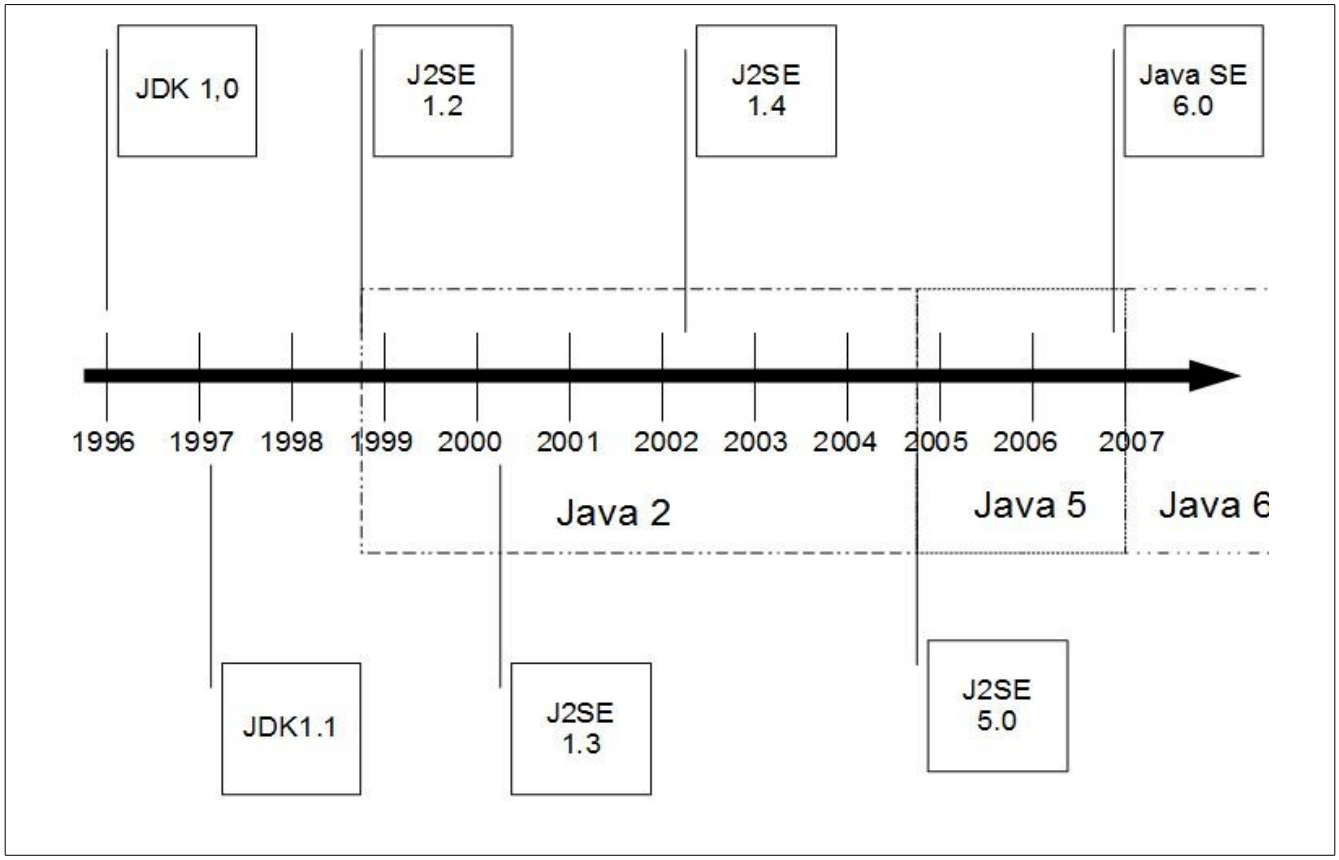
UNIDAD I.- Elementos de Interfaces Gráficas

LECCIÓN 1.1.- Creación de interfaz gráfica para usuarios

1.1.1.- La plataforma de Java

Las ediciones de la plataforma de Java

- **Standard Edition:** Permite desarrollar aplicaciones de escritorio
- **Enterprise Edition:** Permite desarrollar aplicaciones web
- **Micro Edition:** Permite desarrollar aplicaciones para dispositivos móviles.



Evolución de la Edición Estándar de Java

JDK 1.0 (23 de enero de 1996)

JDK 1.1 (19 de febrero de 1997).

Una reestructuración intensiva de:

- El modelo de eventos AWT (Abstract Windowing Toolkit)
- clases internas (inner classes)
- JavaBeans
- JDBC (Java Database Connectivity), para la integración de bases de datos.
- RMI (Remote Method Invocation).

J2SE 1.2 (8 de diciembre de 1998)

Ésta y las siguientes versiones fueron recogidas bajo la denominación Java 2.

El nombre "J2SE" (Java 2 Platform, Standard Edition), reemplazó a JDK para distinguir la plataforma base de J2EE (Java 2 Platform, Enterprise Edition) y J2ME (Java 2 Platform, Micro Edition).

Se adicionó:

- La palabra reservada (keyword) strictfp.
- Reflexión en la programación.
- La API gráfica (Swing) fue integrada en las clases básicas.
- La máquina virtual (JVM) de Sun fue equipada con un compilador JIT (Just in Time) por primera vez.
- Java Plug-in.
- Java IDL, una implementación de IDL (Interfaz para Descripción de Lenguaje) para la interoperabilidad con CORBA.
- Colecciones (Collections)

J2SE 1.3 (8 de mayo de 2000)

- La inclusión de la máquina virtual de HotSpot JVM (la JVM de HotSpot fue lanzada inicialmente en abril de 1999, para la JVM de J2SE 1.2)
- RMI fue cambiado para que se basara en CORBA.
- JavaSound
- Se incluyó el Java Naming and Directory Interface (JNDI) en el paquete de librerías principales (anteriormente disponible como una extensión).
- Java Platform Debugger Architecture (JPDA)

J2SE 1.4 (6 de febrero de 2002)

Este fue el primer lanzamiento de la plataforma Java desarrollado bajo el Proceso de la Comunidad Java como JSR 59. Los cambios más notables fueron:

- Palabra reservada assert.
- Expresiones regulares modeladas al estilo de las expresiones regulares Perl.
- Encadenación de excepciones Permite a una excepción encapsular la excepción de bajo nivel original.
- Non-blocking NIO (New Input/Output).
- Logging API.
- API I/O para la lectura y escritura de imágenes en formatos como JPEG o PNG.
- Parser XML integrado y procesador XSLT (JAXP).
- Seguridad integrada y extensiones criptográficas (JCE, JSSE, JAAS) Java Web Start incluido (El primer lanzamiento ocurrió en Marzo de 2001 para J2SE 1.3)

J2SE 5.0 (30 de septiembre de 2004)

(Originalmente numerado 1.5, esta notación aún es usada internamente).

