#1

UNDERCODE

TALLER DE JAVA

TEMAS

- INTRODUCCIÓN
- ENTORNO
- IDES
- PRIMER PROGRAMA
- COMENTARIOS
- CONVENIOS
- Y MÁS!

TUTOR

EXPERMICID

Introducción

Java surgió en 1991 dentro de la empresa Sun Microsystems como un lenguaje de programación sencillo y universal destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje que pudiera generar código de tamaño muy reducido.

El lenguaje deriva mucho de su sintaxis de C y C++, pero tiene menos facilidades en bajo nivel que cualquiera de ellos. Las aplicaciones de Java se ejecutan sobre una "máquina virtual", denominada **Java Virtual Machine** (JVM). La **JVM** interpreta el código convirtiéndolo a código propio de la máquina concreta, independizándolo del procesador.

A continuación veremos las características generales de **Java**:

1) Es un lenguaje enteramente orientado a **objetos** y **clases**.
    - Una **clase** es una agrupación de variables (datos) y de los métodos (funciones) que manipulan esas variables.
    - Un **objeto** es un ejemplar concreto de una clase.
2) Maneja dos tipos de datos, tipos **primitivos** y **referencias**.
    - Los tipos **primitivos** de variables son: **boolean**, **char**, **byte**, **short**, **int**, **long**, **float** y **double**.
    - Las **referencias** son nombre de objetos de una clase determinada.
3) Características que se desprenden de ser un lenguaje orientado a objetos.
    - **Encapsulamiento**. Se puede regular el acceso a los miembros (variables y métodos) de una clase, declarándolos como **public**, **private** y **protected**.
    - **Herencia**. Una clase puede derivar de otra, heredando sus variables y métodos, y además añadir o redefinir algunas variables y métodos nuevos. En Java existe una jerarquía de clases en donde todo se deriva de la clase **Object**. Cabe destacar que no existe la herencia múltiple.
    - **Polimorfismo**. Se puede tratar a una colección de objetos de distintas clases, similares pero distintos, de una manera unificada.

Preparando el entorno

Lo que necesitaremos para programar en Java, será un editor de texto o IDE y la JDK.

La **JDK** (Java Development Kit) también llamada **Java SDK** (Software Development Kit) ofrece las herramientas para que los programadores compilen, depuren y ejecuten programas en Java. Para obtenerlo entramos en:

http://www.oracle.com/technetwork/es/java/javase/downloads/index.html

Descargamos el que corresponda a nuestro sistema operativo e instalamos.

Java SE Development Kit 7u17

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement **Decline License Agreement**

Product / File Description	File Size	Download
Linux x86	106.65 MB	jdk-7u17-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u17-linux-i586.tar.gz
Linux x64	104.78 MB	jdk-7u17-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u17-linux-x64.tar.gz
Mac OS X x64	143.78 MB	jdk-7u17-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.39 MB	jdk-7u17-solaris-i586.tar.Z
Solaris x86	91.67 MB	jdk-7u17-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.92 MB	jdk-7u17-solaris-sparc.tar.Z
Solaris SPARC	95.32 MB	jdk-7u17-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.97 MB	jdk-7u17-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.59 MB	jdk-7u17-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.61 MB	jdk-7u17-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u17-solaris-x64.tar.gz
Windows x86	88.75 MB	jdk-7u17-windows-i586.exe
Windows x64	90.42 MB	jdk-7u17-windows-x64.exe

Los **IDE's** (Integrated Development Enviroment o Entornos Integrados de Desarrollo) ofrecen un ambiente gráfico en los que se tiene acceso a un mayor número de herramientas no ofrecidas por la JDK.

