

¡HOLA MUNDO!

## COBOL

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. HELLO-WORLD.  
PROCEDURE DIVISION.  
    DISPLAY 'Hello, world!'.  
    STOP RUN.
```

## FORTRAN

```
program hello  
  write(*,*) 'Hello, world!'  
end program hello
```

## Pascal

```
program HelloWorld;  
  
begin  
  WriteLn('Hello, world!');  
end.
```

## Lisp

```
(print "Hello, world!")
```

## Ocaml

```
print_endline "Hello, world!"
```

## Haskell

```
main = putStrLn "Hello, world!"
```

## C

```
#include <stdio.h>  
  
int main(void)  
{  
    puts("Hello, world!");  
}
```

## C++

```
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello, world!\n";  
}
```

## C#

```
using System;  
class Program  
{  
    public static void Main(string[] args)  
    {  
        Console.WriteLine("Hello, world!");  
    }  
}
```

## Clojure

```
(println "Hello, world!")
```

## JAVA

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

## Python

```
print "Hello, world!"
```

## Javascript

```
console.log('Hello, world!');
```

## SCALA

```
object HelloWorld extends App {  
    println("Hello, world!")  
}
```

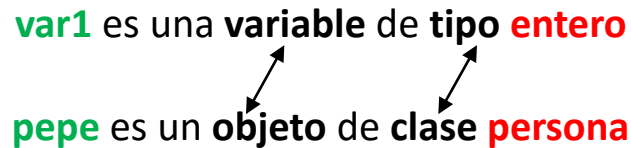
# Antes de ver la versión Java...

## ...aclaremos la cuestión CLASE / OBJETO

**Clase** es a **tipo** como **objeto** es a **variable**

```
int var1;  
Persona pepe;
```

**var1** es una **variable** de **tipo entero**  
**pepe** es un **objeto** de **clase persona**



Una **clase** es un “**tipo complejo**”; una agrupación de **variables** (constantes), **objetos**, e incluso **código** que puede actuar sobre sus propios elementos u otros.

Un **objeto** es una cápsula (de memoria de ordenador) que tiene un “estado” (determinado por los valores de sus variables y el estado de sus objetos) así como un comportamiento (definido por el código que encierra).

La clase es la “definición” a partir de la cual se generan (“instancian”) los elementos que participan en la aplicación. (esto tiene cuestiones de detalle que se matizarán más adelante)

HolaMundo.java

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

HolaMundo.java

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

Clase

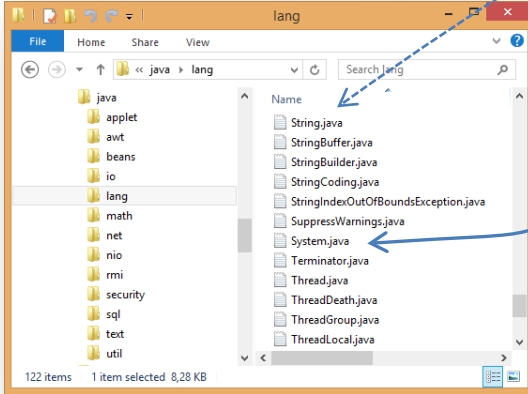
Método

HolaMundo.java

Clase

Método

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



System.java

```
/**  
 * The <code>System</code> class contains several useful class fields  
 * and methods. It cannot be instantiated.  
 *  
 * <p>Among the facilities provided by the <code>System</code> class  
 * are standard input, standard output, and error output streams;  
 * access to externally defined properties and environment  
 * variables; a means of loading files and libraries; and a utility  
 * method for quickly copying a portion of an array.  
 *  
 * @author unascrbed  
 * @since JDK1.0  
 */  
public final class System {  
  
    /* register the natives via the static initializer.  
     * VM will invoke the initializeSystemClass method to complete  
     * the initialization for this class separated from clinit.  
     * Note that to use properties set by the VM, see the constraints  
     * described in the initializeSystemClass method.  
     */  
    private static native void registerNatives();  
    static {  
        registerNatives();  
    }  
  
    /* Don't let anyone instantiate this class */  
    private System() {  
    }  
  
    /**  
     * The "standard" input stream. This stream is already  
     * open and ready to supply input data. Typically this stream  
     * corresponds to keyboard input or another input source specified by  
     * the host environment or user.  
     */  
    public final static InputStream in = null;  
  
