


CICLO: D.A.M. CURSO: SEGUNDO	MÓDULO: PROGRAMACIÓN DE SERVICIOS Y PROCESOS	
1ª EVALUACIÓN: PRUEBA PRÁCTICA RESULTADO DE APRENDIZAJE 2		

### Instrucciones de Entrega

#### 1. Archivos de Código Fuente:

- Entregar los archivos de código fuente en formato **.java**.
- No comprimir los archivos.
- No incluir las estructuras de carpetas; solo los archivos **.java**.

#### 2. Memoria Descriptiva:

- Crear un documento en formato PDF para cada ejercicio.
- El documento debe describir detalladamente el desarrollo del ejercicio (sin caer en el exceso).
- Incluir los siguientes apartados en cada memoria:
  - Introducción: Breve descripción del objetivo del ejercicio.
  - Desarrollo: Explicación detallada del proceso de desarrollo, incluyendo decisiones de diseño y justificaciones.
  - Implementación: Descripción de la implementación del código, destacando las partes más importantes.
  - Pruebas: Explicación de las pruebas realizadas para verificar el correcto funcionamiento del código.
  - Conclusión: Resumen de los resultados obtenidos y posibles mejoras futuras.

#### 3. Formato y Entrega:

- Asegurarse de que cada PDF esté bien estructurado y sea claro.
- Nombrar los archivos PDF de manera coherente, por ejemplo: Ejercicio2\_Memoria.pdf, Ejercicio3\_Memoria.pdf, etc.

**Ejercicio 1** (CE b). (1 punto)


- a) (0,5 puntos) Escribe la clase **TareaCalculo** que extienda **Thread**. En el método **run()** se generará un número aleatorio **n** con un valor entre 100 y 1.000 cada 10 segundos de manera indefinida. El hilo tiene que ir mostrando el valor acumulado (suma) de los números generados en la consola.

Escribe la clase **Ejercicio\_1** que lance 30 hilos **TareaCalculo** cada uno con su nombre.

- b) (0,25 puntos) Modifica el código para que **TareaCalculo** implemente **Runnable**.
- c) (0,25 puntos) Modifica la última versión de **TareaCalculo** para que incluya un método que permita detener el hilo usando una variable booleana cuando la suma alcance 1.000.000.

**Ejercicio 2** (CE c) (1 punto)

- a) (0,25 puntos) Escribe una clase que genere 200 identificadores alfanuméricos diferentes de 6 caracteres cada uno.
- b) (0,25 puntos) Escribe una clase que genere un fichero en formato **csv** con 50.000 líneas de pares de elementos. El primer elemento debe ser uno de los identificadores alfanuméricos del fichero anteriormente generado. El segundo elemento debe ser un número entero entre 0 y 20.000.
- c) (0,25 puntos) Escribe una clase que genere 100 archivos de 100.000 líneas cada uno en formato **csv** utilizando 100 hilos de la clase generadora de pares.
- d) (0,25 puntos) Modifica la clase del apartado b) para que genere dos ficheros en lugar de uno, y la clase del apartado c) para que genere 50 hilos en lugar de 100. Compara los tiempos de ejecución de cada clase.

CICLO: D.A.M. CURSO: SEGUNDO	MÓDULO: PROGRAMACIÓN DE SERVICIOS Y PROCESOS	
1ª EVALUACIÓN: PRUEBA PRÁCTICA RESULTADO DE APRENDIZAJE 2		

### **Ejercicio 3** (CE e) (1,5 puntos)

Escribe la clase **CuentaVocales** que utilice 5 hilos para contar el número de vocales que hay en un determinado texto. Cada hilo se encargará de contar una vocal diferente, actualizando todos los hilos la misma variable común que representa el número de vocales totales.

Rúbrica:

Criterio	Descripción	Puntos
Estructura y Diseño	- Definición de hilos	0.5
	- Variable compartida	
	- Asignación de tareas	
	- Mecanismo de sincronización	
Corrección y Eficiencia	- Cuenta correctamente	0.3
	- Evita condiciones de carrera	
	- Eficiencia	
Legibilidad y Mantenibilidad	- Comentarios	0.2
	- Estilo de codificación	
	- Modularidad	
Uso de Concurrencia	- Conocimiento de conceptos	0.5
	- Aplicación de conceptos	
Total		1.5

#### **Ejercicio 4** (CE f) (1,5 puntos)

- a) (0,5 puntos) Implementa una clase **Lenguaje** que escriba en un fichero indicado por el usuario conjuntos de letras generadas de forma aleatoria (sin sentido real). Cada conjunto de letras se escribirá en una línea distinta. El número de conjuntos de letras a generar por el proceso será dado también por el usuario en el momento de la ejecución. Por ejemplo la ejecución de este comando:


```
java Lenguaje 40 miFichero.txt
```

generará 40 palabras que se guardarán en el archivo *miFichero.txt*

- b) (1 punto) Escribe una clase llamada **Colaborar** que lance al menos 10 hilos de **Lenguaje**, haciendo que todas ellas colaboren en generar un gran fichero de palabras. Cada hilo generará un número creciente de palabras: 10, 20, 30, .... Cada hilo seguirá escribiendo su palabra en una línea independiente de las otras. Es decir, si lanzamos 10 hilos de **Lenguaje**, al final debemos tener un fichero con  $10+20+30+\dots+100 = 550$  líneas.

