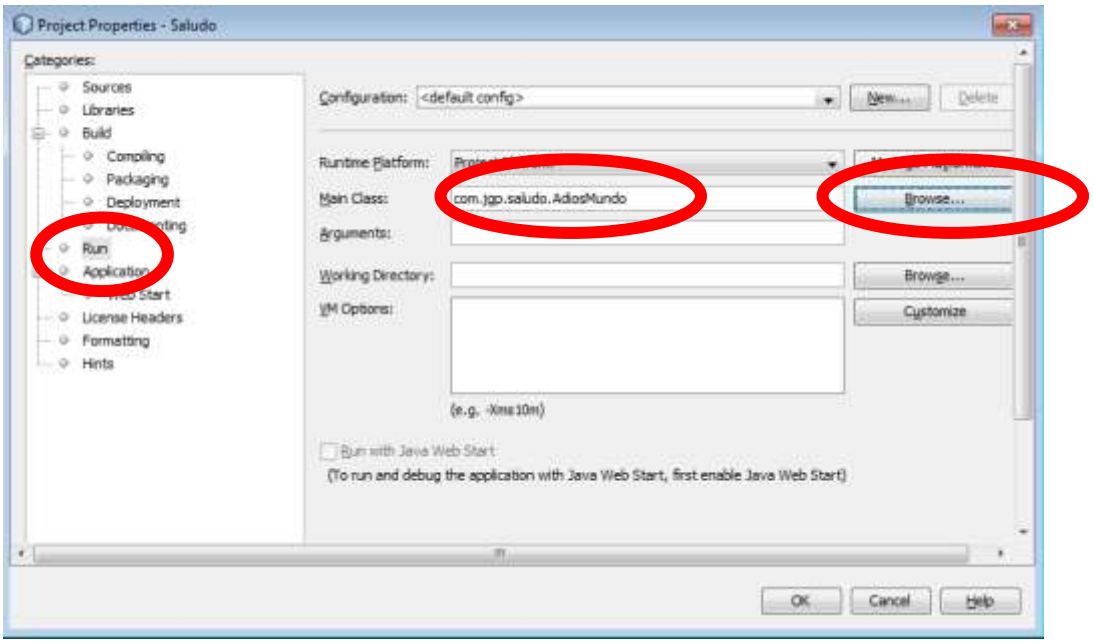
Editamos el código de la misma:

```
public class AdiosMundo {

    public static void main(String[] args){

        System.out.println("Adios mundo");
    }
}
```

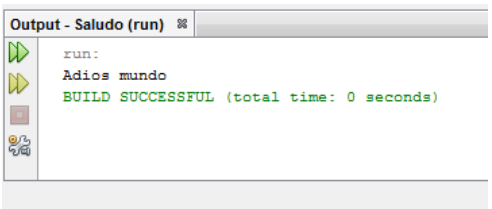
Para ejecutar la clase, podemos pulsar con el botón derecho sobre la misma desde el panel de proyectos, y escoger la opción *Run File*



Otra opción es hacerla la clase principal del proyecto. En este caso pulsamos con el botón derecho sobre el nombre del proyecto y escogemos *Properties.* En el apartado *Run*, podemos escoger la clase principal del proyecto (bien escogiéndola directamente o mediante el botón *Browse*).

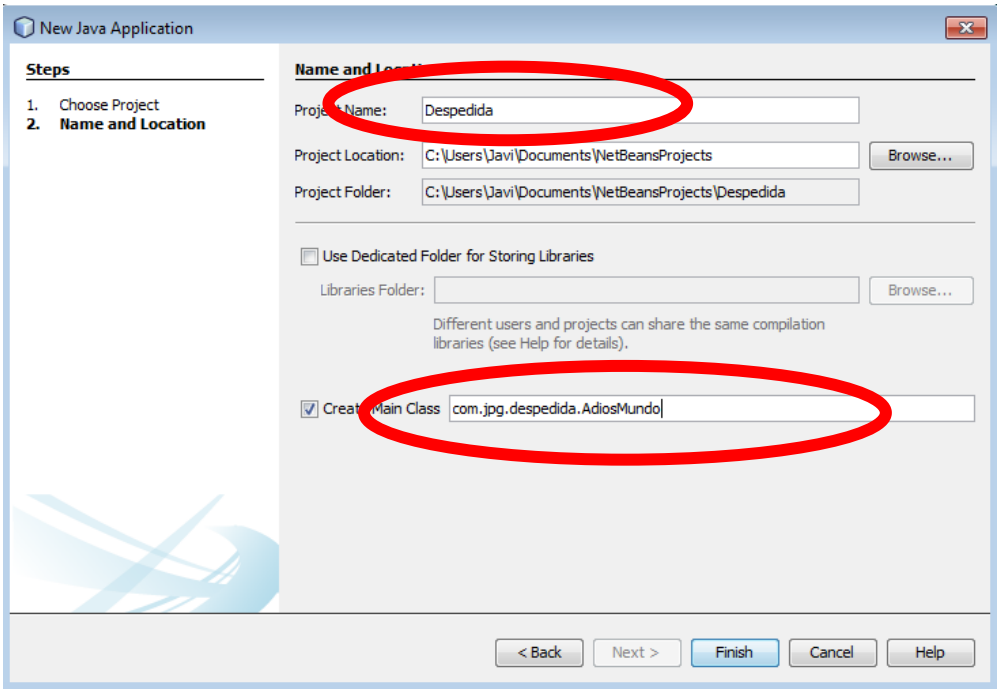


La clase principal del proyecto será la que se ejecute al hacer *Run*:



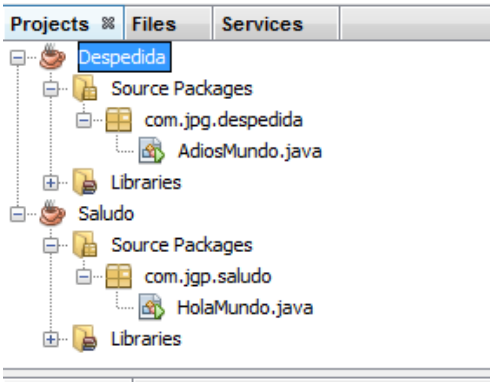
2. *Hacer un proyecto distinto para cada ejemplo que vayamos creando.*

   En nuestro ejemplo, crearíamos un nuevo proyecto llamado *Despedida* siguiendo los mismos pasos que seguimos al crear el proyecto *Saludo*. La clase principal sería `AdiosMundo`.



   Editaríamos el archivo y al ejecutar veríamos el mismo resultado que en el caso anterior.

   La parte negativa de este segundo enfoque es que podemos acabar con un número excesivo de proyectos:



En principio es más conveniente usar el enfoque anterior, organizando nuestras clases convenientemente por medio de paquetes.

Un enfoque intermedio es realizar un proyecto para cada grupo de ejercicios (por ejemplo, para cada capítulo) y dentro del mismo ir organizando las clases en paquetes convenientemente. Se deja este tema a criterio del alumno.