Existe un montón de IDE's disponibles para trabajar con Java, pero entre los más conocidos están:

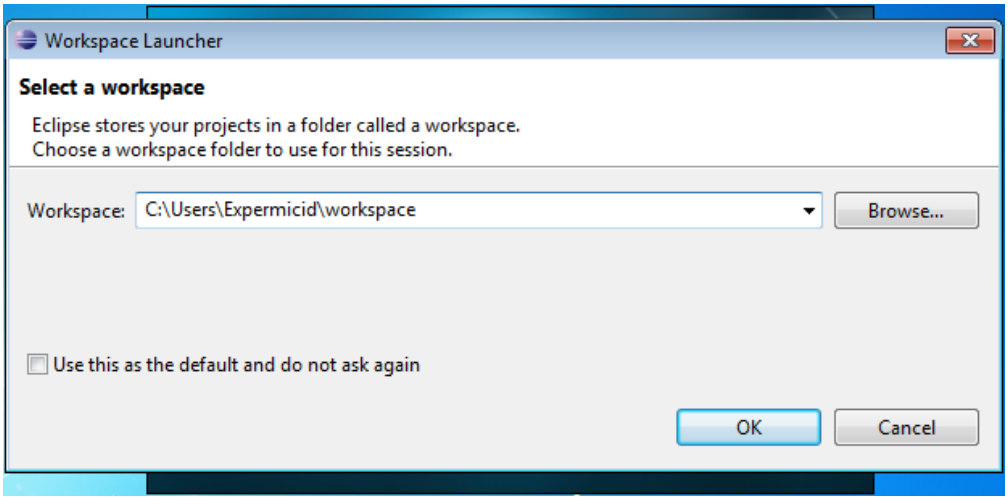
NetBeans IDE, disponible en https://netbeans.org/

Eclipse, disponible en http://www.eclipse.org/

Nosotros trabajaremos con el **Eclipse**.

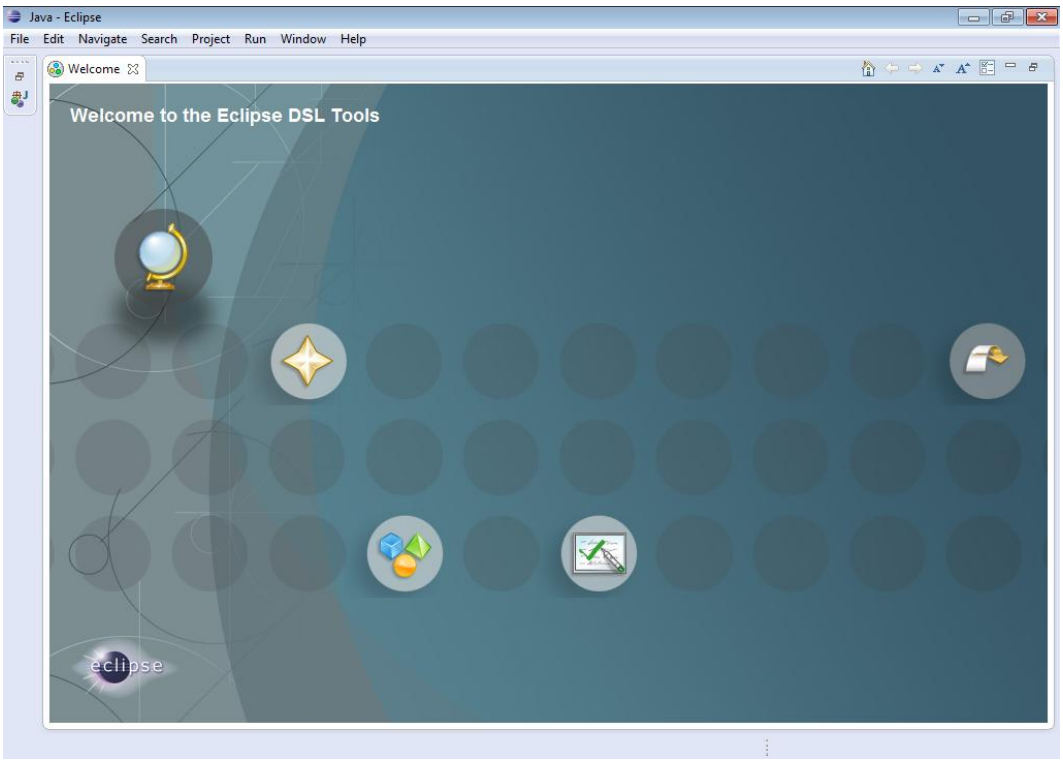
Conociendo el IDE

Cuando abramos el IDE nos aparecerá una ventana que nos da para elegir nuestro *lugar de trabajo* (WorkSpace). Y va a ser en donde se vayan guardando todos los proyectos que vayamos haciendo.



