SAD DOCUMENTACIÓ PRÀCTICA 1 26/02/2019

* **Build and run eclipse java projects from the command line**



**Objective:**

Simple javac and java commands that I could run (or add to a Makefile) such that the .class files generated would go into the bin folder created in the eclipse project.

**Building the project:**

Run the following command to compile your .java file in src and put them in bin

javac -d bin/ -cp src /path/to/java/file

My package was called cs5223 and I was trying to build cs5223.Server.java (which was the entry point into my project).

javac -d bin/ -cp src src/cs5223/Server.java

**Running the project:**

Once you've compiled your .java files, you run them using the following command.

java -cp bin JAVA\_CLASS

My Server class had the following fully qualified package path: cs5223.Server so I ran the following command.

java -cp bin cs5223.Server

**Problem #1:**

* "Error: Could not find or load main class XXX"

This error comes when you are trying to run your Java program using java command with the main class as HelloWorld but Java is not able to find the class. In order to solve this error, you must know how Java find and loads the classes, that's a little bit complex topic for beginners, but we will touch the same base here.

In Eclipse, it's easy to compile and run the program because Eclipse takes care of all Classpath setup, but when you run your Java program from command line, CLASSPATH environment variable comes in picture.  
  
Personally, I don't like this environment variable and doesn't define in my environment variable, because its confusing and source of so many classpath related issue. Instead, I use -cp or -classpath option with **java** command to run my program. This way you always know which JARs are included in your classpath.

Another important thing to understand is the [difference between PATH and CLASSPATH](http://java67.blogspot.sg/2012/08/what-is-path-and-classpath-in-java-difference.html), you must know that PATH is used locate system executable, commands or .exe, .dll files (in Windows) and .so files (in Linux). It is also used to locate native libraries used by your Java program. While, CLASSPATH is used to locate the class file or JAR files. It's Java class loader who looked into CLASSPATH for loading classes.

**Example of solution:**

If your class is inside a[non-default package](http://java67.blogspot.sg/2012/08/what-is-package-in-java-how-to-use.html) e.g. "dto" in our case then compiler the will put the HelloHP.class file, which contains Java bytecode in a directory named "dto". In our case the full name of class dto.HelloHP and it is present in

**C:\Users\WINDOWS 8\workspace\Demo\target\classes\dto**.



Still same error, right. Why? because I don't have any CLASSPATH environment variable, neither I am using -classpath or -cp option to suggest the path, So *by default Java is only searching in the current directory*. It is looking for dto/HelloHP.class but since we are already inside dto, it is  not able to find the class. So, what should we do now? let's go to the parent directory

**"C:\Users\WINDOWS 8\workspace\Demo\target\classes"**

And execute the same command, this time, it should work:





There are many ways:

**Error: Could not find or load main class HelloWorld**manifests itself, but if you know the [basics of Java Classpath](http://javarevisited.blogspot.sg/2011/08/classnotfoundexception-in-java-example.html), you can easily sort out the problem.

Most of the time you just need to either correct your CLASSPATH environment variable or run your program with java -cp or -classpath option.

* **IMPLEMENTACIÓ DEL MODE RAW I COOKED:**

**TTY**

The TTY subsystem is central to the design of Linux, and UNIX in general. Unfortunately, its importance is often overlooked, and it is difficult to find good introductory articles about it. I believe that a basic understanding of TTYs in Linux is essential for the developer and the advanced user.

Line editing. Most users make mistakes while typing, so a backspace key is often useful. This could of course be implemented by the applications themselves, but in accordance with the UNIX design philosophy, applications should be kept as simple as possible. So as a convenience, the operating system provides an editing buffer and some rudimentary editing commands (backspace, erase word, clear line, reprint), which are enabled by default inside the line discipline. Advanced applications may disable these features by putting the line discipline in raw mode instead of the default cooked (or canonical) mode.

Most interactive applications (editors, mail user agents, shells, all programs relying on curses or readline) run in raw mode, and handle all the line editing commands themselves. The line discipline also contains options for character echoing and automatic conversion between carriage returns and linefeeds. Think of it as a primitive kernel-level sed(1), if you like.

Incidentally, the kernel provides several different line disciplines. Only one of them is attached to a given serial device at a time. The default discipline, which provides line editing, is called N\_TTY (drivers/char/n\_tty.c, if you're feeling adventurous). Other disciplines are used for other purposes, such as managing packet switched data (ppp, IrDA, serial mice), but that is outside the scope of this article.