Desarrollado bajo JSR 176, Añadió un número significativo de nuevas características:

- Plantillas (genéricos): Provee conversion de tipos (type safety) en tiempo de compilación para colecciones y elimina la necesidad de la mayoría de conversion de tipos (type casting).

- Metadatos: También llamados anotaciones, permite a estructuras del lenguaje como las clases o los métodos, ser etiquetados con datos adicionales, que puedan ser procesados posteriormente por utilidades de proceso de metadatos.
- Autoboxing/unboxing: Conversiones automáticas entre tipos primitivos (Como los int) y clases de envoltura primitivas (Como Integer).
- Enumeraciones: La palabra reservada enum crea una typesafe, lista ordenada de valores (como Dia.LUNES, Dia.MARTES, etc.). Anteriormente, esto solo podía ser llevado a cabo por constantes enteras o clases construidas manualmente (enum pattern).
- Varargs (número de argumentos variable) - El último parámetro de un método puede ser declarado con el nombre del tipo seguido por tres puntos. En la llamada al método, puede usarse cualquier número de parámetros de ese tipo, que serán almacenados en un array para pasarlos al metodo.
- Bucle for mejorado - La sintaxis para el bucle for se ha extendido con una sintaxis especial para iterar sobre cada miembro de un array o sobre cualquier clase que implemente Iterable, como la clase estándar Collection.

Java SE 6 (11 de diciembre de 2006)
En esta versión, Sun cambió el nombre "J2SE" por Java SE y eliminó el ".0" del número de versión. Los cambios más importantes introducidos en esta versión son:

- Incluye un nuevo marco de trabajo y APIs que hacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript.
- Incluye el motor Rhino, de Mozilla, una implementación de Javascript en Java.
- Incluye un cliente completo de Servicios Web y soporta las últimas especificaciones para Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP.
- Mejoras en la interfaz gráfica y en el rendimiento.

Java SE 7 - En el año 2006 aún se encontraba en las primeras etapas de planificación. Se espera que su desarrollo dé comienzo en la primavera de 2006, y se estima su lanzamiento

para 2008.

- Soporte para XML dentro del propio lenguaje.
- Un nuevo concepto de superpaquete
- Soporte para closures
- Introducción de anotaciones estándar para detectar fallos en el software.

Además de los cambios en el lenguaje, con el paso de los años se han efectuado muchos más cambios dramáticos en la librería de clases de Java *(Java class library)* que ha crecido de unos pocos cientos de clases en JDK 1.0 hasta más de tres mil en J2SE 5.0. APIs completamente nuevas, como Swing y Java2D, han sido introducidas y muchos de los métodos y clases originales de JDK 1.0 están desaprobados.

En base a la documentación oficial de Java, Sun Microsystems provee dos principales productos de software en la edición estándar de Java 2 (Java SE), como se muestra en la tabla 1.1.

Tabla 2.1.- Productos de la edición estándar de Java 2.

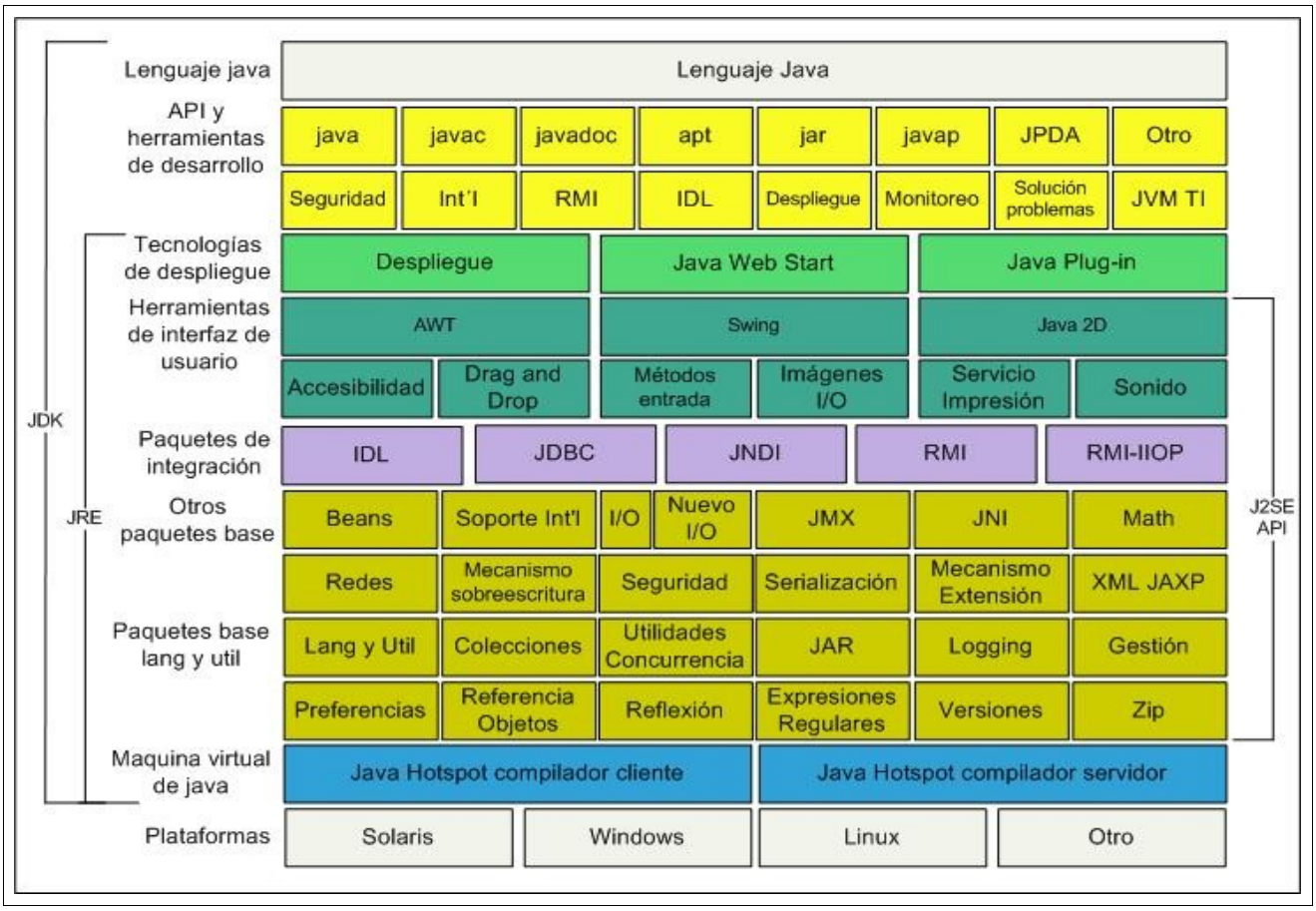
Producto	**Descripción**
JRE	El entorno de ejecución provee las librerías, máquina virtual y otros componentes necesarios para poder ejecutar applets y aplicaciones escritas en Java. Este entorno puede ser redistribuido con las aplicaciones para hacerlas autónomas.
JDK	El kit de desarrollo incluye el JRE más la consola de línea de comandos, las herramientas de desarrollo tales como compiladores y depuradores que son necesarios o útiles para el desarrollo de applets y aplicaciones.