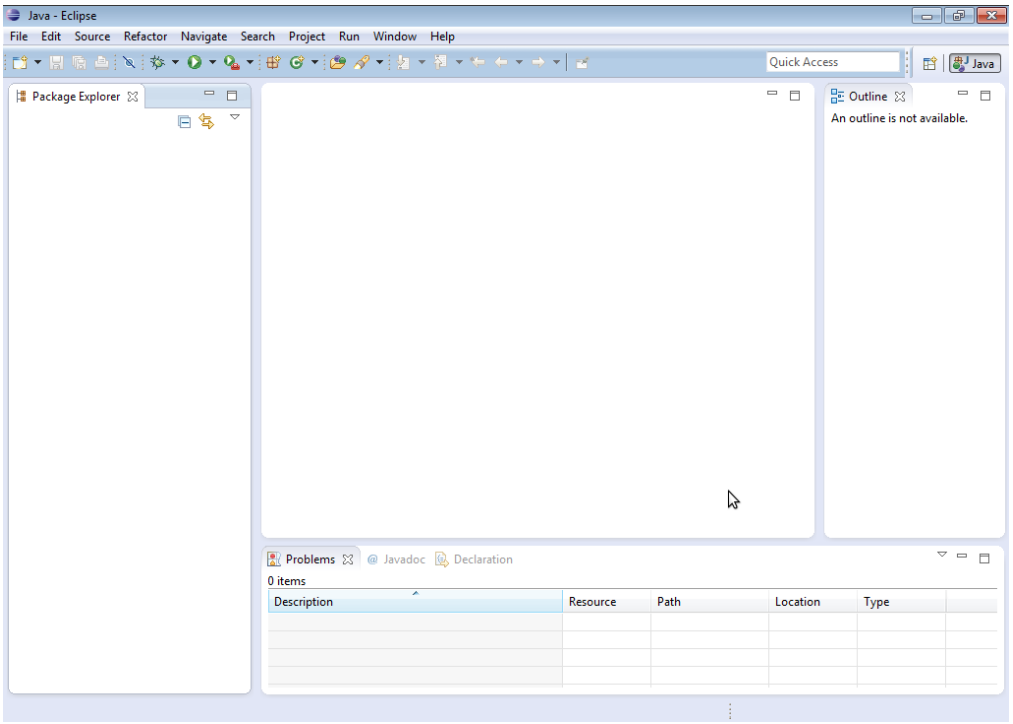
Eligen la ubicación que desean o dejan la que está por defecto. Si no quieren que le pregunte nuevamente clickean el CheckBox que dice “Use this as the default and do not ask again” y presionan OK.

Se abrirá el IDE con una bienvenida. Ustedes pueden leer las guías y ayudas que da.



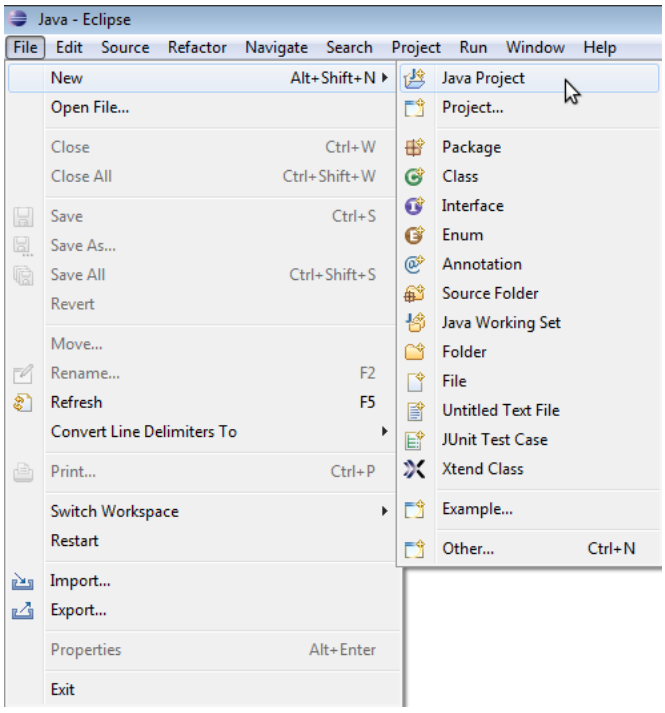
Cerramos la pestaña de bienvenida.

