EJERCICIO 1

```
import java.io.File;
import java.io.IOException;

public class EjecutarBat {

    public static void main(String[] args) {
        // Crear un archivo .bat con los comandos
        File salida = new File("salida.log");   // Archivo de salida
        File errores = new File("errores.log"); // Archivo de errores

        // Crear una instancia de ProcessBuilder para ejecutar el archivo .bat
        ProcessBuilder processBuilder = new ProcessBuilder("cmd.exe", "/c",
"comandos.bat");

        // Redirigir la entrada desde el archivo .bat, la salida y los errores
a los archivos correspondientes
        processBuilder.redirectOutput(salida);   // Redirigir la salida
estándar al archivo de salida
        processBuilder.redirectError(errores);   // Redirigir los errores al
archivo de errores

        try {
            // Iniciar el proceso
            Process process = processBuilder.start();

            // Esperar a que el proceso termine
            int exitCode = process.waitFor();
            System.out.println("El proceso ha terminado con el código: " +
exitCode);
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

EJERCICIO 2

```
public class HilosEjemplo {
    public static void main(String[] args) {
// Comprobar si se ha pasado el parámetro del tiempo de espera
        if (args.length < 1) {
            System.out.println("Debe introducir el tiempo de espera máximo
como argumento.");
                    System.exit(1);
        }
// Convertir el argumento a un entero que representa los segundos de espera
        int tiempoMaximo = Integer.parseInt(args[0]);
// Crear una instancia del hilo secundario
        HiloSecundario hiloSecundario = new HiloSecundario();
// Obtener el tiempo inicial
        long inicio = System.currentTimeMillis();
// Iniciar el hilo secundario
        Thread hilo = new Thread(hiloSecundario);
        hilo.start();
// Hilo principal muestra que está esperando
        System.out.println("Hilo: " + Thread.currentThread().getName() + ".
Tiempo de espera: " + tiempoMaximo + "s");
// Hilo principal en espera, mostrando cada segundo que espera
        try {
            for (int i = 0; i < tiempoMaximo; i++) {
                System.out.println("Hilo: " + Thread.currentThread().getName()
+ ". Todavía esperando...");

                Thread.sleep(1000); // Esperar 1 segundo
```

```
                if (!hilo.isAlive()) { // Si el hilo secundario ha terminado,
salir del bucle
                    break;
                }
            }

            if(hilo.isAlive()) {
                System.out.println("Hilo: " + Thread.currentThread().getName()
+ ". Tiempo máximo superado. Interrumpiendo hilo secundario...");
                hilo.interrupt();
            }
// Esperar a que el hilo secundario termine
            hilo.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
// Mostrar el tiempo total de ejecución
        long fin = System.currentTimeMillis();
        System.out.println("Hilo: " + Thread.currentThread().getName() + ".
*** Finalizado. Tiempo de ejecución: " + (fin - inicio) / 1000 + "s. ***");
    }
}
// Clase que representa el hilo secundario
class HiloSecundario implements Runnable {
    private final String[] mensajes = {"Programas", "Procesos", "Servicios",
"Hilos"};
    private int mensajeActual = 0;

    @Override
    public void run() {
        while (mensajeActual < mensajes.length) {
            System.out.println("Hilo: " + Thread.currentThread().getName() +
". " + mensajes[mensajeActual]);
            try {
                Thread.sleep(4000); // Esperar 4 segundos entre los mensajes
                mensajeActual++; // Incrementar el índice del mensaje
            } catch (InterruptedException e) {
                // Si el hilo es interrumpido, mostrar los mensajes restantes
sin esperar
                System.out.println("Hilo: " + Thread.currentThread().getName()
+ ". Interrumpido, mostrando mensajes restantes sin esperas.");
                break;
            }
        }
        mensajeActual++;

        // Mostrar los mensajes restantes sin esperas
        while (mensajeActual < mensajes.length) {
            System.out.println("Hilo: " + Thread.currentThread().getName() +
". " + mensajes[mensajeActual]);
            mensajeActual++;
        }

        System.out.println("Hilo: " + Thread.currentThread().getName() + ".
*** Finalizado ***");
    }
}
```