La edición estandar se instala en una carpeta que tiene el nombre de la versión liberada:
(por ejemplo C:\Program Files\Java\jdk1.6.0_10).
Las carpetas mas importantes son:

- bin: Donde se alojan las herramientas de desarrollo
- jre: Donde se aloja la API de Java

La JRE tiene dos carpetas (bin y lib).

Dentro de lib existe un archivo (rt.jar) que contiene los paquetes que conforman la API de Java.



Los paquetes de la API de Java

Un paquete es un conjunto de clases relacionadas organizadas en carpetas.

El paqueta java.util significa que las clases estan en la carpeta /java/util/

Tarea 1.- Describa el contenido de los siguientes paquetes:
- java.lang
- java.io
- java.util

- java.net
- java.bean

Tarea 2.- Cite almenos tres clases de cada uno de los siguientes paquetes
- java.lang
- java.io
- java.util
- java.net
- javax.swing

Proceso de creación y ejecución de un programa en Java

1.1.2.- Pasos para crear un código ejecutable

Los pasos para generar un código ejecutable en Java se pueden definir como la codificación, la compilación y la ejecución.

Codificación

Para crear un ejemplo sencillo de un código ejecutable, se debe usar un editor de texto (por ejemplo el block de notas), donde se escribe el código funte (un ejemplo es mostrado en la figura 1.1). Hay que tener en cuenta que Java hace distinción entre mayúsculas y mayúsculas.

```
public class Saludo
{
      public static void main (String [ ]  args )
      {
            System.out.println ("Bienvenido a Java");
      }
}
```

Figura. 1.1.- Programa para mostrar un texto de saludo.

Se procede a guardar este programa en un archivo de texto llamado Saludo.java (el

nombre del archivo debe ser el mismo nombre que se le ha dado a la clase). Si se está utilizando el block de notas, antes de guardar el archivo se debe elegir "*Todos los archivos*" en la opción "*Tipo*", dentro del cuadro de diálogo "*Guardar*", especificando en la opción "*Nombre*" el nombre del archivo y su extensión (figura 1.2).

Compilación

La compilación de un archivo de código fuente .java se realiza a través del comando *javac.exe* del JDK. Si se ha establecido correctamente la variable de entorno PATH, *javac* podrá ser invocado desde el directorio en el que se encuentre el archivo .java (figura 1.3) Tras ejecutar el comando, se generarán tantos archivos .class como clases existan en el código fuente, en este ejemplo se creará solamente el archivo Saludo.class.

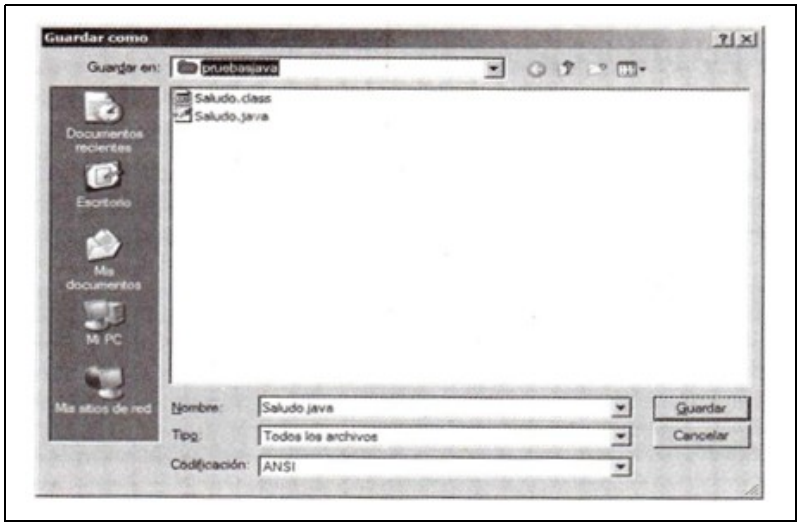


Figura. 1.2.- Guardar archivos de código Java con el block de notas.

En caso de que existan errores sintácticos en el código fuente, el compilador informa al usuario y por supuesto, el bytecode no se generaría. Por ejemplo, si en el código anterior se cambia *System* por *system,* al intentar la compilación se obtiene un mensaje de error como el indicado en la figura 1.4.

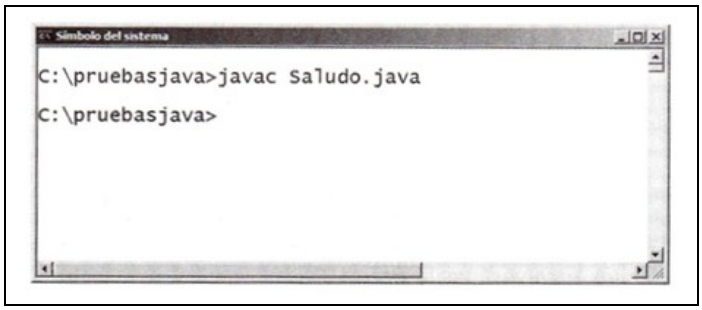


Figura. 1.3.- Compilación de un archivo de código fuente Java

Ejecución

Para ejecutar el programa, se utiliza el comando *java.exe*, seguido del nombre de la clase que contiene el método *main();* en este caso será Saludo, que es la única que existe. Es necesario que la variable de entorno CLASSPATH esté correctamente configurada e incluya el caracter ”.” (directorio actual) en la lista de direcciones, lo que permitirá invocar al comando java desde el directorio en el que se encuentra el .class.

