ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2019-2020

# GUION DE LA PRÁCTICA 0

## OBJETIVOS:

- **Trabajo con *benchmarks***
- **Establecer el entorno de trabajo.**
- **Operaciones con matrices**

## 1. BENCHMARKING

### 1.1. CONCEPTOS BÁSICOS BENCHMARKING

Repasaremos algunos conceptos básicos de cómo realizar mediciones de rendimiento.
Un ***benchmark*** o *banco de pruebas* es un conjunto de experimentos que se realiza para evaluar el funcionamiento de un sistema o una parte del mismo. Un ejemplo típico sería la ejecución de un *benchmark* en varios ordenadores para conocer cuál es más rápido o potente.
Para realizar una evaluación de un sistema podemos distinguir los siguientes pasos:

- **Analizar los factores que pueden influir en su rendimiento**. Los factores que tienen influencia en la velocidad de un ordenador serían: la memoria RAM, el sistema operativo, el disco duro, la CPU, etc.

- **Selección de métricas de rendimiento**. Elegir la magnitud que se desea medir. Puede ser una magnitud **menor es mejor** como "el tiempo que tarda en ejecutarse un programa" o una magnitud **mayor es mejor** como el "número de operaciones por segundo" de una CPU. También es habitual ejecutar varios *benchmark* y ponderar las mediciones a un valor en puntos.

- **Realizar experimentos de rendimiento**. Normalmente se realizan diferentes configuraciones con todos los factores constantes salvo uno que varía. Por ejemplo, un ordenador con Windows 10, 16 GB de RAM, disco duro SSD de 128 MB se probaría con diferentes CPUs.

En esta asignatura vamos a estudiar la complejidad temporal de diferentes programas y algoritmos, analizaremos el tiempo de ejecución de un programa frente al tamaño del problema.
En esta práctica veremos varios factores que dificultan la medición de tiempos de ejecución de un programa. El principal objetivo es introducir una serie de pautas que se deberán seguir siempre que se realicen mediciones de rendimiento.
**En cada una de las tareas que componen esta práctica se marcarán de este color azul los resultados y preguntas que tienes que contestar y que debes anotar en un documento a parte para posteriormente entregarlo.**

### 1.2. ACTIVIDAD 1: POTENCIA DE LAS CPUs

En esta actividad vamos a ejecutar el mismo programa en varios ordenadores para analizar el tiempo que tarda en ejecutarse. De todos los factores de un ordenador que influyen en el tiempo de ejecución de un programa nos fijaremos en uno de los más importantes: su **CPU**.

## *Tarea 1*

1. Busca el modelo del procesador de tu ordenador de prácticas. Para ello, lo más sencillo es abrir los detalles del sistema que nos proporciona Windows (`Este equipo > Propiedades`, o el atajo de teclado: `Win + Pausa`). **Anota el modelo de procesador y la memoria de tu sistema.**