#### **Ejercicio 5** (CE g) (1 punto)

- a) (0,25 puntos) Escribe la clase **HiloPrioridades** con un constructor que permita establecer el nombre y la prioridad del hilo. El método **run()** de esta clase mostrará un mensaje al iniciar el hilo con su nombre. A continuación, en un bucle sin fin se comprobará la prioridad del hilo. Si tiene prioridad 1 se ejecutará el método **tarea1()**, si tiene prioridad 3 el método **tarea3()** y si tiene prioridad 5 el método **tarea5()**. Al final de estas comprobaciones añade una demora de 1 segundo.
- b) (0,5 puntos) Escribe los métodos **tarea1()**, **tarea2()** y **tarea3()** de **HiloPrioridades** haciendo que muestren un mensaje con el nombre del hilo y su prioridad, añadiendo “tarea lenta”, “tarea normal” y “tarea rápida” al final en cada caso.
- c) (0,25 puntos) Escribe la clase **Ejercicio\_6** que cree 3 hilos y los lance. Observa y documenta como las prioridades afectan a la ejecución de los hilos.

CICLO: D.A.M. CURSO: SEGUNDO	MÓDULO: PROGRAMACIÓN DE SERVICIOS Y PROCESOS	
1ª EVALUACIÓN: PRUEBA PRÁCTICA RESULTADO DE APRENDIZAJE 2		

### **Ejercicio 6** (CE d) (1,5 puntos)


Escribe la clase **EstadosHilo** en la que se muestren todos los posibles estados de un hilo utilizando los métodos **wait()**, **notify()**, **join()** y **sleep()**.

Criterio	Descripción	Puntos
Estructura y Diseño	- Métodos para cada estado	0.5
	- Uso correcto de los métodos	
	- Sincronización	
Corrección y Completitud	- Demostración de todos los estados	0.3
	- Explicación de los estados	
	- Manejo de excepciones	
Legibilidad y Mantenibilidad	- Comentarios	0.2
	- Estilo de codificación	
	- Modularidad	
Comprensión de Conceptos	- Estados de un hilo	0.5
	- Propósito de los métodos	
	- Mecanismos de sincronización	
Total		1.5

**Ejercicio 7** (CE h) (0,5 puntos)

Crear documentación detallada para las clases y métodos desarrollados en los ejercicios anteriores utilizando comentarios y herramientas de documentación como *Javadoc*.

- Añadir comentarios *Javadoc* para la clase y cada uno de sus métodos.
- Describir el propósito de la clase y de cada método.
- Incluir información sobre los parámetros de entrada y los valores de retorno.
- Proporcionar ejemplos de uso para cada método.

CICLO: D.A.M. CURSO: SEGUNDO	MÓDULO: PROGRAMACIÓN DE SERVICIOS Y PROCESOS	
1ª EVALUACIÓN: PRUEBA PRÁCTICA RESULTADO DE APRENDIZAJE 2		

## Rúbricas

Ejercicio 1 (CE b). (1 puntos)

- **(0.5 puntos)** clase `TareaCalculo`:
  - **0.25 puntos**: El método `run()` ejecuta correctamente su tarea.
  - **0.25 puntos**: La clase `Ejercicio_1` lanza los hilos.
- **(0.25 puntos)** Se modifican con éxito las clases `TareaCalculo` y `Ejercicio_1`
- **(0.25 puntos)** Se modifican con éxito las clases `TareaCalculo` y `Ejercicio_1`

Ejercicio 2 (CE c) (1 puntos)

