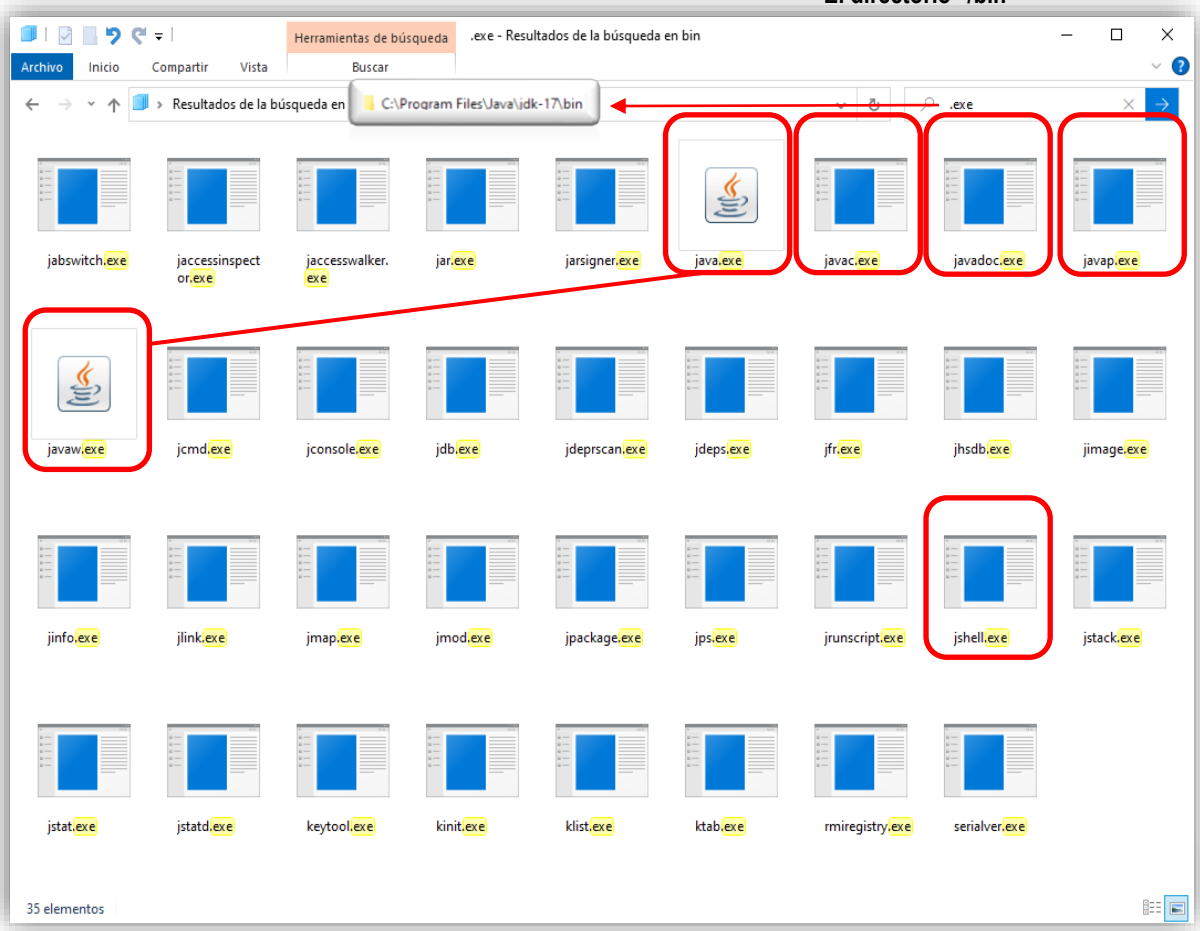
¿QUÉ TENDRÉ A MI DISPOSICIÓN PARA DESARROLLAR EN JAVA?

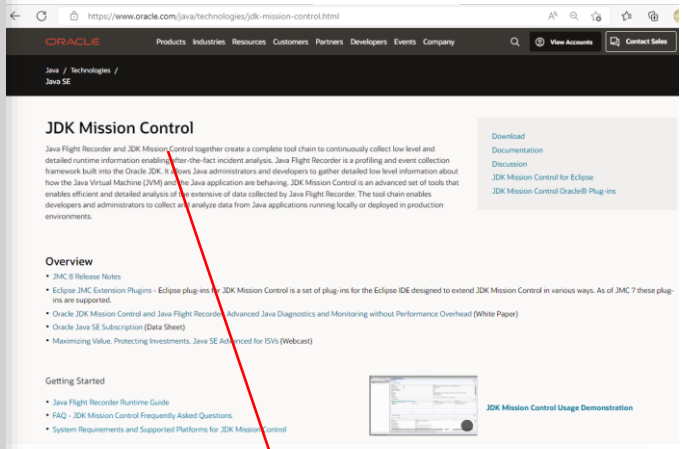
(COMPILACIÓN, EJECUCIÓN, ETC.)

