

1.E. Programando con Java.

1. Programas en Java.

1.4. Afinando la configuración.

Para que podamos compilar y ejecutar ficheros Java es necesario que realicemos unos pequeños ajustes en la configuración del sistema. Vamos a indicarle dónde encontrar los ficheros necesarios para realizar las labores de compilación y ejecución, en este caso **Javac.exe** y **Java.exe**, así como las librerías contenidas en la API de Java y las clases del usuario.

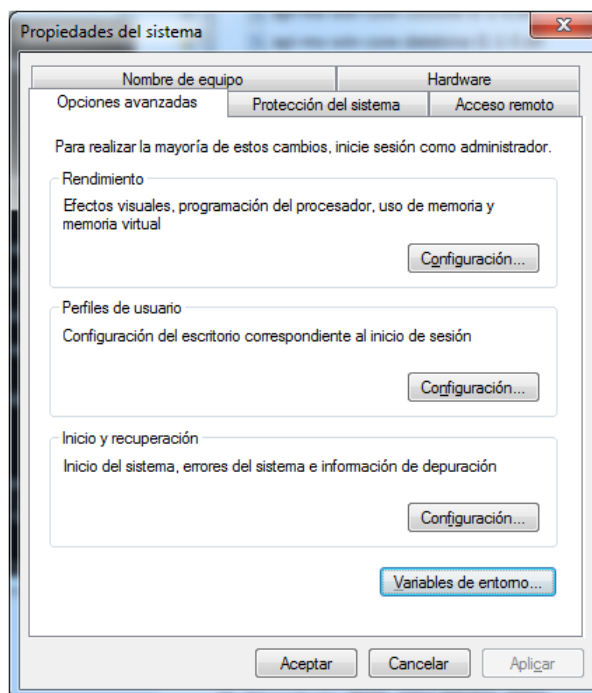


Imagen extraída de curso Programación del MECD.

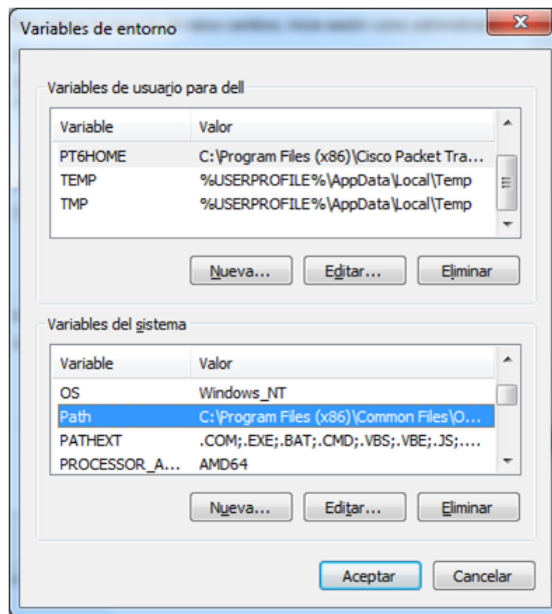
La variable PATH: Como aún no disponemos de un IDE (Integrated Development Environment - Entorno Integrado de Desarrollo) la única forma de ejecutar programas es a través de línea de comandos. Pero sólo podremos ejecutar programas directamente si la ruta hacia ellos está indicada en la variable PATH del ordenador. **Es necesario que incluyamos la ruta hacia estos programas en nuestra variable PATH.** Esta ruta será el lugar donde se instaló el JDK hasta su directorio **bin**.

En Windows:

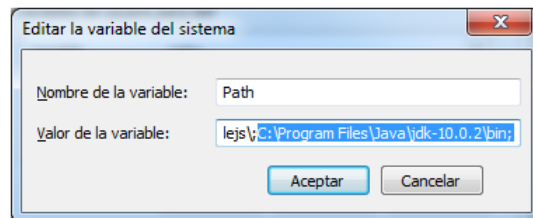
Vamos a las propiedades del sistema:



Vamos a las Variables de entorno...:



En la variable Path, damos a Editar... e incorporamos la ruta a la carpeta bin del JDK, la cual contiene los ficheros "javac.exe" y "java.exe".



