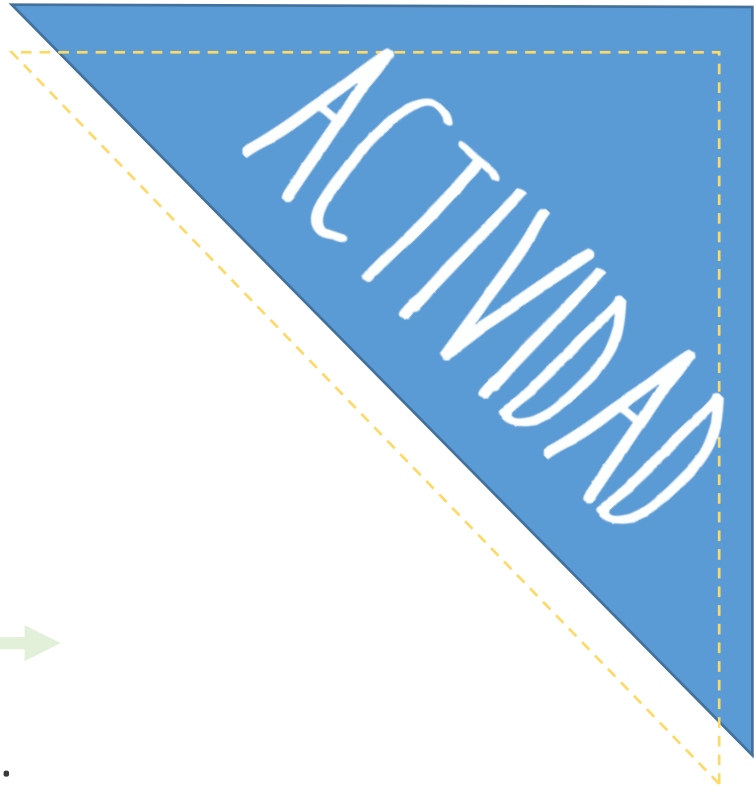

Dosier de Actividades del Tema 1:
Programación de procesos

Desarrollo de Aplicaciones Multiplataforma

Programación de Servicios y Procesos

Linkia FP

Ejercicio 1:

Explica las diferencias que existen entre los diferentes tipos de programación: concurrente, paralela, multiprogramación y distribuïda.

- **Programación concurrente**: Se basa en ejecutar múltiples tareas al mismo tiempo, pero no en paralelo y de forma no secuencial. En un programa concurrente las tareas puede continuar sin la necesidad que otras comiencen o finalicen. Cuando se ejecutan tareas de forma concurrente a estas se les asigna un x periodo de tiempo antes de cambiar de tarea, será en ese periodo en el cual se inicie, continúe, o se complete la tarea.
    - Por ejemplo, un programa que utiliza varios hilos para realizar tareas simultáneamente, como procesar solicitudes de usuarios en un servidor web.
- **Programación paralela**: La programación paralela es una técnica de programación en la que muchas instrucciones se ejecutan simultáneamente. Las tareas se realizarán de forma simultánea, comenzarán y finalizarán sin interrupciones.
    - Por ejemplo, dividir una tarea en partes y realizar esas partes simultáneamente en varios núcleos de un procesador para acelerar el proceso.
- **Multiprogramación**: Es una técnica de procesamiento que permite que varias aplicaciones se ejecuten de forma simultánea en una misma computadora, con la particularidad de que se utilizan los recursos de forma compartida y coordinada.
    - Por ejemplo, cuando hay varias aplicaciones abiertas al mismo tiempo, como un navegador web, un reproductor de música y una aplicación de procesamiento de texto. Permite alternar entre estas aplicaciones y realizar tareas en cada una de ellas sin necesidad de cerrar una antes de abrir la siguiente.
- **Programación distribuida**: Es un metódo en el que las tareas y los recursos de un programa se distribuyen en múltiples dispositivos o sistemas interconectados en una red.
    - Por ejemplo, la construcción de aplicaciones web, donde el servidor y los clientes se comunican a través de una red para intercambiar datos y realizar tareas.

Ejercicio 2:

Define los 5 estados por los que puede pasar un determinado proceso.

Un proceso puede pasar por los estados Nuevo, Preparado, En Ejecución, Bloqueado y Muerto.