    /**  
     * The "standard" output stream. This stream is already  
     * open and ready to accept output data. Typically this stream  
     * corresponds to display output or another output destination  
     * specified by the host environment or user.  
     * <p>  
     * For simple stand-alone Java applications, a typical way to write  
     * a line of output data is:  
     * <blockquote><pre>  
     *     System.out.println(data)  
     * </pre></blockquote>  
     * <p>  
     * See the <code>println</code> methods in class <code>PrintStream</code>.  
     *  
     * @see java.io.PrintStream.println()  
     * @see java.io.PrintStream.println(boolean)  
     * @see java.io.PrintStream.println(char)  
     * @see java.io.PrintStream.println(char[])  
     * @see java.io.PrintStream.println(double)  
     * @see java.io.PrintStream.println(float)  
     * @see java.io.PrintStream.println(int)  
     * @see java.io.PrintStream.println(long)  
     * @see java.io.PrintStream.println(java.lang.Object)  
     * @see java.io.PrintStream.println(java.lang.String)  
     */  
    public final static PrintStream out = null;  
}
```

HolaMundo.java

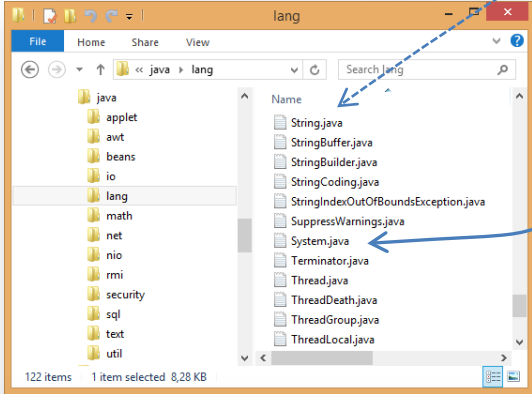
```

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

```

Clase

Método



System.java

```

/**
 * The <code>System</code> class contains several useful class fields
 * and methods. It cannot be instantiated.
 *
 * <p>Among the facilities provided by the <code>System</code> class
 * are standard input, standard output, and error output streams;
 * access to externally defined properties and environment
 * variables; a means of loading files and libraries; and a utility
 * method for quickly copying a portion of an array.
 *
 * @author unascr@red
 * @since JDK1.0
 */
public final class System {

    /* register the natives via the static initializer.
     * VM will invoke the initializeSystemClass method to complete
     * the initialization for this class separated from clinit.
     * Note that to use properties set by the VM, see the constraints
     * described in the initializeSystemClass method.
     */
    private static native void registerNatives();
    static {
        registerNatives();
    }

    /** Don't let anyone instantiate this class */
    private System() {
    }

    /**
     * The "standard" input stream. This stream is already
     * open and ready to supply input data. Typically this stream
     * corresponds to keyboard input or another input source specified by
     * the host environment or user.
     */
    public final static InputStream in = null;

    /**
     * The "standard" output stream. This stream is already
     * open and ready to accept output data. Typically this stream
     * corresponds to display output or another output destination
     * specified by the host environment or user.
     *
     * <p>
     * For simple stand-alone Java applications, a typical way to write
     * a line of output data is:
     * <blockquote><pre>
     *     System.out.println(data)
     * </pre></blockquote>
     * <p>
     * See the <code>println</code> methods in class <code>PrintStream</code>.
     *
     * @see java.io.PrintStream#println()
     * @see java.io.PrintStream#println(boolean)
     * @see java.io.PrintStream#println(char)
     * @see java.io.PrintStream#println(char[])
     * @see java.io.PrintStream#println(double)
     * @see java.io.PrintStream#println(float)
     * @see java.io.PrintStream#println(int)
     * @see java.io.PrintStream#println(long)
     * @see java.io.PrintStream#println(java.lang.Object)
     * @see java.io.PrintStream#println(java.lang.String)
     */
    public final static PrintStream out = null;
}

```

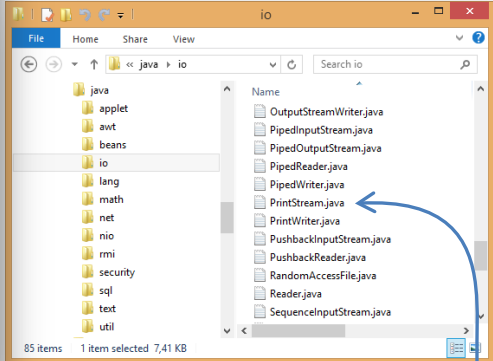
```