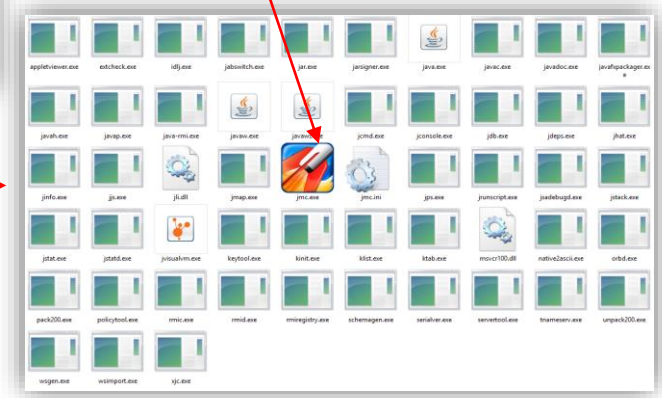
Las herramientas de ejecución y desarrollo de Java

El directorio "/bin"



Java Mission Control (JMC)
Se obtiene ahora por separado
[Ojo a parámetro de instalación]





En una versión anterior…

En distintas versiones van cambiando herramientas, pero en todo caso pueden obtenerse las que no se encuentran por defecto.
Las imprescindibles las encontramos siempre.

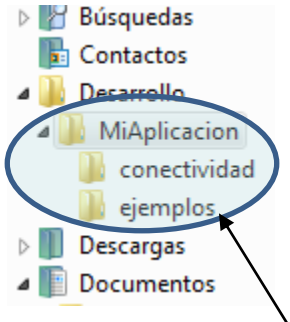
Compilación y ejecución

```
D:\>java
Usage: java [options] <mainclass> [args...]
           (to execute a class)
   or  java [options] -jar <jarfile> [args...]
           (to execute a jar file)
   or  java [options] -m <module>[/<mainclass>] [args...]
       java [options] --module <module>[/<mainclass>] [args...]
           (to execute the main class in a module)
   or  java [options] <sourcefile> [args]
           (to execute a single source-file program)

 Arguments following the main class, source file, -jar <jarfile>,
 -m or --module <module>/<mainclass> are passed as the arguments to
 main class.

 where options include:

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
    --class-path <class search path of directories and zip/jar files>
                  A ; separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -p <module path>
    --module-path <module path>...
                  A ; separated list of directories, each directory
                  is a directory of modules.
    --upgrade-module-path <module path>...
```

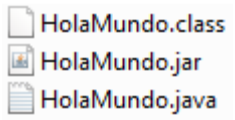
```
D:\>java -version
java version "17" 2021-09-14 LTS
Java(TM) SE Runtime Environment (build 17+35-LTS-2724)
Java HotSpot(TM) 64-Bit Server VM (build 17+35-LTS-2724, mixed mode, sharing)
```

```
D:\>javac -version
javac 17
```

```
D:\>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
        on the module path if <module> is ALL-MODULE-PATH.
  --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
  --enable-preview
        Enable preview language features. To be used in conjunction with either -source or --release.
  -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
```



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
```

Comprobaremos todo esto… escribiendo el HolaMundo.java, compilando, ejecutando, desensamblando, decompilando (http://www.javadecompilers.com/)
Haremos lo mismo con Netbeans. (http://java-decompiler.github.io/)
**Importante: el uso de packages. __Lo contrario: ofuscación__/

Los programas ejecutables Java.

Desensamblado de codebytes y decompilación

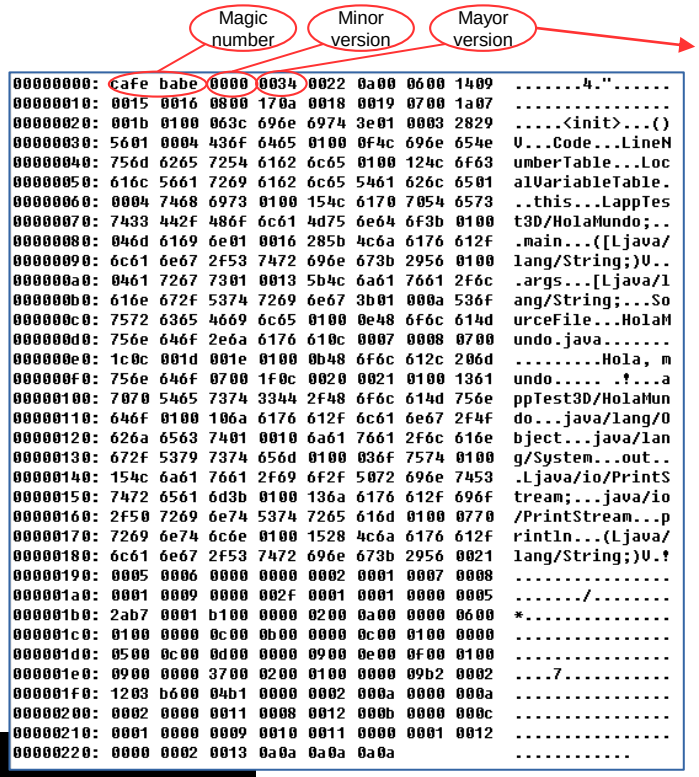
HolaMundo.java

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

HolaMundo.class