Nos aparecerá la ventana común del Eclipse. En donde podrán apreciar la barra de títulos, una barra de herramientas y el resto dividido es partes donde cada una tiene su funcionalidad.

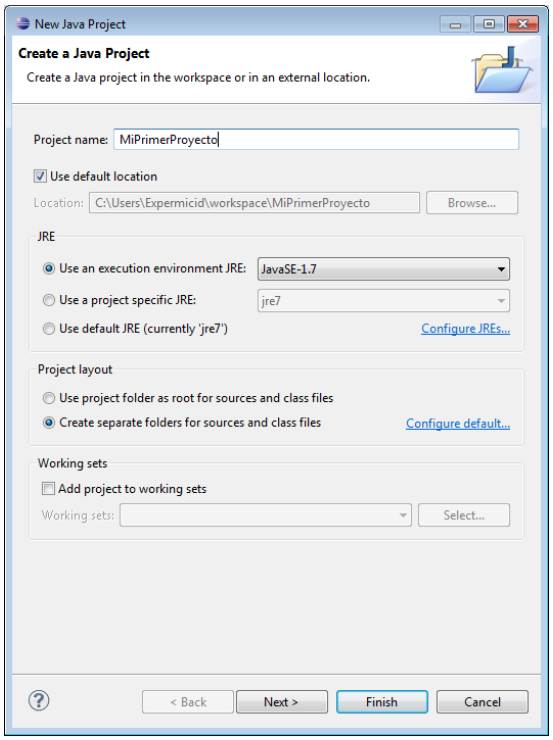


La parte izquierda es un explorador donde se podrán visualizar todos los proyectos con sus respectivos paquetes y clases. La parte central está destinada para ver y editar el código. La división de la derecha muestra las importaciones y todos los métodos de la clase seleccionada. Y por último la parte inferior es la que nos mostrará todos los errores que pudiera haber en el código y también posee la consola en donde podremos correr la aplicación java e interactuar con ella.

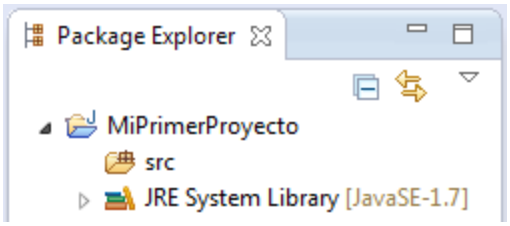
Para crear un Proyecto, debemos ir al menú **File > New > Java Project**.



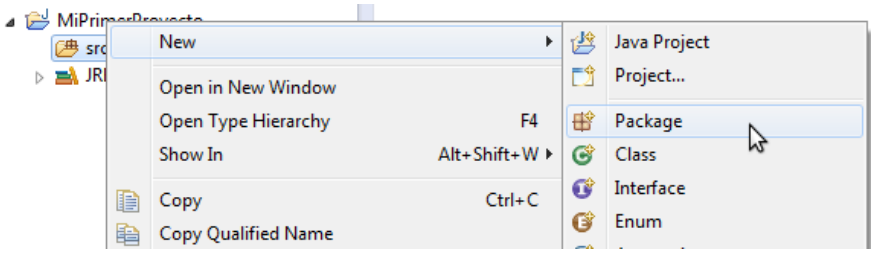
Aparecerá una ventana de configuración. En **Project Name** colocamos el nombre que deseemos y presionamos **Next > Finish**.