/**
 * A <code>PrintStream</code> adds functionality to another output stream,
 * namely the ability to print representations of various data values
 * conveniently. Two other features are provided as well. Unlike other output
 * streams, a <code>PrintStream</code> never throws an
 * <code>IOException</code>; instead, exceptional situations merely set an
 * internal flag that can be tested via the <code>checkError</code> method.
 * Optionally, a <code>PrintStream</code> can be created so as to flush
 * automatically; this means that the <code>flush</code> method is
 * automatically invoked after a byte array is written, one of the
 * <code>println</code> methods is invoked, or a newline character or byte
 * (<code>\n</code>) is written.
 *
 * <p> All characters printed by a <code>PrintStream</code> are converted into
 * bytes using the platform's default character encoding. The <code>@link
 * <code>PrintWriter</code> class should be used in situations that require writing
 * characters rather than bytes.
 *
 * @author Frank Yellin
 * @author Mark Reinhold
 * @since JDK1.0
 */
public class PrintStream extends FilterOutputStream
    implements Appendable, Closeable
{
    //!!!!!! CODIGO BORRADO !!!!!!!

    /**
     * Prints a String and then terminate the line. This method behaves as
     * though it invokes <code>@link #print(String)</code> and then
     * <code>@link #println()</code>.
     *
     * @param x The <code>String</code> to be printed.
     */
    public void println(String x) {
        synchronized (this) {
            print(x);
            newline();
        }
    }
}

```

PrintStream.java



Comprobar el funcionamiento:

Escritura de `GeneradorDeHolaMundo.java`

En shell del SO: compilar, ejecutar, desensamblar, documentar.

Idem con `package`

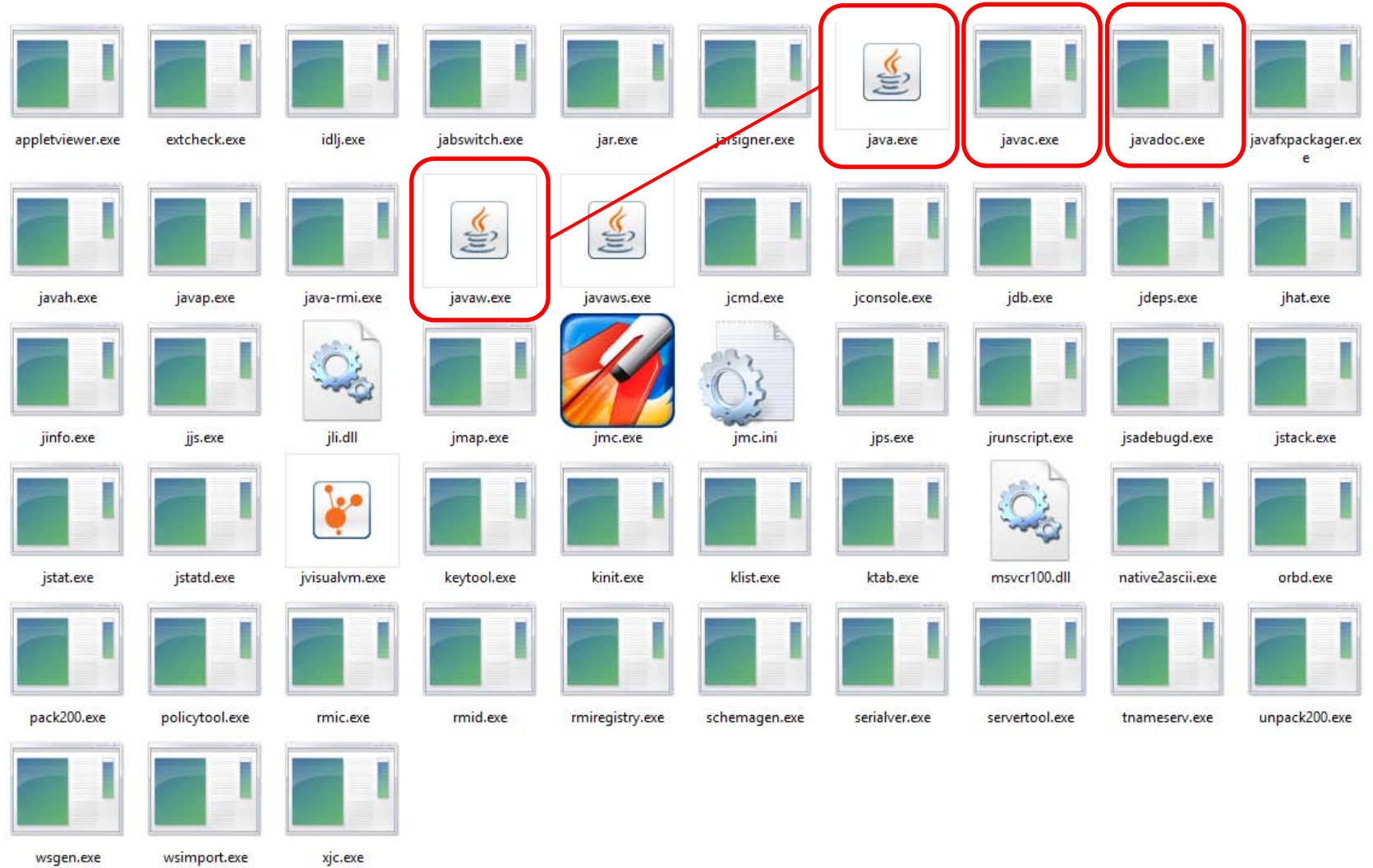
Decompilar

Idem en Netbeans.



# Las herramientas de ejecución y desarrollo de Java

El "bin"



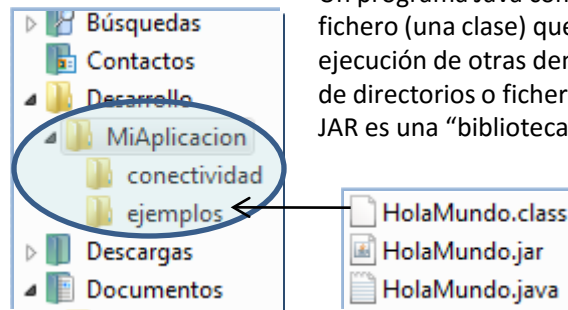
# Compilación y ejecución