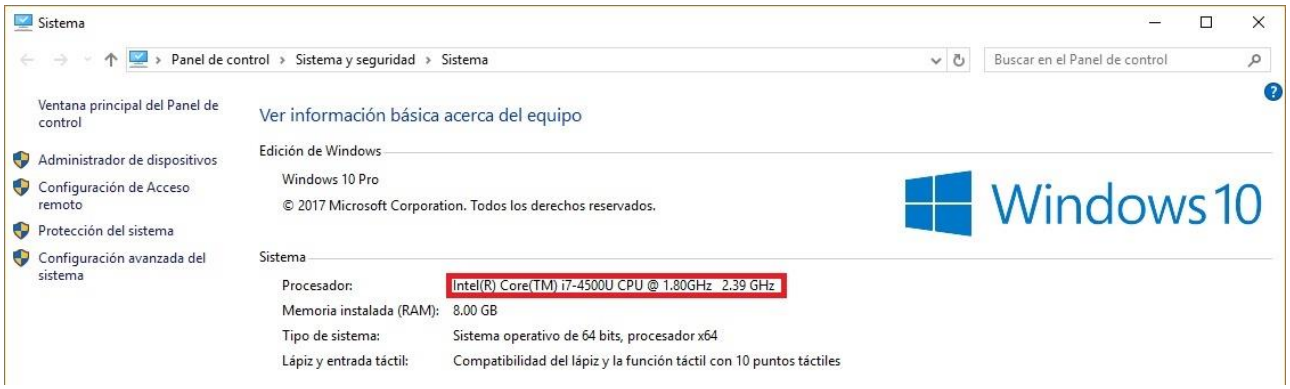


**Figura 1. Información del Sistema**

2. Busca ese modelo de procesador en la página User Benchmark (http://cpu.userbenchmark.com/)

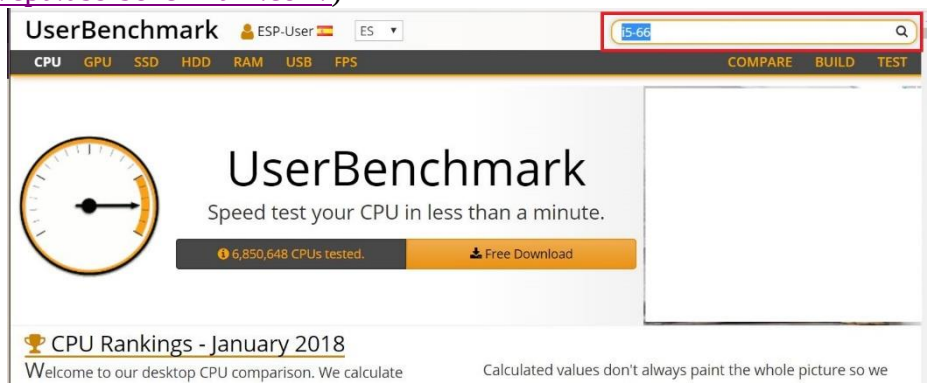


**Figura 2. UserBenchmark: Buscador de CPUs**

3. Busca y **anota el número máximo de operaciones enteras por unidad de tiempo** (Single Core Mixed Speed 1-Core) que realiza el modelo de procesador.

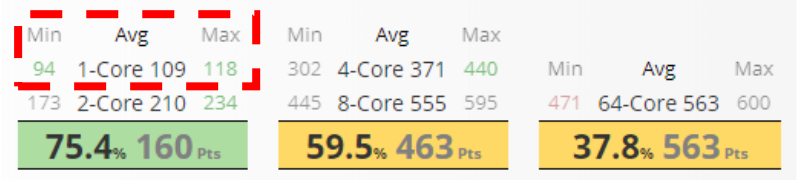


**Figura 3. UserBenchmark: Detalles de una CPU**

4. Compila y ejecuta (ver los apartados 4 y 5 de esta práctica) el programa `Benchmarking1`. **Anota el tiempo que ha tardado en finalizar**.

5. Calcula el **número aproximado de operaciones enteras que necesitó el programa**. Para ello multiplicaremos el tiempo de ejecución por el valor *1-Core Max* para ese procesador.

## *Tarea 2*

Con los datos que se proporcionan (tiempos de ejecución del programa en diferentes máquinas y modelos de CPU), realiza las siguientes tareas:

1. Completa la siguiente tabla con los tiempos de ejecución y *1-Core* de cada CPU. Calcula también el número de operaciones enteras. **Nota Importante**: Aunque ese número de operaciones resultante es teóricamente el mismo para todas las configuraciones, en la práctica habrá ciertas diferencias. **Anota tus resultados igualmente**.

2. **Amplía la tabla con los datos de otros ordenadores a los que tengas acceso** (por ejemplo, el ordenador que tienes en el laboratorio y tu propio ordenador).

| *#* | *CPU* | *milisegundos* | *1-Core (max)* | *Operaciones (aprox.)* |
|---|---|---|---|---|
| **1.** | i7-4790 | 216 | | |
| **2.** | i5-7600 | 205 | | |
| **3.** | i3-3220 | 267 | | |
| **4.** | | | | |
| **5.** | | | | |
| **6.** | | | | |

Tabla 1. Comparativa de CPUs

***Conclusión***

Viendo los resultados en milisegundos, **¿crees que podrías mezclar valores de diferentes CPUs en un mismo estudio analítico de los tiempos de ejecución de un algoritmo?**

## 1.3. ACTIVIDAD 2: INFLUENCIA DEL SISTEMA OPERATIVO

Los sistemas operativos modernos incluyen multitud de características que dificultan la medición de programas. En esta actividad analizaremos la influencia de las políticas de administración de energía del procesador y la ejecución simultánea de varios procesos sobre la misma CPU. Aunque ambos factores están estrechamente relacionados con las características del procesador, gran parte de la responsabilidad de gestionar estos recursos recae en el **sistema operativo** a través de los planes de energía y el planificador de procesos.