```
Êþº¾^@^@^@4^@"
^@^F^@^T        ^@^U^@^V^H^@^W
^@^X^@^Y^G^@^Z^G^@^[^A^@^F<init>^A^@^C()V^A^@^DCode^A^@^OLineNumberTable^A^
@^RLocalVariableTable^A^@^Dthis^A^@^ULappTest3D/HolaMundo;^A^@^Dmain^A^@^V(
[Ljava/lang/String;)V^A^@^Dargs^A^@^S[Ljava/lang/String;^A^@
SourceFile^A^@^NHolaMundo.java^L^@^G^@^H^G^@^\^L^@^]^@^^^A^@^KHola, mundo^G
^@^_^L^@ ^@!^A^@^SappTest3D/HolaMundo^A^@^Pjava/lang/Object^A^@^Pjava/lang/
System^A^@^Cout^A^@^ULjava/io/PrintStream;^A^@^Sjava/io/PrintStream^A^@^Gpr
intln^A^@^U(Ljava/lang/String;)V^@!^@^E^@^F^@^@^@^@^B^@^A^@^G^@^H^@^A^@
    ^@^@^@/^@^A^@^A^@^@^@^E*·^@^A±^@^@^@^B^@
^@^@^@^F^@^A^@^@^@^L^@^K^@^@^@^L^@^A^@^@^@^E^@^L^@^M^@^@^@      ^@^N^@^O^@^
A^@  ^@^@^@7^@^B^@^A^@^@^@   ²^@^B^R^C¶^@^D±^@^@^@^B^@
^@^@^@
^@^B^@^@^@^Q^@^H^@^R^@^K^@^@^@^L^@^A^@^@^@      ^@^P^@^Q^@^@^@^A^@^R^@^@^@^
B^@^S
```

=



Java SE 19 = 62 (0x3F)
Java SE 18 = 61 (0x3E)
Java SE 17 = 61 (0x3D)
Java SE 16 = 60 (0x3C)
Java SE 15 = 59 (0x3B)
Java SE 14 = 58 (0x3A)
Java SE 13 = 57 (0x39)
Java SE 12 = 56 (0x38)
Java SE 11 = 55 (0x37)
Java SE 10 = 54 (0x36)
Java SE 9 = 53 (0x35)
Java SE 8 = 52 (0x34)
Java SE 7 = 51 (0x33)
Java SE 6.0 = 50 (0x32)
Java SE 5.0 = 49 (0x31)
JDK 1.4 = 48 (0x30)
JDK 1.3 = 47 (0x2F)
JDK 1.2 = 46 (0x2E)
JDK 1.1 = 45 (0x2D)

```
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

}
```

Sobre el "Magic Number"

We used to go to lunch at a place called St Michael's Alley. According to local legend, in the deep dark past, the Grateful Dead used to perform there before they made it big. It was a pretty funky place that was definitely a Grateful Dead Kinda Place. When Jerry died, they even put up a little Buddhist-esque shrine. When we used to go there, we referred to the place as Cafe Dead. Somewhere along the line, it was noticed that this was a HEX number. I was re-vamping some file format code and needed a couple of magic numbers: one for the persistent object file, and one for classes. I used CAFEDEAD for the object file format, and in grepping for 4 character hex words that fit after "CAFE" (it seemed to be a good theme) I hit on BABE and decided to use it. At that time, it didn't seem terribly important or destined to go anywhere but the trash can of history. So CAFEBABE became the class file format, and CAFEDEAD was the persistent object format. But the persistent object facility went away, and along with it went the use of CAFEDEAD – it was eventually replaced by RMI.

James Gosling