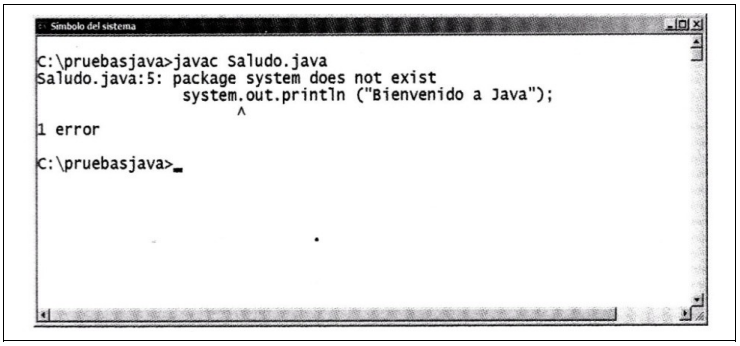


Figura. 1.4.- Error de compilación de una clase.

La llamada a java.exe insta a la maquina virtual a buscar en la clase indicada por el método main() y procede a su ejecución. En caso de que java.exe no encuentre la clase, bien porque la dirección del directorio actual (.) no figure en el CLASSPATH o bien porque el nombre de la clase no sea el correcto, se producirá una excepción (error) de tipo NoClassDelFoundError al intentar ejecutar el comando java.exe.

Si el problema no es la dirección de la clase, sino que el formato del método main() no es correcto, el programa compilará correctamente pero se producirá una excepción de tipo NoSuchMethodError (figura 1.5) al ejecutar el comando.

El procedimiento que se acaba de explicar para compilar y ejecutar la clase Saludo es el mismo que habrá que aplicar para las distintas clases que se creen en Java.

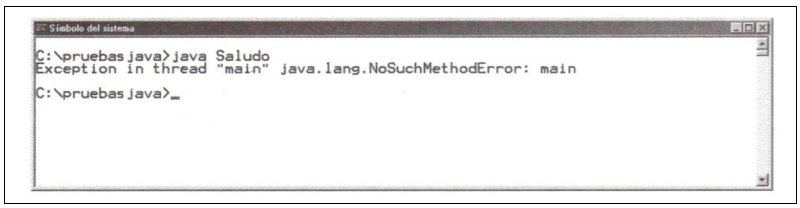


Figura 1.5.- Si el formato del método main() no es correcto la JVM no lo encuentra.

1.1.3.- Tipos de clases en Java

Se tiene que existen dos tipos de clases en Java:

- Clases instanciables: Son clases que modelan un grupo de objetos, que tienen atributos y método, pero carecen del método main().
- Clases aplicación: Son clases que sirven para crear los objetos y enviar los mensajes entre ellos. Contienen el método main().

1.1.4.- Elementos del entorno de NetBeans

La figura 3.6 presenta las ventanas principales que se presentan la pantalla principal de NetBeans. La tabla 3.5 muestra una descripción de estas ventanas.

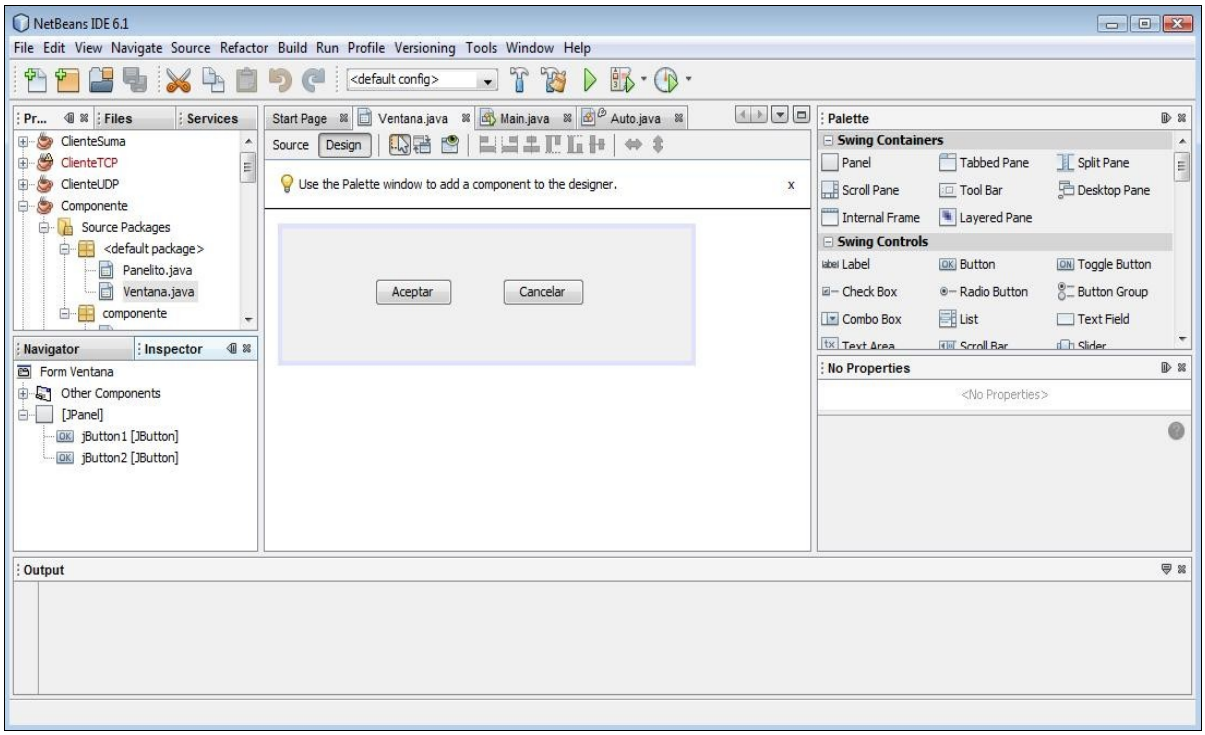


Figura 3.6.- Ventanas del entorno principal de NetBeans.