***Tarea 1***

1. Abre el Administrador de tareas y sitúate en la pestaña CPU.

2. Abre la configuración de energía de Windows: `Panel de control > Hardware y sonido > Opciones de energía`

3. Cambia entre los Planes: *Alto rendimiento*, *Equilibrado* y *Economizador*. Observa como varía la frecuencia de CPU.

Para evitar esto se podrían deshabilitar los servicios de administración de energía de Windows. Sin embargo, serían necesarios permisos de administrador. Dado que ni los profesores ni los alumnos disponemos de tales privilegios en las salas de ordenadores de la escuela, hemos grabado dos vídeos en los que se muestra la variación de frecuencia de CPU en entorno Windows. El primero muestra el funcionamiento por defecto de Windows (**video1.mp4**), mientras que en el segundo se han deshabilitado los servicios de administración de energía (**video2.mp4**).

***Tarea 2***
Realiza esta tarea con el programa `Benchmarking1` de la actividad anterior.

- Ejecución secuencial:

    1. Ejecuta el programa múltiples veces con el script `run.cmd`. No realices ninguna otra actividad que pueda consumir muchos recursos (uso del navegador, compiladores, etc.). Observa cómo el tiempo de ejecución varía ligeramente de una ejecución a otra.

    2. Realiza las mismas pruebas con diferentes planes de energía.

- Ejecución paralela:

    1. Ejecuta el programa **cpuburn.exe**. Este programa consume el 100% de la CPU por lo que es posible que el ordenador te responda más despacio (disponible en https://patrickmn.com/projects/cpuburn/).

    2. Ejecuta el programa `Benchmarking1` múltiples veces (`run.cmd`). Fíjate si los tiempos de ejecución son similares a los de la ejecución secuencial.

    3. Cuando termines asegúrate de detener el *cpuburn.exe*.

***Conclusiones***
Contesta a las siguientes preguntas:

1. **¿Qué plan de energía crees que es el más adecuado para realizar mediciones?**
2. Si tuvieses que realizar la medición de un experimento muy largo, **¿podrías utilizar el ordenador para por ejemplo ver un vídeo de YouTube?**
3. **¿Crees conveniente realizar varias mediciones simultáneamente en el mismo ordenador?**

## 2. MONTAJE DE JAVA EN DISCO DURO

El ordenador puede tener ya JDK (Kit de Desarrollo en Java) instalado (Nota: no confundir con JRE que sólo es un entorno de ejecución). En ese caso debe mirar en el disco duro el lugar o directorio donde está desplegado.

En el caso de que el ordenador no tenga el JDK, debe buscarlo en la red (p.ej. ponga “JDK” en google) y a partir de ahí descargar en su ordenador el fichero ejecutable JDK y posteriormente instalarlo (ejecutarlo) en el ordenador y así ya tendrá el entorno JAVA disponible.

## 3. ENTORNOS INTEGRADOS DE DESARROLLO (IDE)

Además de en forma básica (directamente con el JDK), se puede trabajar en JAVA con un IDE de los distintos que existen, que permiten integrar múltiples características dentro de un entorno gráfico. En los laboratorios de prácticas disponemos del **entorno Eclipse** que está muy extendido tanto en entornos académicos como en profesionales.

# 4. PROGRAMAR CON JAVA EN LÍNEA DE COMANDOS

La **ejecución básica de programas en Java** se hace desde la **ventana DOS**. Para lograr dicha ventana tiene que ejecutar el comando:

```
CMD
```

Cuando queramos cambiar de unidad sólo tenemos que poner:

```
x:                  // irá a la unidad x:
```

Cuando estemos en un directorio y nos queramos colocar en un hijo haremos:

```
CD hijo
```

Para subir de un directorio al padre:

```
CD ..
```

Una forma de configurar el entorno es crear con un editor un **fichero de lotes (.bat)** con el PATH (apuntando al directorio bin de JDK) y el CLASSPATH (apuntando a los directorios de clases Java donde vayamos a trabajar) adecuados al ordenador concreto. Ese fichero de lotes se debe ejecutar cada vez que abramos una ventana DOS.

El comando **SET PATH** permite saber los PATH activos en ese momento.
El comando **SET CLASSPATH** permite saber los CLASSPATH activos en ese momento**.**

Si se cierra la ventana DOS desaparecen las rutas establecidas, por lo que cuando abramos otra nueva ventana DOS existe la necesidad de volver a establecer la ruta.