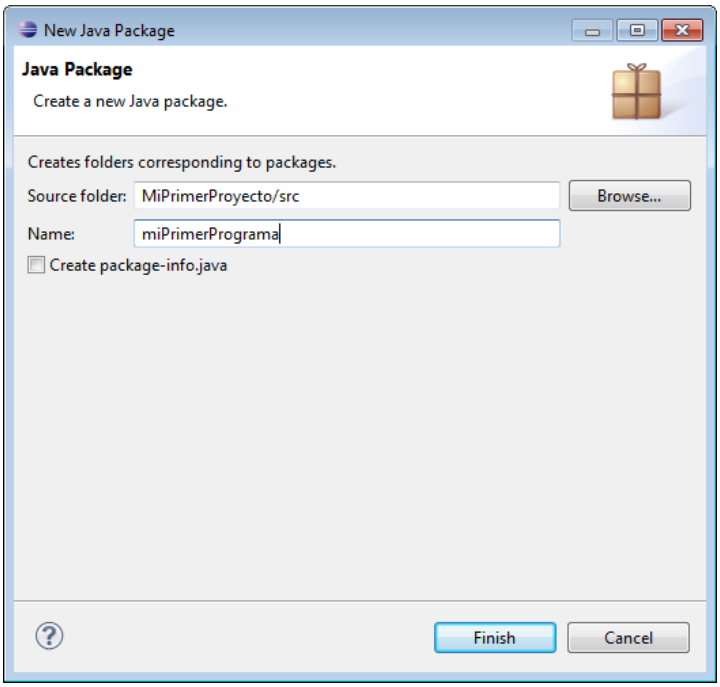
Podrán notar que en la división de la izquierda se agregara el nuevo proyecto, que por el momento está vacío.



Agregamos un nuevo **package** haciendo click derecho en src **New > Package**.

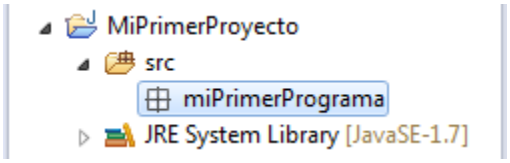


Aparecerá una ventana. En **Name** ponemos el nombre que deseemos para el **package**.

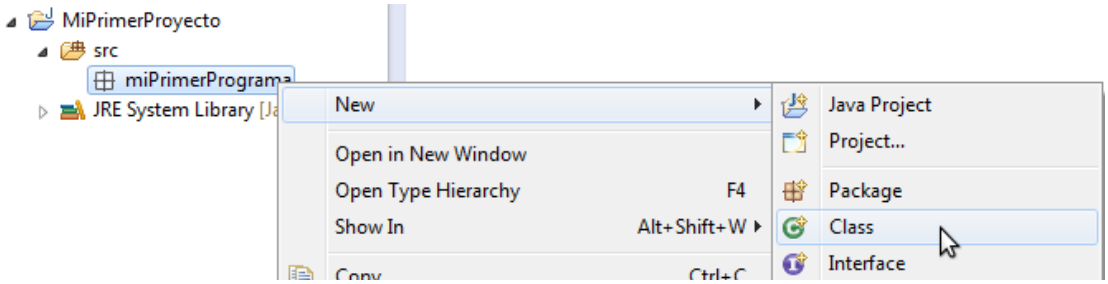


Y presionamos **Finish**.

Ya podremos ver el package creado.