Damos a Aceptar, y a continuación abrimos el interprete de comandos y en él, debemos probar tanto el comando "javac.exe" como el comando "java.exe".

javac.exe funciona:

```
C:\Windows\system32\cmd.exe
C:\Users\dell>javac
Usage: javac <options> <source files>
where possible options include:
  @<filename>          Read options and filenames from file
  -Akey[=value]        Options to pass to annotation processors
  --add-modules <module><,<module>>*
                        Root modules to resolve in addition to the initial modules, or all modules
  --boot-class-path <path>, -bootclasspath <path>
                        Override location of bootstrap class files
  --class-path <path>, -classpath <path>, -cp <path>
                        Specify where to find user class files and annotation processors
  -d <directory>        Specify where to place generated class files
  -deprecation          Output source locations where deprecated APIs are used
  -encoding <encoding>  Specify character encoding used by source files
  -endorseddirs <dirs>  Override location of endorsed standards path
  -extdirs <dirs>       Override location of installed extensions
  -g                   Generate all debugging info
  -g:{lines,vars,source}
                        Generate only some debugging info
  -g:none              Generate no debugging info
  -h <directory>        Specify where to place generated native header files
  --help, -help        Print this help message
  --help-extra, -X      Print help on extra options
  -implicit:{none,class}
                        Specify whether or not to generate class files for implicitly referenced
  files
  -J<flag>              Pass <flag> directly to the runtime system
  --limit-modules <module><,<module>>*
                        Limit the universe of observable modules
  --module <module-name>, -m <module-name>
                        Compile only the specified module, check timestamps
  --module-path <path>, -p <path>
                        Specify where to find application modules
  --module-source-path <module-source-path>
                        Specify where to find input source files for multiple modules
  --module-version <version>
                        Specify version of modules that are being compiled
  -nowarn               Generate no warnings
  -parameters          Generate metadata for reflection on method parameters
  -proc:{none,only}     Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...]
                        Names of the annotation processors to run; bypasses default discovery process
  --processor-module-path <path>
                        Specify a module path where to find annotation processors
  --processor-path <path>, -processorpath <path>
                        Specify where to find annotation processors
  -profile <profile>     Check that API used is available in the specified profile
  --release <release>    Compile for a specific VM version. Supported targets: 10, 6, 7, 8, 9
  -s <directory>        Specify where to place generated source files
  -source <release>      Provide source compatibility with specified release
  --source-path <path>, -sourcepath <path>
                        Specify where to find input source files
  --system <jdk>:none    Override location of system modules
  -target <release>      Generate class files for specific VM version
  --upgrade-module-path <path>
                        Override location of upgradeable modules
  -verbose              Output messages about what the compiler is doing
  -version, -version    Version information
  -Werror               Terminate compilation if warnings occur
```