Cuando creamos un programa o clase JAVA lo *normal* es que forme parte de un paquete (conjunto de clases JAVA con finalidad común). La primera sentencia válida de la clase establece precisamente a qué paquete pertenece esa clase. Además, el directorio en el que se almacena esa clase debe coincidir exactamente con el nombre del paquete.

Para editar vale cualquier **editor de programas** (vale hasta el Bloc de Notas de Windows, en cuyo caso hay que grabar el programa con la **opción de "Todos los Archivos"** y la extensión .java, para que no se añada automáticamente la extensión **.txt**). En los laboratorios del centro también se encuentran instalados otros editores más potentes que proporcionan ventajas como el coloreado de sintaxis o completado de código. Algunos de los disponibles actualmente son: Notepad++, Visual Studio Code y Sublime text.

# 5. PROGRAMAR CON JAVA EN EL ENTORNO DE DESARROLLO ECLIPSE

Uno de los entornos más utilizados en la comunidad de desarrolladores Java es Eclipse. Este entorno de desarrollo integrado permite realizar desde una interfaz común todas las tareas relacionadas con el desarrollo: codificación, análisis de código y eliminación de errores, ejecución y depuración.

En este entorno la unidad mínima de desarrollo es el proyecto, así que debemos crear un nuevo proyecto para empezar a trabajar con el código. Cuando disponemos de código fuente de clases creadas previamente, sólo hace falta, *arrastrar y soltar* la carpeta correspondiente al paquete de esas clases Java a la carpeta *src* dentro del proyecto.

Una de las grandes ventajas del entorno Eclipse es que realiza una compilación automática al modificar y al grabar el código e informa al usuario de posibles errores mediante marcas, tanto subrayando el código implicado, como estableciendo un icono al margen, los cuales proporcionan información extra sobre el error o *warning* si nos situamos encima con el cursor.

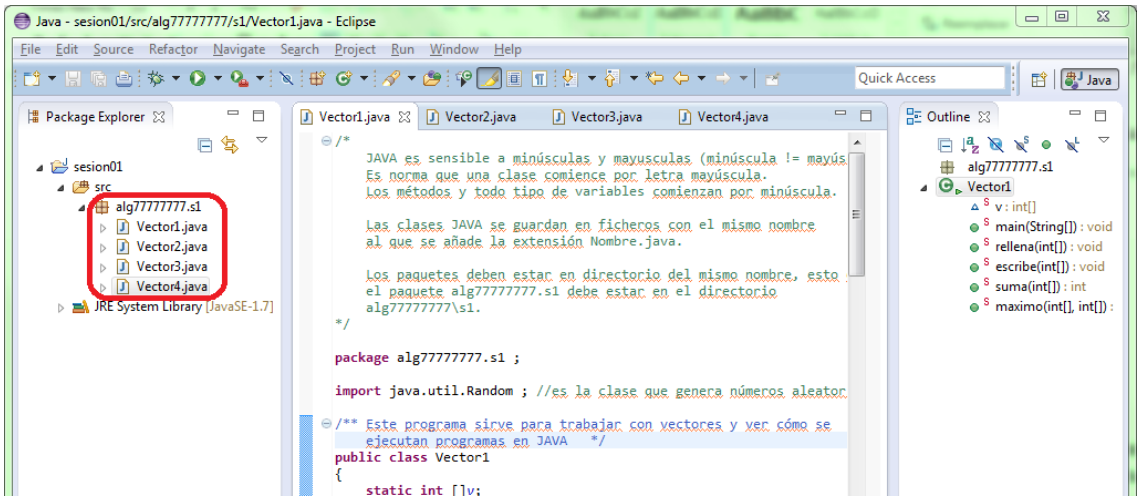


**Figura 4. Espacio de trabajo de Eclipse**

Para ejecutar y depurar debemos utilizar los iconos correspondientes en la barra de herramientas o bien el menú contextual sobre el proyecto y utilizar la opción "**Run as…**". Para realizar esta operación se deben guardar todos los cambios realizados en los ficheros del proyecto y el entorno se encargará de realizar las operaciones necesarias para ejecutarlo. Existen varios modos de ejecución, el que nos interesa es "**Java Application**" que ejecuta las clases Java sin ningún tipo de contenedor ni servidor. Además, Eclipse permite establecer los argumentos que acompañan la llamada a la clase principal para ejecutarla. Esto lo podemos hacer a través de la opción "**Run configurations…**" a la que podemos acceder bien desde el botón de ejecución de la barra de herramientas o bien desde el menú contextual. A partir de aquí deberemos seleccionar en el menú de la izquierda la clase relacionada y en las pestañas de la derecha la denominada "**Arguments**". Aquí podemos introducir los valores para los argumentos en el campo "Program arguments" separados por espacios.