```
C:\Documents and Settings\german>javac
Usage: javac <options> <source files>
where possible options include:
-g Generate all debugging info
-g:none Generate no debugging info
-g:<lines,vars,source> Generate only some debugging info
-nowarn Generate no warnings
-verbose Output messages about what the compiler is doing
-deprecation Output source locations where deprecated APIs are used
-classpath <path> Specify where to find user class files and annotation processors
-cp <path> Specify where to find user class files and annotation processors
-sourcepath <path> Specify where to find input source files
-bootclasspath <path> Override location of bootstrap class files
-extdirs <dirs> Override location of installed extensions
-endorseddirs <dirs> Override location of endorsed standards path
-processor <class1>[,<class2>,<class3>...] Names of the annotation processors to use
-processorpath <path> Specify where to find annotation processors
-d <directory> Specify where to place generated class files
-s <directory> Specify where to place generated source files
-implicit:<none,class> Specify whether or not to generate class files for implicit dependencies
-encoding <encoding> Specify character encoding used by source files
-source <release> Provide source compatibility with specified release
-target <release> Generate class files for specific VM version
-version Version information
-help Print a synopsis of standard options
-Akey[=value] Options to pass to annotation processors
-X Print a synopsis of nonstandard options
-J<flag> Pass <flag> directly to the runtime system
```

```
C:\Documents and Settings\german>java
Usage: java [-options] class [args...]
           (to execute a class)
or java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
-client to select the "client" VM
-server to select the "server" VM
-hotspot is a synonym for the "client" VM (deprecated!)
The default VM is client.