Tabla 3.5.- Ventanas del entorno principal de NetBeans

Ventana	Descripción	Acceso rápido
Proyectos	La ventana de proyectos presenta todos los proyectos abiertos. Los proyectos son el punto de entrada principal de Netbeans que sirve para categorizar y agrupar archivos utilizados en una aplicación. Un proyecto puede ser una Aplicación Java, una Aplicación Web, una librerías de clases, o cualquier otra cosa.	Ctrl+1
Archivos	La ventana de archivos proporciona una vista basada en archivos de los proyectos abiertos. Contiene la misma información de la ventana de Proyectos, pero organizada como un administrador de archivos. Los archivos que forman un proyecto son organizados en carpetas.	Ctrl+2

Tabla 3.5.- Ventanas del entorno principal de NetBeans

Ventana	Descripción	Acceso rápido
Servicios	En esta ventana se visualizan servicios importantes como servidores HTTP, servidores de base de datos, así como una lista de procesos ejecutándose en la computadora.	Ctrl+5
Navegador	La ventana del navegador provee una vista jerárquica de los elementos que conforman una archivo. Estos elementos pueden ser métodos, constructores, campos en una clase.	Ctrl+7
Fuente	En esta ventana es donde se edita el código. Si se trata de elementos gráficos, se presenta un asistente de diseño (Ventanas, Móviles, UML).	Ctrl+0
Salida	La ventana de salida puede desplegar una variedad de información. Si se selecciona la construcción de un proyecto, la compilación de un solo archivo o la ejecución de un archivo, la información o los resultados del proceso se presentan en la ventana de salida	Ctrl+4
Propiedades	La ventana de propiedades despliega los atributos y las propiedades de cada uno de los elementos seleccionados en la ventana de proyectos o en el editor de código.	Ctrl+7
Paleta	La ventana de paleta presenta una lista de elementos que son útiles para el archivo que actualmente se esta editando. Por ejemplo, si se está trabajando en un código con la herramienta de edición gráfica, la paleta se llena de componentes gráficos que pueden ser utilizados en el proyecto.	Ctrl+Shift+8

Tipos de proyectos que soporta NetBeans

NetBeans maneja una gran variedad de proyectos que agrupa en nueve categorías: Java, Web, Empresariales, de Movilidad, UML, SOA, Ruby, C/C++ y modulos NetBeans. Cada categoría tiene una serie de tipos de proyectos que se pueden seleccionar. La tabla 3.6 describe los tipos de proyectos mas representativos de NetBeans.

Tabla 3.6.- Tipos de proyectos en NetBeans

Ventana	Descripción
Aplicación Java	Crea una nueva aplicación Java en un proyecto estándar. Este proyecto contiene una clase Aplicación con el método main().
Aplicación de escritorio Java	Crea un esqueleto de una aplicación de escritorio basada en el Swing Application Framework (JSR 296). Esta plantilla proporciona elementos gráficos básicos como barras de menús. Con esta plantilla también se

Tabla 3.6.- Tipos de proyectos en NetBeans

	puede generar código para crear una interfaz gráfica para manejo de una tabla de una base de datos.
Librería de clases Java	Crea una nueva biblioteca de clases Java. Una biblioteca de clases no tiene una clase aplicación con el método main().
Aplicación web	Crea un proyecto para administrar código de programas para aplicaciones basadas en Web.
Aplicación empresarial	Crea un proyecto para desarrollar aplicaciones web con orientación empresarial, utilizando los componentes empresariales de Java.
Modulo EJB	Crea una nueva empresa JavaBean (EJB) en un módulo estándar IDE.

Creación de una aplicación Java con NetBeans

NetBeans presenta algunas opciones para crear una aplicación Java. La figura 3.7 presenta la ventana para la creación de una aplicación Java. Si se selecciona la creación de una clase Main, NetBeans asociará una clase con ese nombre que será la que se ejecutará cuando el proyecto corra. La figura 3.8 muestra la ventana inicial del proyecto. En este caso se observa en la ventana de Edición una plantilla para la clase Main, conteniendo el método main().

Como se observa en la figura 3.8, la ventana de Proyectos presenta un árbol donde se pueden observar los archivos asociados al proyecto. Un proyecto en NetBeans tiene siempre cuatro grupos de archivos. La tabla 3.7 presenta una descripción de estos grupos.

Tabla 3.7.- Grupos de archivos de un proyecto de NetBeans.

Grupo	**Descripción**
Paquetes origen	Aquí se define la ubicación del código fuente usado en la aplicación. Es en este punto donde el código capturado se puede organizar en paquetes.
Paquetes de prueba	En este lugar se coloca el código de prueba.
Librerías	Si se requieren utilizar librerias no estándares para el proyecto, se deben declarar en este grupo.
Librerías de prueba	De la misma forma que los paquetes de prueba, aquí se colocan las librerias de prueba.