y java.exe también funciona:

```
C:\Windows\system32\cmd.exe
C:\Users\dell>java
Sintaxis: java [opciones] <clase principal> [argumentos...]
    <para ejecutar una clase>
o java [opciones] -jar <archivo jar> [argumentos...]
    <para ejecutar un archivo jar>
o java [opciones] -m <módulo>[/<clase principal>] [argumentos...]
    java [opciones] --module <módulo>[/<clase principal>] [argumentos...]
    <para ejecutar la clase principal en un módulo>

Argumentos que siguen la clase principal, -jar <archivo jar>, -m o --module
<módulo>/<clase principal> se transfieren como argumentos a una clase principal
.

donde las opciones incluyen:

-cp <ruta de búsqueda de clase de directorios y archivos zip/jar>
-classpath <ruta de búsqueda de clase de directorios y archivos zip/jar>
--class-path <ruta de búsqueda de clase de directorios y archivos zip/jar>
    Una lista separada por el carácter ;, archivos JAR
    y archivos ZIP para buscar archivos de clases.
-p <ruta módulo>
--module-path <ruta módulo>...
    Una lista de directorios separada por el carácter ;, cada dire
    ctorio
    es un directorio de módulos.
--upgrade-module-path <ruta módulo>...
    Una lista de directorios separada por el carácter ;, cada dire
    ctorio
    es un directorio de módulos que sustituye a
    los módulos actualizables en la imagen de tiempo de ejecución
--add-modules <nombre módulo>[/<nombre módulo>...]
    módulos de raíz que resolver, además del módulo inicial.
    <nombre módulo> también puede ser ALL-DEFAULT, ALL-SYSTEM,
    ALL-MODULE-PATH.
--list-modules
    mostrar módulos observables y salir
-d <nombre de módulo>
--describe-module <nombre módulo>
    describir un módulo y salir
--dry-run
    crear VM y cargar la clase principal pero sin ejecutar el méto
do principal.
    La opción --dry-run puede ser útil para validar
    las opciones de línea de comandos, como la configuración del s
    istema de módulos.
--validate-modules
    validar todos los módulos y salir
    La opción --validate-modules puede ser útil para encontrar
    conflictos y otros errores con módulos en la ruta de módulos.
-D<nombre>=<valor>
    definir una propiedad de sistema
-verbose:[class|module|gc|jnil
    activar la salida en modo verbose
--version
    imprimir versión de producto en el flujo de errores y salir
--version
    imprimir versión de producto en el flujo de salida y salir
--showversion
    imprimir versión de producto en el flujo de errores y continua
r
--show-version
    --showversion imprimir versión de producto en el flujo de sali
da y continuar
--show-module-resolution
    mostrar la salida de resolución de módulo durante el inicio
-? -h -help
    imprimir este mensaje de ayuda en el flujo de errores
--help
    imprimir este mensaje de ayuda en el flujo de salida
-X
    imprimir ayuda de opciones adicionales en el flujo de errores
--help-extra
    imprimir ayuda de opciones adicionales en el flujo de salida
-ea[:<nombre paquete>...[:<nombre clase>]
-enableassertions[:<nombre paquete>...[:<nombre clase>]
    activar afirmaciones con una granularidad especificada
```

Aquí tienes unos enlaces donde se describe esto con más detalle:

- Variables de entorno para Java en Windows

CONFIGURAR VARIABLES DE ENTORNO JAVA Windows 7 8 8.1 10



- Variables de entorno para Java en Ubuntu (versión antigua)

<http://chicomonte.blogspot.com/2010/04/instalar-java-jdk-jre-en-ubuntu.html>

La variable **CLASSPATH**: esta variable de entorno establece dónde buscar las clases o bibliotecas de la API de Java, así como las clases creadas por el usuario. Es decir, los ficheros **.class** que se obtienen una vez compilado el código fuente de un programa escrito en Java. Es posible que en dicha ruta existan directorios y ficheros comprimidos en los formatos **zip** o **jar** que pueden ser utilizados directamente por el JDK, conteniendo en su interior archivos con extensión **class**.

(Por ejemplo: **C:\Program Files\Java\jdk1.6.0_25\bin**)

Si no existe la variable **CLASSPATH** debes crearla, para modificar su contenido sigue el mismo método que hemos empleado para la modificación del valor de la variable **PATH**, anteriormente descrito. Ten en cuenta que la ruta que debes incluir será el lugar donde se instaló el JDK hasta su directorio **lib**.

(Por ejemplo: **C:\Program Files\Java\jdk1.6.0_25\lib**)

Autoevaluación

¿Qué variable de sistema o de entorno debemos configurar correctamente para que podamos compilar directamente desde la línea de comandos nuestros programas escritos en lenguaje Java?

- ☐ CLASSPATH.
- ☒ PATH.
- ☐ Javac.exe.