- **Nuevo**: Este estado ocurre cuando se crea el objeto del proceso, pero aún no ha comenzado la ejecución.
    - Por ejemplo, cuando se inicia una aplicación, el proceso se encuentra en este estado antes de ejecutarse.
- **Preparado**: En este estado, el proceso está listo y espera una asignación del procesador para ser ejecutado.
    - Por ejemplo, si tienes varias aplicaciones abiertas en tu sistema operativo, todas las que están listas para ejecutarse pero aún no lo están, están en estado preparado.
- **En Ejecución**: El proceso se encuentra en este estado cuando tiene una asignación del procesador y está siendo ejecutado activamente.
    - Por ejemplo, cuando estamos jugando a un videojuego, el proceso aún se está ejecutando.
- **Bloqueado**: En este estado, el proceso está esperando un evento o condición que le permita volver a ejecutarse. Puede ser porque está esperando una operación de entrada/salida, está en espera de algún recurso compartido o está suspendido temporalmente.
    - Por ejemplo, cuando un programa espera a que se cargue una página web, queda en estado "Bloqueado" hasta que la página se haya descargado completamente.
- **Muerto**: El hilo ha finalizado su ejecución correctamente o debido a alguna excepción.
    - Por ejemplo, cuando se cierra una aplicación, su proceso entra en estado muerto si se cierra de manera normal, pero si se produce un error grave, también puede terminar en estado muerto.

Ejercicio 3:

Construir un proyecto Java que cumpla con las siguientes especificaciones:

- Existirá un programa padre llamado **Principal**, que será ejecutado al lanzar el proyecto.
- Creará un proceso hijo, que será la invocación a un programa Java diferenciado llamado **Sumador**.
- El proceso padre solicitará dos números desde la entrada estándar y los enviará a la entrada estándar del proceso Sumador.
- El proceso Sumador devolverá al proceso Principal la suma comprendida entre los dos números recibidos (ambos incluidos).
- El proceso padre leerá tantos números como el usuario decida introducir, hasta que detecte dos 0. Esta línea no deberá ser tratada por el proceso Sumador
- El proceso Principal muestra por pantalla lo que le envía el proceso Sumador.

Se debe incluir en este apartado el código de los programas y una captura del funcionamiento de la aplicación.

Principal.java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;

public class Principal {
    public static void main(String[] args) {

        try {
            while (true) {
                Process sumador = new ProcessBuilder("java", "-jar",
"M09_UF2_T1_Ejercicio3_Sumador.jar").start();

                BufferedReader sumadorIn = new BufferedReader(new
InputStreamReader(sumador.getInputStream()));
                PrintStream sumadorOut = new
PrintStream(sumador.getOutputStream(), true);

                int first = getUserNumber();

                int second = getUserNumber();

                if (first == 0 && second == 0) {break;}

                sumadorOut.println(first);
                sumadorOut.println(second);

                System.out.println("La suma es " + sumadorIn.readLine());
            }

            System.out.println("FIN");

        } catch (IOException e) {
            System.out.println("Error : " + e.getMessage());
        }
    }

    private static int getUserNumber() {
        BufferedReader userIn = new BufferedReader(new
InputStreamReader(System.in));

        while (true) {
            System.out.print("Escribe un número entero: ");
            try {
                return Integer.parseInt(userIn.readLine());
```

```
            } catch (NumberFormatException e) {
                System.out.println("No es un número válido. Introduce un número
entero.");
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }


}
```

Sumador.java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Sumador {
    public static void main(String[] args) {
        try {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
            int first = Integer.parseInt(reader.readLine());
            int second = Integer.parseInt(reader.readLine());

            int result = 0;

            for (int i = first; i <= second; i++) result += i;

            System.out.println(result);

            reader.close();
        } catch (IOException e) {
            System.out.println("Error : " + e.getMessage());
        }
    }
}
```

Ejemplo de ejecución del conjunto:

```
Escribe el primer número: 5
Escribe el segundo número: 8
La suma es 26
Escribe el primer número: 10
Escribe el segundo número: 15
La suma es 75
Escribe el primer número: 0
Escribe el segundo número: 0
FIN
```

Este es el output del programa:

```
C:\Users\Joel\.jdks\openjdk-21-1\bin\java.exe "-java
Escribe un número entero: 5
Escribe un número entero: 8
La suma es 26
Escribe un número entero: ajwoijdaj
No es un número válido. Introduce un número entero.
Escribe un número entero: 20
Escribe un número entero: 90
La suma es 3905
Escribe un número entero: 0
Escribe un número entero: 0
FIN

Process finished with exit code 0
```

Formato de entrega.

Un archivo comprimido (en formato zip o rar) con el siguiente contenido:

- Archivos .java y jar de los diferentes ejercicios.
- Un documento en formato pdf con las capturas del código y una prueba de funcionamiento de cada actividad.