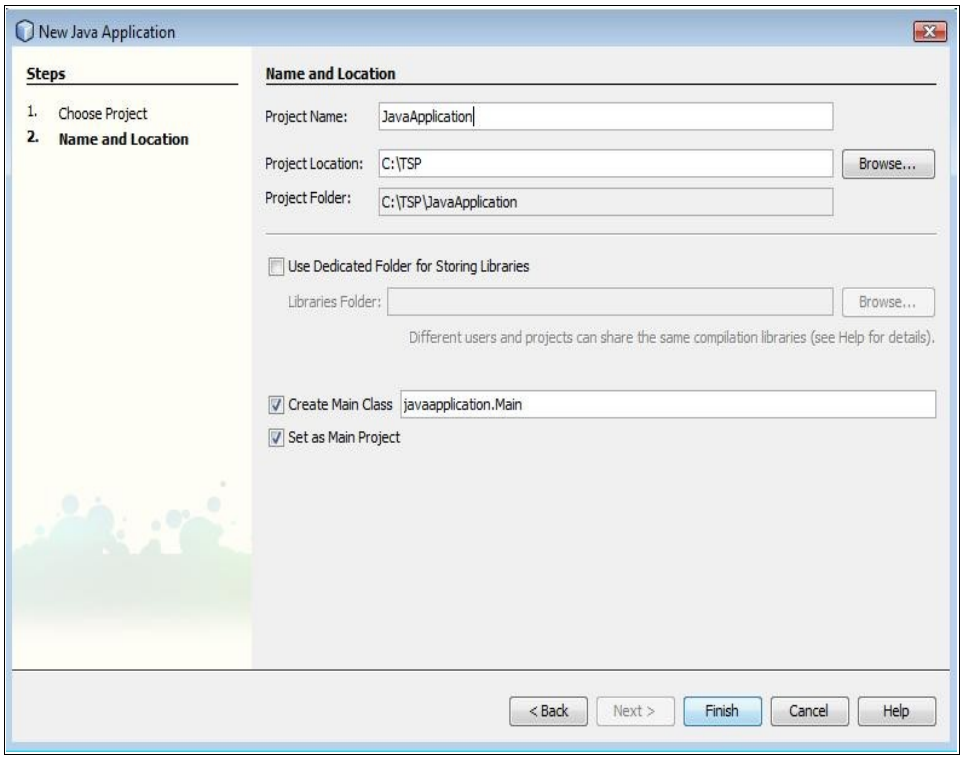


Figura 3.7.- Opciones de para la creación de una aplicación Java.

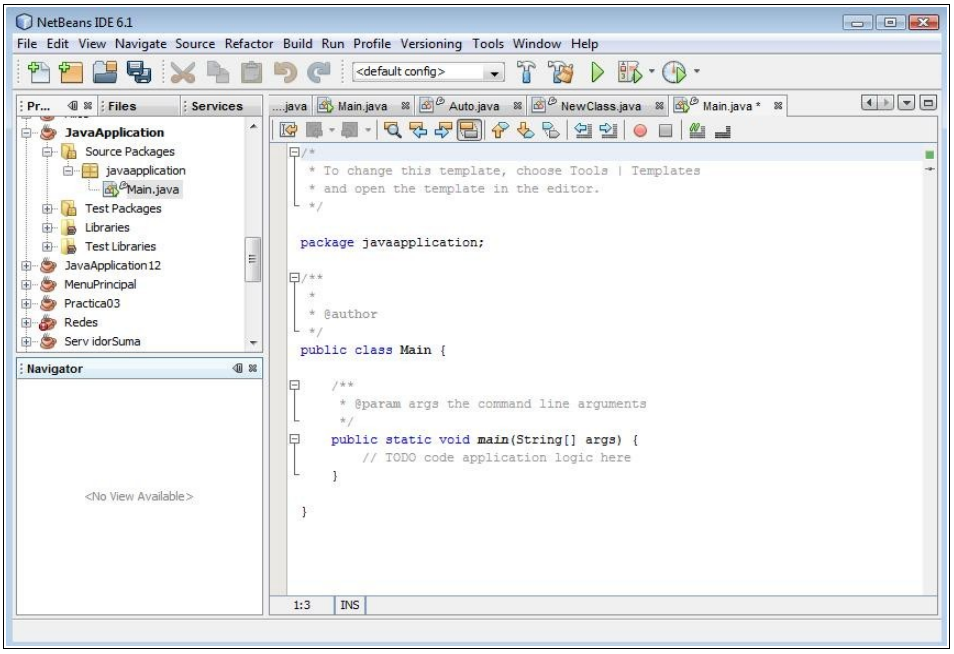


Figura 3.8.- Ventana inicial del proyecto “JavaApplication”.

La ventana de archivos (figura 3.9) presenta los mismos archivos que la ventana de proyectos, pero ordenadas en una forma más tradicional. NetBeans coloca el código fuente de una aplicación dentro de la carpeta “src”, los binaros resultado de la compilación (bytecodes) en la carpeta “classes” y el archivo comprimido de clases en una carpeta denominada “dist”.

La ventana de edición es rica en ayudas al programador, indicando los errores de sintaxis y proporcionando elementos para construir código rapidamente (autocompletado).

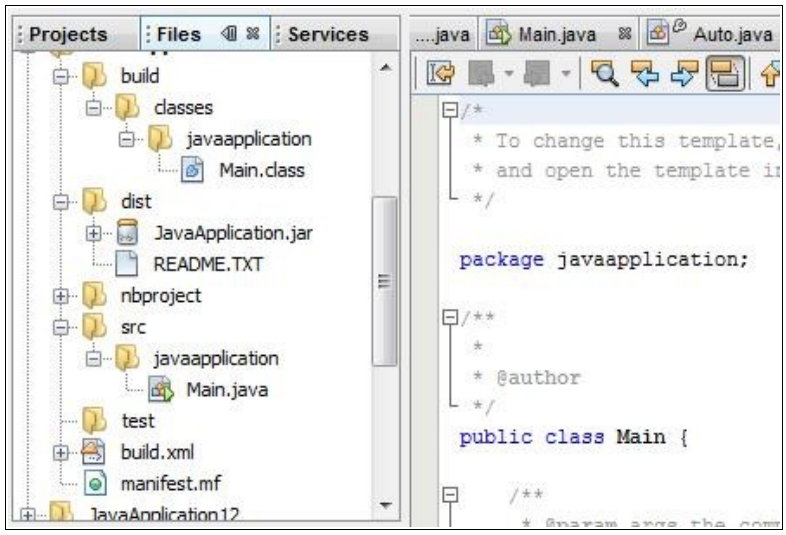


Figura 3.9.- Ventana de archivos del proyecto.

Creación de una clase de Java (instanciable)

Para que se pueda ejecutar un programa en Java, se necesita incluir código de clases instanciables al proyecto, a partir de las cuales se puedan crear objetos y enviarles mensajes desde la clase Main. NetBeans proporciona plantillas para una variedad de tipos de clases de Java. Estos tipos de archivos se agrupan en categorias. La figura 3.10 muestra la ventana de selección de tipo de archivo, la tabla 3.8 presenta una descripción de las categorias de archivos y la tabla 3.9 muestra una descripción de los tipos de archivos de la categoria “clase

de Java".