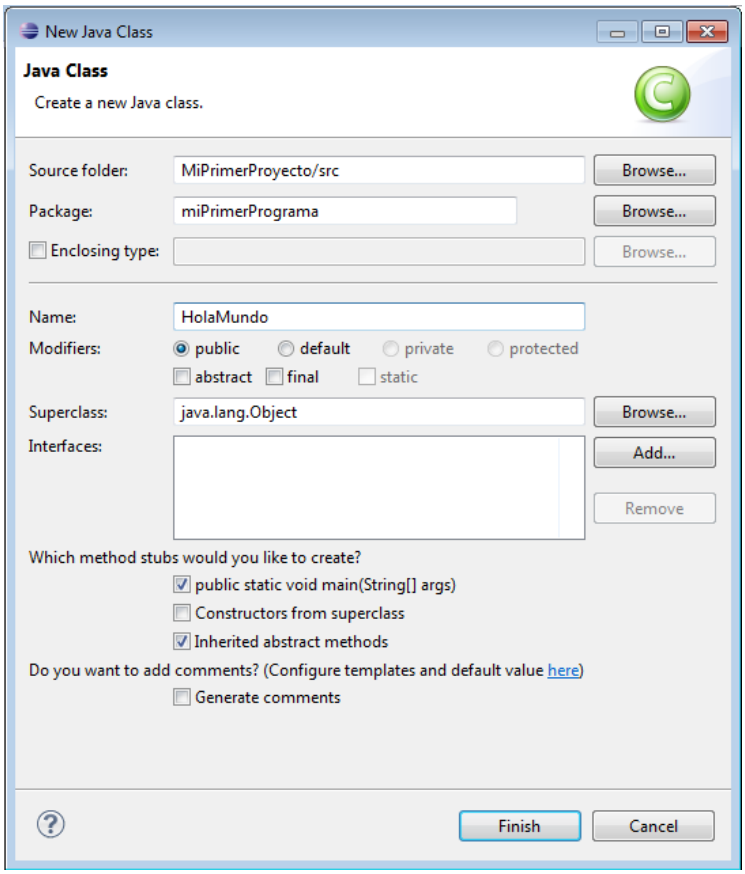
Le agregamos una clase haciendo click derecho en el package **New > Class**.



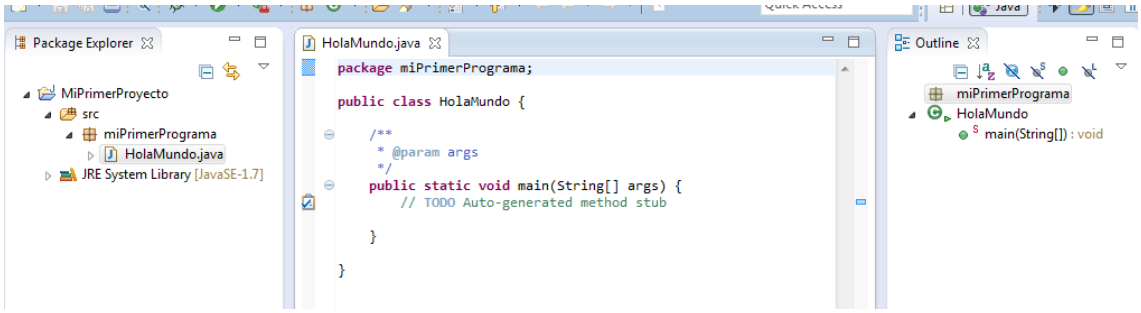
Aparecerá la ventana de configuración. En **Name** colocamos el nombre que deseemos.

Tildemos la casilla que dice **public static void main(String[] args)**si queremos que nos agregue automáticamente el método main. En caso contrario pueden agregarlo manualmente.

Por último presionamos **Finish**.



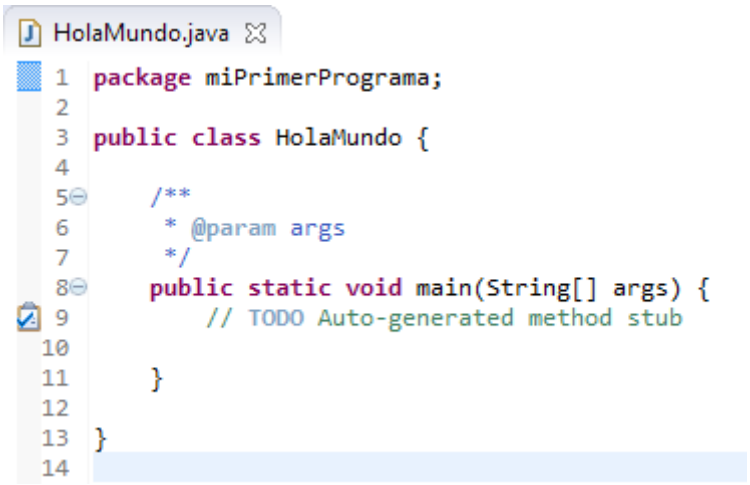
Notaran varios cambios realizados. En la parte izquierda se verá la nueva clase agregada, en la parte central en código de dicha clase y en la derecha la enumeración de métodos, etc.