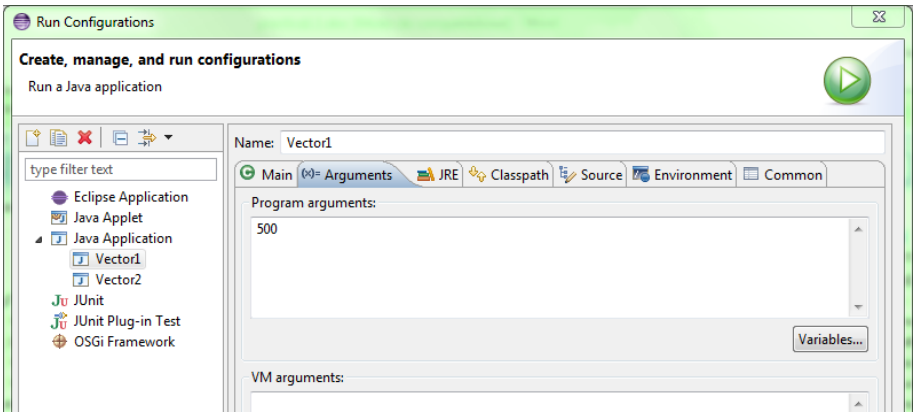


**Figura 5. Ejecución de un programa con argumentos**

# 6. OPERACIONES CON MATRICES

Durante las prácticas para esta asignatura trabajaremos muchas veces con vectores o matrices multidimensionales de enteros u otros tipos de datos. Esto es debido a que es una buena forma de representar propiedades que definen los elementos con los que vamos a trabajar y en esta asignatura el número de elementos será considerablemente grande, aunque para poner a prueba nuestro algoritmo partamos de ejemplos limitados.

En este apartado debes crear el código Java de una clase MatrizOperaciones que implemente una matriz cuadrada de dos dimensiones cuyo tamaño introduciremos como parámetro en el momento de crearla y varias operaciones asociadas que vamos a enumerar a continuación:

1. `MatrizOperaciones(int n)`. Crea una matriz de tamaño nxn y la rellena con valores aleatorios, estos valores aleatorios deben de ser parametrizables entre un máximo y un mínimo.
2. `MatrizOperaciones(String nomFich)`. Crea una matriz a partir de los datos del fichero. El formato del fichero será: Primera línea, un entero con tamaño de la matriz (n), resto de las líneas, la fila correspondiente en la que cada valor estará separado por un tabulador del siguiente.
3. `getTam()`. Devuelve el tamaño de la matriz.
4. `escribir()`. Muestra el contenido de la matriz por pantalla.
5. `sumarDiagonal1()`. Calcula de forma iterativa la suma de la diagonal. Forma 1: recorrer toda la matriz, pero sólo sumando los elementos de la diagonal
6. `sumarDiagonal2()`. Calcula de forma iterativa la suma de la diagonal. Forma 2: recorrer los elementos de la diagonal sumándolos
7. `recorrerCamino(int i, int j)`. En una matriz cuyos valores varían entre 1 y 4 vamos a trazar un "camino" partienendo de la posición (i,j) que pasamos como parámetro y utilizando los valores de la matriz como códigos de dirección: 1 - arriba, 2 - derecha, 3 - abajo, 4 – izquierda. Vamos a utilizar para marcar el camino el código -1. El proceso finalizará cuando el camino salga de los límites de la matriz o bien alcance una casilla ya recorrida.

# 7. TRABAJO PEDIDO

**Haz un documento y en él contesta a las diferentes preguntas de la sección 1 Benchmarking. Recuerda que debes completar la Tabla 1 y contestar a las preguntas que se plantean en las tareas y en las conclusiones.**

**Guarda el código creado en el apartado 6 para la próxima sesión de prácticas.**

*La entrega de esta práctica se realizará junto a las siguientes, en la tarea creada a tal efecto en el campus virtual. Se entregará un documento con las tablas y las respuestas a las preguntas.*