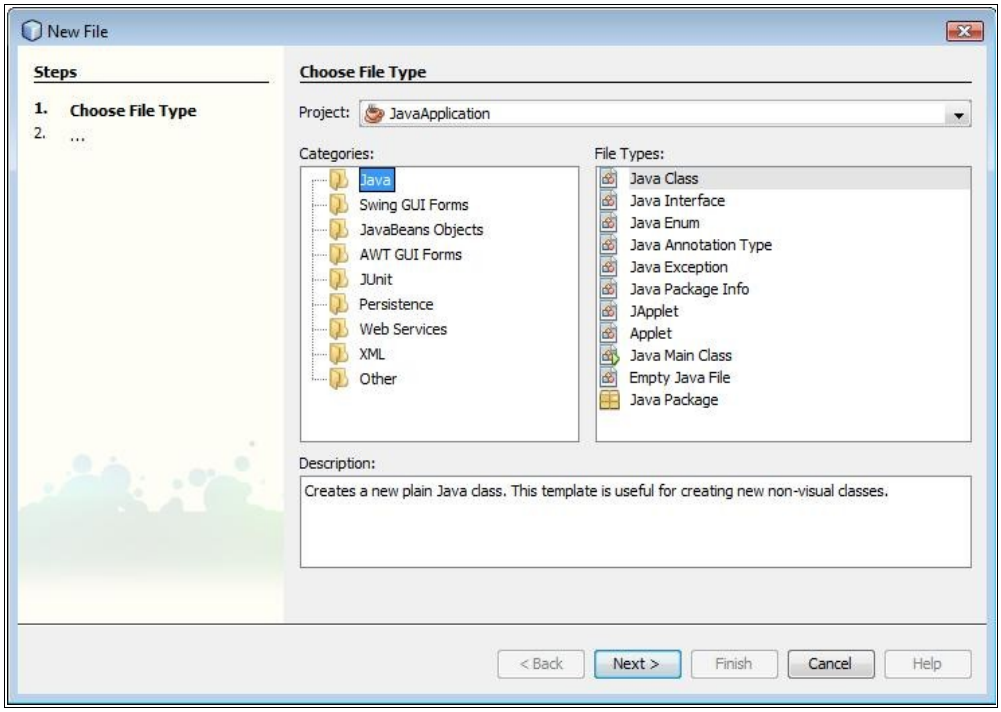


Figura 3.10.- Ventana de selección de tipo de achivo para un proyecto en NetBeans

Tabla 3.8.- Categorías de tipos de archivos para una Aplicación Java en NetBeans.

Categoría	Descripción
Java	Crea una nueva clase Java simple. Esta plantilla es útil para la creación de nuevas clases no visuales.
Swing GUI Forms	Crea una forma Swing. Una forma es un asistente para el diseño de interfaces gráficas utilizando paletas de componentes.
JavaBean Objects	Crea una plantilla para escribir componentes en Java (JavaBeans).
AWT GUI Forms	Crea una forma AWT, semejante a la forma Swing. La diferencia está en que se utilizan las clases del paquete AWT.
JUnit	Crea un archivo de tipo JUnit para casos de prueba.
Persistence	Crea una clase vacia para persistencia de Java.
Web Services	Crea un cliente de servicios web que es compatible con el estándar JSR-109.
XML	Crea un nuevo documento XML. En el asistente se puede especificar el formato a utilizar (DTD, XML Schema, etc.)

Tabla 3.8.- Categorías de tipos de archivos para una Aplicación Java en NetBeans.

Categoría	**Descripción**
Other	Crea un nuevo archivo HTML.

Tabla 3.9.- Tipos de archivos de la categoría de clases de Java

Grupo	**Descripción**
Java Class	Crea una nueva clase Java. Esta plantilla es útil para la creación de nuevas clases no visuales.
Java Interface	Crea una nueva interfaz de Java
Java Enum	Crea una clase de tipo enumerado.
Java Annotation Type	Crea una nueva anotación de tipo Java.
Java Exception	Crea una nueva subclase de excepción con un mensaje detallado opcional.
Java Package Info	Crea un nuevo paquete Java-info.
JApplet	Crea un nuevo applet de Swing. Un applet es una clase Java que puede ejecutarse en cualquier navegador con Java.
Applet	Crea un nuevo applet de AWT.
Java Main Class	Crea una nueva clase Java con un método principal que le permite ser ejecutado como una aplicación de consola.
Empty Java File	Crea un archivo vacío de Java. Es plantilla se utiliza para crear una clase partiendo de cero.
Java Package	Crea un paquete de archivos fuente Java. Este paquete físicamente tiene la forma de una carpeta vacía en su disco.

Al seleccionar la creación de un tipo de clase, se presenta una ventana (figura 3.11) para determinar el nombre y la ubicación del archivo. Es en esta parte donde se indica a que grupo y a que paquete pertenece la clase.

.

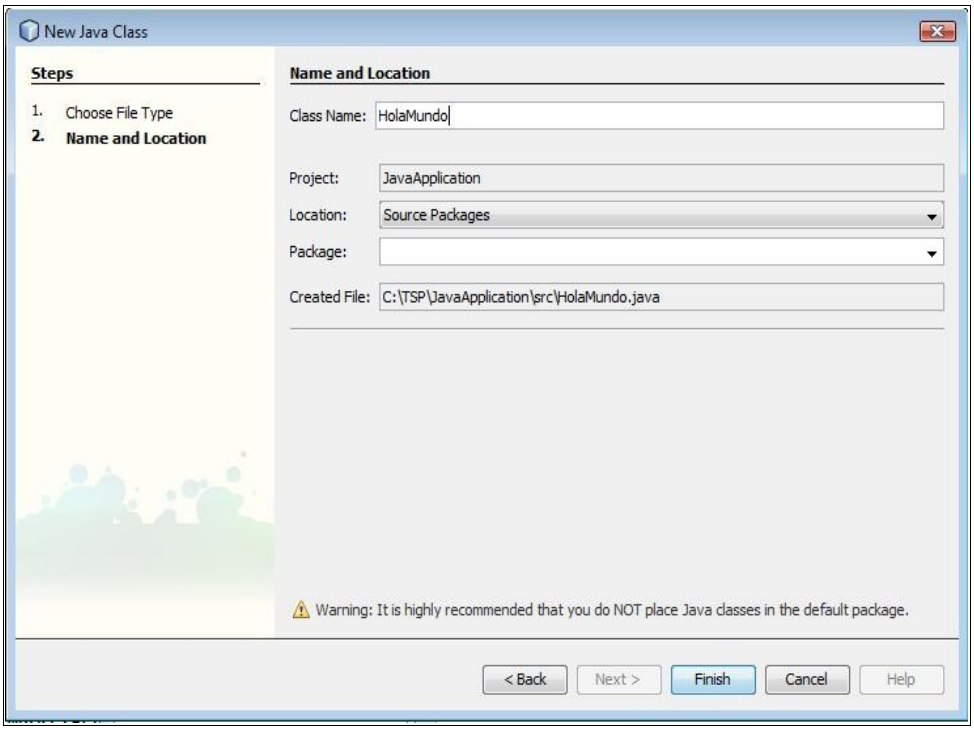


Figura 3.10.- Ventana para determinar el nombre y tipo de archivo.