**stty -echo;**

As you type, your terminal emulator transmits information to the kernel. Usually, the kernel echoes the same information back to the terminal emulator, allowing you to see what you type. Without character echoing, you can't see what you type, but we're in cooked mode so the line editing facilities are still working. Once you press enter, the line discipline will transmit the edit buffer to cat, which will reveal what your wrote.

Finally, stty sane will restore your TTY device configuration to something reasonable.

**Runtime.exec()**

public [Process](https://docs.oracle.com/javase/7/docs/api/java/lang/Process.html) exec([String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)[] cmdarray,

[String](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html)[] envp,

[File](https://docs.oracle.com/javase/7/docs/api/java/io/File.html) dir)

throws [IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html)

Executes the specified command and arguments in a separate process with the specified environment and working directory.

Given an array of strings cmdarray, representing the tokens of a command line, and an array of strings envp, representing "environment" variable settings, this method creates a new process in which to execute the specified command.

This method checks that cmdarray is a valid operating system command. Which commands are valid is system-dependent, but at the very least the command must be a non-empty list of non-null strings.

If envp is null, the subprocess inherits the environment settings of the current process.

A minimal set of system dependent environment variables may be required to start a process on some operating systems. As a result, the subprocess may inherit additional environment variable settings beyond those in the specified environment.

[ProcessBuilder.start()](https://docs.oracle.com/javase/7/docs/api/java/lang/ProcessBuilder.html#start()) is now the preferred way to start a process with a modified environment. The working directory of the new subprocess is specified by dir. If dir is null, the subprocess inherits the current working directory of the current process.

If a security manager exists, its [checkExec](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityManager.html" \l "checkExec(java.lang.String)) method is invoked with the first component of the array cmdarray as its argument. This may result in a [SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html" \o "class in java.lang) being thrown.

Starting an operating system process is highly system-dependent. Among the many things that can go wrong are:

- The operating system program file was not found.

- Access to the program file was denied.

- The working directory does not exist.

In such cases an exception will be thrown. The exact nature of the exception is system-dependent, but it will always be a subclass of [IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html" \o "class in java.io).

**Parameters**:

cmdarray - array containing the command to call and its arguments.

envp - array of strings, each element of which has environment variable settings in the format *name*=*value*, or null if the subprocess should inherit the environment of the current process.

dir - the working directory of the subprocess, or null if the subprocess should inherit the working directory of the current process.

**Returns**:

A new [Process](https://docs.oracle.com/javase/7/docs/api/java/lang/Process.html) object for managing the subprocess

**Throws**:

[SecurityException](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityException.html) - If a security manager exists and its [checkExec](https://docs.oracle.com/javase/7/docs/api/java/lang/SecurityManager.html" \l "checkExec(java.lang.String)) method doesn't allow creation of the subprocess

[IOException](https://docs.oracle.com/javase/7/docs/api/java/io/IOException.html) - If an I/O error occurs

[NullPointerException](https://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html) - If cmdarray is null, or one of the elements of cmdarray is null, or one of the elements of envp is null

[IndexOutOfBoundsException](https://docs.oracle.com/javase/7/docs/api/java/lang/IndexOutOfBoundsException.html) - If cmdarray is an empty array (has length 0)

**STDIO – STDOUT problematic:**

Since the JVM redirects stdio/stdout/stderr, and note that stty (usually) operates on stdin not stdout, you have to:

String[] cmd = {"/bin/sh", "-c", "stty raw </dev/tty"};

SET RAW

UNSET RAW

READ LINE I READ

LINE

**Trobar el nombre de columnes i files**

<https://unix.stackexchange.com/questions/185509/how-do-i-determine-the-number-of-rows-and-columns-in-a-gnome-terminal-window-whi>

**PROBLEMÀTICA READ:**

Cal observar que les lletres A-Z van des del enter 65 al 90

Les lletres a-z van des del enter 97-122

FLETXA ADALT -> ^[[A

AVALL -> ^[[B

DRETA -> ^[[C

ESQUERRA -> ^[[D

SUPRIMIR -> ^[[3~

FIN -> ^[[F

INICIO -> ^[[H

INSERT -> ^[[2~

ESC -> ^[[