Nuestra primera aplicación

Realizaremos el simple y clásico *HolaMundo*para comenzar a ver el código de Java.

Volviendo al proyecto creado anteriormente, miremos el código que eclipse nos ha creado por defecto en la **clase HolaMundo**.



La primera línea especifica el package. La sintaxis es la palabra ***package*** seguido del *nombre del package* y terminando con un *punto y coma* (**;**). El punto y coma se utiliza para finalizar la línea, si no se coloca dará un error de sintaxis.

En la línea 3 está la declaración de la ***clase HolaMundo*** que está como pública (palabra ***public***). A continuación está el corchete de apertura (**{**) que se cierra en la línea 13 (**}**), y son los encargados de marcar el comienzo y el fin de la clase. Entre ellos se colocará todo el código de la *clase HolaMundo.*

NOTA: El corchete de apertura no necesita estar en la misma línea de la clase. Puede colocarse en la línea siguiente (en este caso la línea 4).

De la línea 5 a la 7 hay un comentario de múltiple línea. Más adelante hablaremos mejor de los comentarios.

En la línea 8 comienza el método main. Toda aplicación debe tener una clase que contenga el método main, porque es el que se ejecuta primero. El método esta declarado como ***public*** y ***static*** (mas adelante lo detallaremos mejor), no devuelve nada (palabra ***void***) y toma como argumentos un arreglo de String.

La línea 9 contiene un comentario de una sola línea.

Y en la línea 11 se cierra el método main con el corchete de cierre (**}**).

Ahora vamos a agregarle una línea más que va a ser la encargada de imprimir la frase "*¡Hola Mundo!*".

En java disponemos de una ***clase System*** que es accesible de cualquier sitio. Esta clase contiene, entre otras cosas, un atributo ***out*** que es público. Para acceder a ese atributo debemos indicar que está dentro de System, eso se hace poniendo ***System.out***. Este atributo es otra clase que a su vez tiene métodos, algunos de ellos están destinados a imprimir una cadena de caracteres por pantalla. Los métodos más usados son ***print()***que imprime la serie

de caracteres que se le pasa como parámetro y ***println()***que además de imprimir agrega una nueva línea al final.

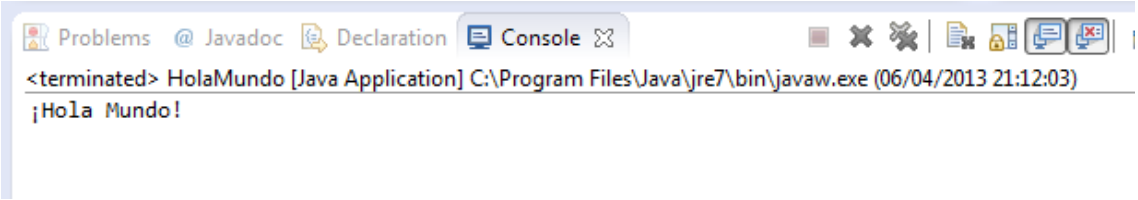
El código nos quedaría:

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.print("¡Hola Mundo!");
}
```

Para probarlo se puede hacer de varias formas, una de ella es ir a la barra de herramientas y presionar el botón . 

NOTA: Recuerden guardar todos los cambios hechos sino le aparecerá una ventana que pide hacerlo antes de ejecutar.

Si todo está bien, en la parte inferior, aparecerá la consola y en ella la frase *"¡Hola Mundo!"*



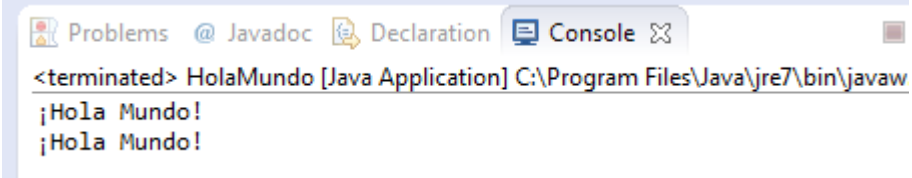
Voy a hacer unas modificaciones al código para que veamos como trabajan ***println()*** y ***print()***.

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.out.println("¡Hola Mundo!");

    System.out.print("¡Hola ");
    System.out.print("Mundo!");
}
```