NetBeans utiliza una plantilla para cada tipo de archivo. En la figura 3.11 se muestra la plantilla utilizada para una clase de Java. El usuario debe completar esta plantilla para producir un código ejecutable. Para el ejemplo desarrollado aquí, se incluye un método en la clase instanciable, y en la clase Main se crea un objeto y se envía un mensaje a este objeto. La figura 3.12 muestra las modificaciones realizadas.

La fase final del desarrollo de un proyecto de Java en NetBeans es la ejecución. Para realizar esto se requiere compilar los archivos que forman el proyecto en desarrollo. NetBeans permite compilar de manera independiente cada código o construir el proyecto completo, que significa que NetBeans verifique las dependencias entre los archivos y compile aquellos que han sido modificados. La compilación y la construcción se realiza a través del menú “Build” de la barra de menús proporcionada por NetBeans. Para correr el proyecto se utiliza el menú “Run” de la misma barra.

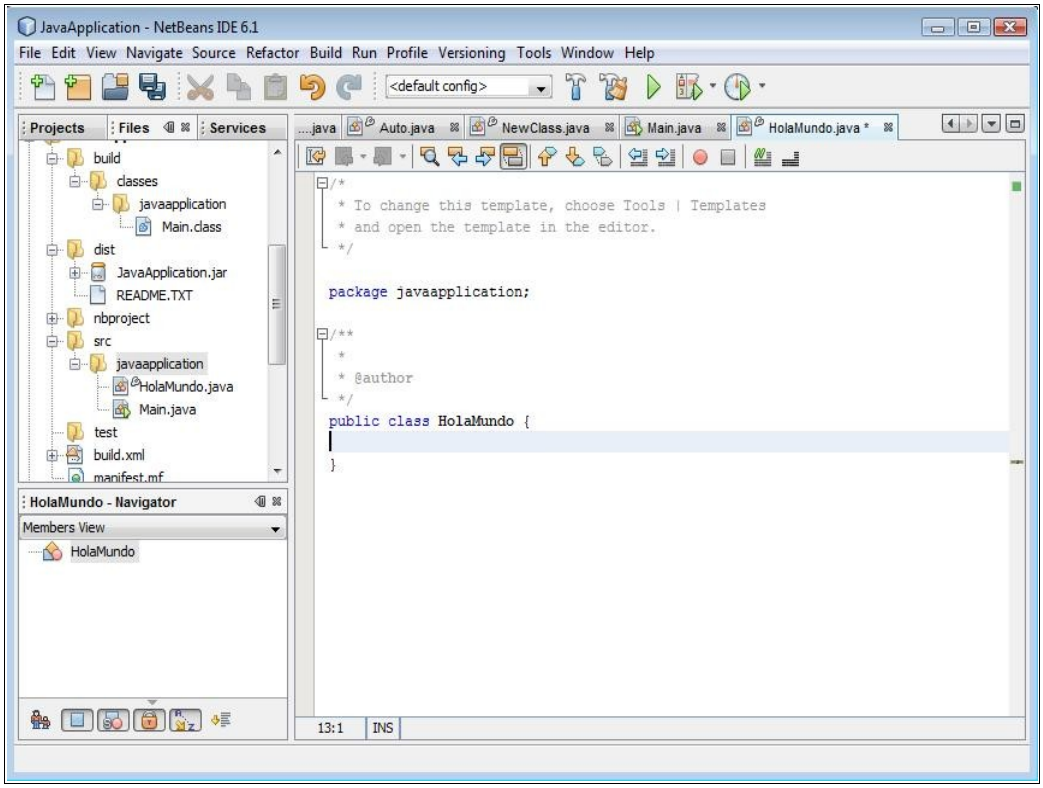


Figura 3.11.- Plantilla de una clase de Java.

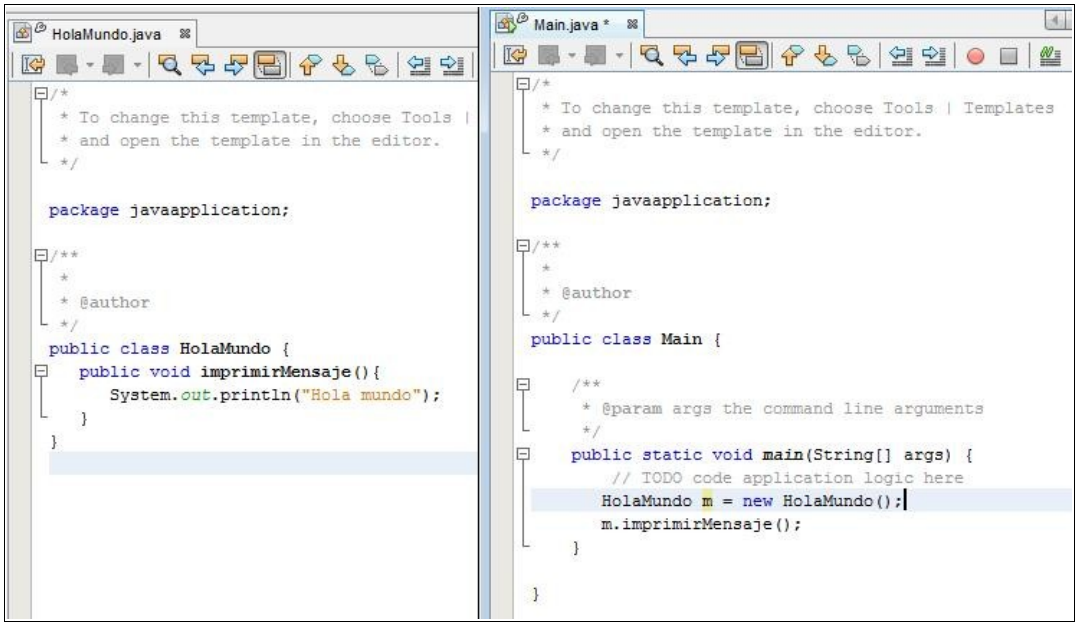


Figura 3.12.- Ejemplo de clase instaciable y clase aplicación en un proyecto de NetBeans.