Enunciado Práctica Evaluable 1
Suma paralela de números enteros

Objetivo

Que el alumno ponga en práctica los conocimientos teóricos y prácticos adquiridos en los Temas 1 y 2 de la asignatura de Programación Concurrente. La práctica se podrá realizar de forma individual o por parejas. Los alumnos que tengan la dispensa académica deberán realizar la práctica en las mismas condiciones (fecha y forma de entrega) que el resto de alumnos.

Fecha y forma de entrega

La fecha límite para la entrega de la práctica será el día indicado en la actividad correspondiente en el Aula Virtual, a través del cual se realizará la entrega de la práctica. Se deberá entregar un .zip cuyo contenido será el proyecto Eclipse en el que se ha realizado la práctica. El nombre del fichero .zip tiene que ser igual al identificador del alumno en la URJC (la parte antes de la @ del correo electrónico del alumno en la URJC). En caso de que la práctica haya sido realizada por dos alumnos, se incluirán ambos nombres separados por "-" y ambos alumnos deben entregar la práctica de manera independiente.

Además del código fuente, se deberá elaborar una memoria explicativa del mismo en formato PDF. El contenido de la memoria se detalla más adelante. La memoria deberá guardarse en la carpeta src/main/resources y tendrá el nombre "memoria.pdf".

Los alumnos que hayan cursado esta asignatura el curso anterior y que hubieran aprobado la práctica 1 en dicho curso no tendrán que volver a realizarla. Esta práctica será convalidada por defecto con la práctica 1 del curso anterior, asignando la nota que se obtuviera. No obstante, si el alumno desea realizar la práctica nuevamente, puede hacerla y entregarla en forma y fecha.

Enunciado

Se dispone de un programa que realiza la suma de un array de números enteros. El programa recorre el array y va sumando en un acumulador cada uno de los valores del mismo. Para sumar se utiliza la función **`suma`** que, dados dos valores de tipo entero, devuelve el resultado de sumar ambos. Pese a que la operación suma es simple, simulamos una operación compleja (con un tiempo de cómputo mayor) utilizando un retardo aleatorio.

Para facilitar la depuración del programa, el array de números enteros se generará de forma aleatoria cada vez que se ejecute el programa.

El código de dicho programa se muestra a continuación:

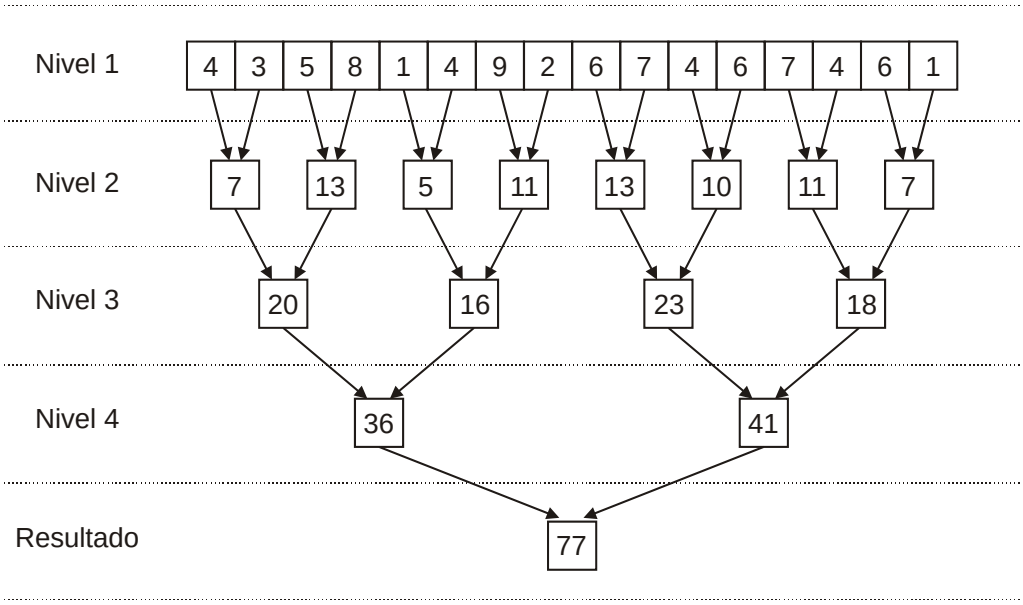
Enunciado Práctica Evaluable 1

```
package practica1;

import java.util.Arrays;

import static simpleconcurrent.SimpleConcurrent.*;

public class SumaSecuencial {

    public static final int N_PROCESOS = 5;
    public static final int N_DATOS = 16;

    private static int[] datos = new int[N_DATOS];

    private static int suma(int a, int b) {
        sleepRandom(1000);
        return a + b;
    }


    private static void inicializaDatos() {

        for (int i = 0; i < N_DATOS; i++) {
            datos[i] = (int) (Math.random() * 11);
        }

        println("Los datos a sumar son: "+Arrays.toString(datos));
    }

    public static void main(String[] args) {

        inicializaDatos();

        int sum = 0;
        for (int i = 0; i < N_DATOS; i++) {
            sum = suma(sum,datos[i]);
        }
        println("Suma: " + sum);
    }
}
```

Se desea paralelizar este programa para que se ejecute más rápidamente en un sistema con varios procesadores. Para ello, el programa estará formado por varios procesos. La ejecución en paralelo requiere un cambio en la forma de calcular la suma de todos los elementos, para que varias operaciones de suma puedan realizarse concurrentemente.