Al principio uso un ***println()*** para imprimir la frase *"¡Hola Mundo!"*. Y seguido uso dos ***print()*** para imprimir la misma frase pero en dos partes.

Si lo probamos nos dará la siguiente salida:



Si analizamos el resultado veremos que después de la primer frase hay una nueva línea, eso es por el uso de ***println()*** y la segunda frase a pesar de haber usado dos ***print()*** esta toda completa como si hubiera usado uno solo.

Comentarios

Para poner comentarios en nuestro código fuente tenemos tres posibilidades.

Comentarios de una sola línea. Comienzan con doble barra (**//**) y a partir de ellas todo lo que se encuentra en la línea es considerado como comentario.

Puede ir en una línea nueva o después de código.

```
1  package miPrimerPrograma;
2
3  public class HolaMundo {
4
5      // Metodo main
6      public static void main(String[] args) {
7          System.out.println("¡Hola Mundo!"); // Imprimir por pantalla
8      }
9
10 }
11
```

Comentarios de más de una línea. Comienzan por **/***y hasta que no encuentre un ***/**todo lo que haya en el medio se considera como comentario.

```
1  package miPrimerPrograma;
2
3  public class HolaMundo {
4
5      /* Metodo main
6         Imprime en pantalla ¡Hola Mundo!
7       */
8      public static void main(String[] args) {
9          System.out.println("¡Hola Mundo!");
10     }
11
12 }
13
```

Comentarios para Javadoc. Son comentarios de múltiple línea que comienzan por **/**** y terminan con ***/**. Pero se diferencian en que estos salen en la documentación generada por Javadoc.

```
1  package miPrimerPrograma;
2
3  public class HolaMundo {
4
5      /**
6         @param args
7       */
8      public static void main(String[] args) {
9          System.out.println("¡Hola Mundo!");
10     }
11
12 }
13
```

Es recomendable usar comentarios en lugares claves (ejemplos: qué realiza cada método, para qué sirve cada variable, etc.) para que cualquiera que lea el código le sea más fácil entenderlo.

Convenios

En java existen convenios de nomenclatura comunes. Emplear estos convenios trae enormes beneficios ya que facilita la comprensión de códigos ajenos.

Alguno de ellos son:

- Los nombres de las clases e interfaces deben empezar siempre por mayúscula (Nombre). Si el nombre surge de componer varias palabras la letra inicial de cada una de esas palabras irá en mayúscula (NombreLargo).
- Los nombres de las variables y los métodos deben empezar por minúscula (variable, método()). Si los nombres surgen de componer varias palabras la primera palabra empezará por minúsculas y todas las demás por mayúscula. (variableNombreLargo, metodoNombreLargo()).
- Los nombres de los paquetes deben ir siempre en minúscula (paquete).
- Las constantes deben ir totalmente en mayúsculas. (CONSTANTE).

Palabras reservadas

Existen 48 palabras que forman parte de java y por lo tanto no pueden ser utilizadas para los nombres de variables, clases o métodos.

abstract	*default*	*if*	*private*	*this*
boolean	*do*	*implements*	*protected*	*throw*
break	*double*	*import*	*public*	*throws*
byte	*else*	*instanceof*	*return*	*transient*
case	*extends*	*int*	*short*	*try*
catch	*final*	*interface*	*static*	*void*
char	*finally*	*long*	*strictfp*	*volatile*
class	*float*	*native*	*super*	*while*
const	*for*	*new*	*switch*	

continue goto package synchronized

Además las palabras *null*, *true* y *false* que sirven como constantes no se pueden usar.