1.2 Introducción al lenguaje Java

1. Historia

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para que los desarrolladores de aplicaciones escribieran el programa una vez y lo ejecutaran en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

El lenguaje de programación Java fue originalmente desarrollado por James Gosling, de Sun Microsystems (constituida en 1982 y posteriormente adquirida el 27 de enero de 2010 por la compañía Oracle). Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos.

El lenguaje Java se creó con cinco objetivos principales:

1. Debería usar el paradigma de la programación orientada a objetos.
2. Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
3. Debería incluir por defecto soporte para trabajo en red.
4. Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
5. Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

2. Bytecode, JVM, JRE, JDK

En el mundo de la programación siempre se ha hablado de lenguajes compilados y de lenguajes interpretados. El resultado del proceso de compilación (en realidad de compilación y enlazado) es un **archivo ejecutable**. Un archivo ejecutable es un programa que se puede lanzar directamente en el sistema operativo; en el caso de Windows o Linux simplemente con hacer doble clic sobre el archivo, se ejecutan sus instrucciones. La ventaja es que los programas ejecutables no necesitan compilarse de nuevo, son programas terminados. El problema es que los sistemas operativos utilizan diferentes tipos de archivos ejecutables: es decir, un archivo ejecutable en Linux no sería compatible con Windows.

En Java el código no se traduce a código ejecutable. En Java el proceso se conoce como precompilación y sirve para producir un archivo (de extensión **class**) que contiene código que no es directamente ejecutable (no es código Java). Es un código intermedio llamado **bytecode**. Al no ser ejecutable, el archivo class no puede ejecutarse directamente con un doble clic en el sistema. El bytecode tiene que ser interpretado (es decir, traducido línea a línea) por una aplicación conocida como la máquina virtual de Java (**JVM**).

JRE es el Java Runtime Environment o, en español, el Entorno de Ejecución de Java. Contiene a la JVM y otras herramientas que permiten la ejecución de las aplicaciones Java. La gran ventaja es que el entorno de ejecución de Java se fabrica para todas las plataformas; lo que significa que un archivo class se puede ejecutar en cualquier ordenador o máquina que incorpore el JRE. Sólo hay una pega, si programamos utilizando por ejemplo la versión 10 de Java, el ordenador en el que queramos ejecutar el programa deberá incorporar el JRE al menos de la versión 10.

A la forma de producir código final de Java se la llama JIT (**Just In Time**,justo en el momento) ya que el código ejecutable se produce sólo en el instante de ejecución del programa. Es decir, no hay en ningún momento código ejecutable.

JRE no posee compiladores ni herramientas para desarrollar las aplicaciones Java, solo posee las herramientas para ejecutarlas. **JDK** es el Java Development Kit o, en español, Herramientas de Desarrollo de Java. Sirve para construir programas usando el lenguaje de programación Java. Trae herramientas útiles como el compilador (javac), el debugger, herramientas de evaluación de rendimiento de aplicaciones, etc. Una instalación de JDK ya contiene un JRE dentro de las carpetas.

Para programar en Java, el primer paso que tiene que realizar el alumno es instalarse el JDK de la última versión de Java. La descarga la puede efectuar desde la página web de Oracle https://www.oracle.com/technetwork/java/javase/downloads/index.html en la pestaña *Downloads*. Buscar la última versión y descargar el JDK del Sistema Operativo con el que el alumno va a trabajar. Pero si al instalar el JDK ya existe una instalación del JRE en el ordenador de una versión anterior, el JDK no actualizará el JRE a la última versión. En este caso, hay que desinstalar primero el JRE y ya después instalar el JDK de la última versión.

3. Entornos de desarrollo integrado (IDE): Eclipse

El código en Java se puede escribir en cualquier editor de texto, y para compilar el código en bytecodes, sólo hace falta descargar la versión del JDK deseada. Sin embargo, la escritura y compilación de programas hecha de esta forma es un poco incómoda. Por ello numerosas empresas fabrican sus propios entornos de edición, algunos incluyen el compilador y otras utilizan el propio JDK de Java.

Un **IDE (*integrated development environment*)** es un entorno de programación que consiste básicamente en un editor de código, un compilador y un depurador.

Algunas ventajas que ofrecen son:

- Facilidades para escribir código: coloreado de las palabras clave, autocorrección al escribir, abreviaturas,...
- Facilidades de depuración, para probar el programa.
- Facilidad de configuración del sistema.
- Facilidades para organizar los archivos de código.
- Facilidad para exportar e importar proyectos.

Algunos IDEs para programar en Java son *Eclipse*, *Netbeans* e *IntelliJ IDEA*.

En esta asignatura utilizaremos como IDE el Eclipse, que el alumno puede descargar de la página web https://www.eclipse.org/downloads/eclipse-packages/. Buscar la versión más reciente del ***Eclipse IDE for Java Developers***.

4. Sentencias

Una sentencia es la unidad mínima de ejecución de un programa. Un programa se compone de conjunto de sentencias que acaban resolviendo un problema. Al final de cada una de las sentencias encontraremos un punto y coma (`;`).

Veamos algunos ejemplos de sentencias en java:

- Sentencias de declaración: `int x;`
- Invocaciones o llamadas a métodos de tipo void: `System.out.println("Bienvenidos a Programación");`
- Sentencias de control de flujo: alteran el fujo de ejecución para tomar decisiones o repetir sentencias.

5. Expresiones

Una expresión es una combinación de operadores y operandos que se evalúa generándose un único resultado de un tipo determinado.

La diferencia entre las sentencias y los operadores es que las expresiones devuelven un valor y las sentencias no devuelven nada.

6. Bloques

Un bloque es un conjunto de sentencias las cuales están delimitadas por llaves:

```
{
    sentencias
}
```