Para realizar la suma, los procesos deben coordinarse para tomar cada uno de ellos dos valores, sumarlos y guardar el resultado. Cuando un proceso haya sumado dos valores, comprobará si quedan valores por sumar y si es así, tomará otros dos y repetirá el proceso. De esta forma, la suma de 16 valores se realizaría siguiendo este esquema:

Enunciado Práctica Evaluable 1



Como se puede observar, la suma se realiza por niveles. Una posible implementación para realizar esta suma por niveles podría utilizar las siguientes estructuras de datos y variables:

```
public static int[] datos1 = new int[16];
public static int[] datos2 = new int[8];
public static int[] datos3 = new int[4];
public static int[] datos4 = new int[2];
public static int resultado5;
```

Usando estos arrays, la suma de **datos1[1]** y **datos1[2]** deberá guardarse en **datos2[1].** De la misma forma, la suma de **datos1[3]** y **datos1[4]** deberá guardarse en **datos2[2].** De forma general, la suma de los valores de **datos1** se podría implementar con el siguiente código:

```
int a = datos1[posSumar];
int b = datos1[posSumar + 1];

int valorSuma = suma(a, b);

int posResult = posSumar / 2; // ¡Cuidado!¡División entera!
datos2[posResult] = valorSuma;
```

No obstante, queremos realizar una implementación que consuma la menor cantidad de memoria posible. Debido a esta restricción, solo vamos a poder utilizar dos arrays de enteros para la suma.

Como solo disponemos de dos arrays, vamos a tener que sumar los niveles de uno en uno. Usaremos un array llamado **datos** que contendrá los datos que hay que sumar y otro array llamado **resultados** donde se deben ir almacenando los resultados intermedios. Cuando todos los valores de **datos** se hayan sumado podremos proceder a la suma del siguiente nivel. Así se procederá hasta que se alcance el último nivel cuyo resultado es el valor final de la suma. Cada vez que se termine de calcular la suma de un nivel, el array **resultados**, deberá copiarse al array de **datos** para comenzar de nuevo con el siguiente nivel.

En el siguiente esquema, se puede ver como se realizaría la suma por niveles usando el array de **datos** y **resultados**.

Enunciado Práctica Evaluable 1

Se pide

Se pide implementar completamente en Java, utilizando **SimpleConcurrent,** un programa concurrente que calcule la suma de un array de números enteros. La suma debe realizarse por niveles, de forma que no se puede comenzar a sumar los enteros de un nivel hasta que no se haya finalizado con el nivel anterior.

El programa estará formado por 5 procesos o *threads* del mismo tipo que llevarán a cabo de forma concurrente la suma de los enteros de un nivel.

Cada proceso tiene que realizar tres acciones:

1. Determinar cuáles son los dos números que tiene que sumar.
2. Realizar de forma concurrente la suma de los datos utilizando la función **`suma`** proporcionada. Hay que notar que mientras un proceso está realizando la suma, otro proceso también lo podría hacer. Por ejemplo, si el **`Proceso2`** está sumando **`datos[1]`** y **`datos[2]`**, el **`Proceso4`** podría estar sumando **`datos[5]`** y **`datos[6].`**
3. Guardar el resultado de la suma en el array **`resultados`**.

Un proceso no puede sumar más datos cuando todos los datos de ese nivel se hayan sumado ya o bien los que quedan por sumar están siendo sumados por otros procesos.

Cuando ya no hay más datos que sumar los procesos deben quedarse bloqueados esperando a que el último proceso en calcular la suma prepare todo lo necesario para comenzar con el siguiente nivel y los desbloquee.

Enunciado Práctica Evaluable 1

El último proceso en sumar debe realizar las siguientes acciones:

1. Copiar el contenido del array **`resultados`** al array **`datos`**.
2. Reinicializar aquellas variables necesarias para la suma del siguiente nivel.
3. Desbloquear al resto de procesos.

Al finalizar la suma de todos los datos, el último proceso deberá mostrar el resultado por pantalla.

Para facilitar la correcta implementación de la solución, se podrá realizar el cálculo de la suma de forma secuencial antes de comenzar la suma concurrente. De esa forma se podrá verificar de forma sencilla si la solución implementada calcula la suma de forma correcta.

La sincronización de los procesos deberá llevarse a cabo únicamente con **semáforos**. No se podrá utilizar la espera activa.