-cp <class search path of directories and zip/jar files>
-classpath <class search path of directories and zip/jar files>
           A ; separated list of directories, JAR archives,
           and ZIP archives to search for class files.
-D<name>=<value> set a system property
-verbose[:class[:gc[:jni]] enable verbose output
-version print product version and exit
-version:<value> require the specified version to run
-showversion print product version and continue
-jre-restrict-search | -jre-no-restrict-search
           include/exclude user private JREs in the version search
-? -help print this help message
-X print help on non-standard options
-ea[:<packagename>...[:<classname>]]
-enableassertions[:<packagename>...[:<classname>]]
           enable assertions
-dal[:<packagename>...[:<classname>]]
-disableassertions[:<packagename>...[:<classname>]]
           disable assertions
-esa | -enablesystemassertions
           enable system assertions
-dsa | -disablesystemassertions
           disable system assertions
-agentlib:<libname>[=<options>]
           load native agent library <libname>, e.g. -agentlib:hprof
           see also, -agentlib:jdwp=help and -agentlib:hprof=help
-agentpath:<pathname>[=<options>]
           load native agent library by full pathname
-javaagent:<jarpath>[=<options>]
           load Java programming language agent, see java.lang.instrument
-plash:<imagepath>
           show splash screen with specified image
```

Un programa Java comienza por la ejecución de un fichero (una clase) que puede ir invocando la ejecución de otras dentro de uno o varios subárboles de directorios o ficheros JAR (cada subárbol o cada JAR es una "biblioteca de clases")



## Compilar y ejecutar estando en "MiAplicacion"

```
>javac ejemplos/HolaMundo.java
>java ejemplos.HolaMundo
```

## Compilar y ejecutar estando en otro directorio

```
>javac -cp c:/Desarrollo/MiAplicacion ejemplos/HolaMundo.java
>java -cp c:/Desarrollo/MiAplicacion ejemplos.HolaMundo
```

## Ejecutar mediante un JAR

```
>java -cp c:/Desarrollo/MiAplicacion/ejemplos/HolaMundo.jar HolaMundo
>java -jar c:/Desarrollo/MiAplicacion/ejemplos/HolaMundo.jar
```

← OJO: para hacer esta prueba hay que incluir la línea **package ejemplos;** en el código fuente de HolaMundo.java

# Documentación

The screenshot shows a web browser window displaying the Java Platform Standard Ed. 7 API documentation for the `System` class. The browser address bar shows the URL `docs.oracle.com/javase/7/docs/api/index.html?java/lang/System.html`. The page title is "System (Java Platform SE 7)". The navigation menu includes "Overview", "Package", "Class", "Use", "Tree", "Deprecated", "Index", and "Help". The "Class" tab is selected. The left sidebar shows a list of "All Classes" and "Packages". The main content area displays the class hierarchy for `java.lang.System`, which extends `java.lang.Object`. The class signature is `public final class System extends Object`. A description states that the `System` class contains several useful class fields and methods and cannot be instantiated. It lists facilities provided by the class, such as standard input, output, and error streams, and access to environment variables. The "Since" section indicates that the class is available since JDK 1.0. Below the description, there are sections for "Field Summary" and "Method Summary". The "Field Summary" section shows a table with three fields: `err`, `in`, and `out`, each with a description of its role as a standard stream.

System (Java Platform SE 7)

docs.oracle.com/javase/7/docs/api/index.html?java/lang/System.html

Java™ Platform Standard Ed. 7

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

## Class System

java.lang.Object  
java.lang.System

---

```
public final class System
extends Object
```

The `System` class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the `System` class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:

JDK1.0

### Field Summary

**Fields**

Modifier and Type	Field and Description
static <code>PrintStream</code>	<code>err</code> The "standard" error output stream.
static <code>InputStream</code>	<code>in</code> The "standard" input stream.
static <code>PrintStream</code>	<code>out</code> The "standard" output stream.

### Method Summary

**Methods**

Modifier and Type	Method and Description
-------------------	------------------------

# Los programas ejecutables Java. Desensamblado de codebytes y decompilación

Desensamblado y decompilación

```
C:>javap -c HolaMundo
Compiled from "HolaMundo.java"
public class HolaMundo extends java.lang.Object{
  public HolaMundo();
    Code:
      0:   aload_0
      1:   invokespecial   #1; //Method java/lang/Object."<init>":()V
      4:   return

  public static void main(java.lang.String[]);
    Code:
      0:   getstatic       #2; //Field java/lang/System.out:Ljava/io/PrintStream;
      3:   ldc            #3; //String Hola, mundo
      5:   invokevirtual  #4; //Method java/io/PrintStream.println:(Ljava/lang/String;)V
      8:   return
}
```

Decompilación: probar con [Java Optimize and Decompile Environment \(JODE\)](#)

“Ofuscación”