- Escribe una clase que genere 200 identificadores alfanuméricos diferentes de 6 caracteres cada uno.
  - Escribe una clase que genere un fichero en formato `csv` con 50.000 líneas de pares de elementos. El primer elemento debe ser uno de los identificadores alfanuméricos del fichero anteriormente generado. El segundo elemento debe ser un número entero entre 0 y 20.000.
  - Escribe una clase que genere 100 archivos de 100.000 líneas cada uno en formato `csv` utilizando 100 hilos de la clase generadora de pares.
  - Modifica la clase del apartado b) para que genere dos ficheros en lugar de uno, y la clase del apartado c) para que genere 50 hilos en lugar de 100. Compara los tiempos de ejecución de cada clase.
- **(0.25 puntos)** Iniciar un subproceso que ejecute el comando `tar`:
    - **0.25 puntos**: El subproceso se inicia correctamente.
    - **0.25 puntos**: El comando `tar` se ejecuta correctamente.
  - **(0.5 puntos)** Enviar la lista de archivos a comprimir al subproceso:
    - **0.25 puntos**: La lista de archivos se envía correctamente al subproceso.
    - **0.25 puntos**: El subproceso recibe y procesa la lista de archivos correctamente.
  - **(0.5 puntos)** Leer la respuesta del subproceso y mostrarla en la consola:
    - **0.25 puntos**: El subproceso devuelve un mensaje de éxito o error.
    - **0.25 puntos**: El programa principal lee y muestra la respuesta del subproceso en la consola.


Ejercicio 3 (CE g) (1,5 puntos)

Criterio de Evaluación: Se han utilizado mecanismos para sincronizar y obtener el valor devuelto por los subprocesos iniciados.

- **(0.5 puntos)** Iniciar dos subprocesos para contar líneas y palabras:
  - **0.25 puntos**: El primer subproceso cuenta correctamente el número de líneas en un archivo de texto.
  - **0.25 puntos**: El segundo subproceso cuenta correctamente el número de palabras en otro archivo de texto.
- **(0.5 puntos)** Sincronizar la obtención de los resultados:
  - **0.25 puntos**: El programa principal espera correctamente a que ambos subprocesos terminen.
  - **0.25 puntos**: El programa principal obtiene correctamente los resultados de ambos subprocesos.

- **(0.5 puntos)** Mostrar el total combinado de líneas y palabras en la consola:
  - **0.25 puntos:** El programa principal combina correctamente los resultados de ambos subprocesos.
  - **0.25 puntos:** El programa principal muestra correctamente el total combinado de líneas y palabras en la consola.



CICLO: D.A.M. CURSO: SEGUNDO	MÓDULO: PROGRAMACIÓN DE SERVICIOS Y PROCESOS	
1ª EVALUACIÓN: PRUEBA PRÁCTICA RESULTADO DE APRENDIZAJE 2		

#### Ejercicio 4 (CE h) (2 puntos)

Criterio de Evaluación: Se han desarrollado aplicaciones que gestionen y utilicen procesos para la ejecución de varias tareas en paralelo.

- **(0.5 puntos)** Permitir al usuario seleccionar varios archivos para procesar:
  - **0.25 puntos:** La aplicación permite al usuario seleccionar varios archivos.
  - **0.25 puntos:** La selección de archivos se maneja correctamente.
- **(0.5 puntos)** Iniciar múltiples procesos en paralelo para comprimir cada archivo:
  - **0.25 puntos:** La aplicación inicia correctamente múltiples procesos en paralelo.
  - **0.25 puntos:** Cada proceso comprime un archivo de forma independiente.
- **(0.5 puntos)** Esperar a que todos los procesos terminen y obtener sus resultados:
  - **0.25 puntos:** La aplicación espera correctamente a que todos los procesos terminen.
  - **0.25 puntos:** La aplicación obtiene correctamente los resultados de todos los procesos.
- **(0.5 puntos)** Mostrar el estado final de cada tarea en la consola:
  - **0.25 puntos:** La aplicación muestra correctamente el estado final de cada tarea (éxito o error).
  - **0.25 puntos:** La información mostrada en la consola es clara y precisa.

#### Ejercicio 5 (CE i) (1 punto)

Criterio de Evaluación: Se han depurado y documentado las aplicaciones desarrolladas.

- **(0.25 puntos)** Añadir comentarios *Javadoc* para la clase y cada uno de sus métodos:
  - **0.125 puntos:** Comentarios *Javadoc* añadidos para todas las clases.
  - **0.125 puntos:** Comentarios *Javadoc* añadidos para todos los métodos.
- **(0.25 puntos)** Describir el propósito de la clase y de cada método:
  - **0.125 puntos:** Propósito de cada clase descrito claramente.
  - **0.125 puntos:** Propósito de cada método descrito claramente.
- **(0.25 puntos)** Incluir información sobre los parámetros de entrada y los valores de retorno:
  - **0.125 puntos:** Información sobre los parámetros de entrada incluida.
  - **0.125 puntos:** Información sobre los valores de retorno incluida.
- **(0.25 puntos)** Proporcionar ejemplos de uso para cada método:
  - **0.125 puntos:** Ejemplos de uso proporcionados para todos los métodos.
  - **0.125 puntos:** Ejemplos de uso claros y funcionales.