La salida por pantalla al ejecutar la aplicación deberá ser similar a la siguiente ejecución:

```
Los datos a sumar son: [3, 8, 5, 5, 7, 6, 7, 3, 1, 10, 8, 7, 1, 6, 8, 9]
Suma: 94
Proceso 1: Se inicia la suma de datos[0]=3 y datos[1]=8
Proceso 0: Se inicia la suma de datos[6]=7 y datos[7]=3
Proceso 4: Se inicia la suma de datos[2]=5 y datos[3]=5
Proceso 2: Se inicia la suma de datos[8]=1 y datos[9]=10
Proceso 3: Se inicia la suma de datos[4]=7 y datos[5]=6
Proceso 4: Se guarda la suma en resultados[1]=10
Proceso 4: Se inicia la suma de datos[10]=8 y datos[11]=7
Proceso 0: Se guarda la suma en resultados[3]=10
Proceso 0: Se inicia la suma de datos[12]=1 y datos[13]=6
Proceso 2: Se guarda la suma en resultados[4]=11
Proceso 2: Se inicia la suma de datos[14]=8 y datos[15]=9
Proceso 2: Se guarda la suma en resultados[7]=17
Proceso 2: Esperando a los demás procesos. Han terminado 1 procesos
Proceso 0: Se guarda la suma en resultados[6]=7
Proceso 0: Esperando a los demás procesos. Han terminado 2 procesos
Proceso 3: Se guarda la suma en resultados[2]=13
Proceso 3: Esperando a los demás procesos. Han terminado 3 procesos
Proceso 1: Se guarda la suma en resultados[0]=11
Proceso 1: Esperando a los demás procesos. Han terminado 4 procesos
Proceso 4: Se guarda la suma en resultados[5]=15
Proceso 4: Actualiza el array de datos a [11,10,13,10,11,15,7,17,]
Proceso 4: Finalizado el nivel 1
------------------------------------------------------
Proceso 3: Se inicia la suma de datos[4]=11 y datos[5]=15
Proceso 1: Esperando a los demás procesos. Han terminado 1 procesos
Proceso 0: Se inicia la suma de datos[2]=13 y datos[3]=10
Proceso 4: Se inicia la suma de datos[6]=7 y datos[7]=17
Proceso 2: Se inicia la suma de datos[0]=11 y datos[1]=10
Proceso 2: Se guarda la suma en resultados[0]=21
Proceso 2: Esperando a los demás procesos. Han terminado 2 procesos
Proceso 0: Se guarda la suma en resultados[1]=23
Proceso 0: Esperando a los demás procesos. Han terminado 3 procesos
Proceso 4: Se guarda la suma en resultados[3]=24
Proceso 4: Esperando a los demás procesos. Han terminado 4 procesos
Proceso 3: Se guarda la suma en resultados[2]=26
Proceso 3: Actualiza el array de datos a [21,23,26,24,]
Proceso 3: Finalizado el nivel 2
------------------------------------------------------
Proceso 2: Esperando a los demás procesos. Han terminado 1 procesos
```

```
Proceso 3: Se inicia la suma de datos[0]=21 y datos[1]=23
Proceso 1: Se inicia la suma de datos[2]=26 y datos[3]=24
Proceso 3: Se guarda la suma en resultados[0]=44
Proceso 0: Esperando a los demás procesos. Han terminado 2 procesos
Proceso 1: Se guarda la suma en resultados[1]=50
Proceso 4: Esperando a los demás procesos. Han terminado 3 procesos
Proceso 3: Esperando a los demás procesos. Han terminado 4 procesos
Proceso 1: Actualiza el array de datos a [44,50,]
Proceso 1: Finalizado el nivel 3
-----------------------------------------------------
Proceso 2: Esperando a los demás procesos. Han terminado 1 procesos
Proceso 0: Esperando a los demás procesos. Han terminado 2 procesos
Proceso 1: Se inicia la suma de datos[0]=44 y datos[1]=50
Proceso 4: Esperando a los demás procesos. Han terminado 3 procesos
Proceso 3: Esperando a los demás procesos. Han terminado 4 procesos
Proceso 1: Se guarda la suma en resultados[0]=94
Proceso 1: Finalizado el nivel 4
-----------------------------------------------------
El resultado es: 94
```

Memoria

La memoria deberá contener las siguientes secciones:

- **Descripción general:** describe de forma general, sin entrar en detalles, cómo funciona el programa implementado.
- **Tipos de sincronización:** explica qué tipos de sincronización utilizas en el programa y en qué lugares lo haces.
- **Semáforos:** explica para qué sirven cada uno de los semáforos que has usado. Justifica el valor de inicialización.
- **Variables de control:** explica para qué sirve cada una de las variables de control que usas (contadores y variables booleanas)

Criterios de Corrección

La práctica se considerará APTA cuando se cumplan los siguientes criterios:

1. El valor de la suma tiene que ser correcto.
2. Los procesos tienen que poder calcular la suma de forma concurrente. Si la suma se realiza de forma secuencial, el programa será considerado NO APTO.
3. Los procesos no pueden comenzar a calcular las sumas de un nivel hasta que no se hayan calculado las sumas del nivel anterior.
4. La salida por pantalla deberá mostrar la misma información que la que se muestra de ejemplo.
5. El programa deberá finalizar su ejecución cuando se haya calculado la suma total.
6. No se podrá usar espera activa, únicamente semáforos.

El incumplimiento de cualquiera de estos criterios supondrá la calificación de la práctica como suspensa.