PROGRAMACIÓN ORIENTADA A OBJETOS

TRABAJO PRÁCTICO N°6

1) Implemente un programa que escriba un “hola mundo” por cada hilo de ejecución que se cree (seis es un número razonable) y que además indique desde qué hilo se imprime. Luego haga que cada uno espere un tiempo proporcional a su identificador antes de imprimir el mensaje (el thread 1, un segundo, el 2, dos segundos, el 3, tres segundos).

2) Implemente un programa que lance cuatro threads, cada uno incrementará una variable contador de tipo entero, compartida por todos, 5000 veces y luego imprima.

   Java: ¿Que diferencia hay entre trabajar con Runnable o Threads?

   Python:¿Cómo debe ser compartida si trabajamos con Thread? ¿Y con Process?

3) Implemente un cronómetro que cumpla con las siguientes funcionalidades:

   - Iniciar el conteo de tiempo (Iniciar el conteo lo que debe realizar es el inicio del contador principal).
   - Detener el conteo de tiempo (Volver a cero los contadores).
   - Crear parciales de tiempo.
   - Para crear un parcial de tiempo se deberá utilizar un sub contador que trabajará a la par del conteo principal. Para realizar este ejercicio utilice los patrones MVC y Observer.

4) Implemente una simulación de un entrenamiento de carrera de posta para un equipo de cuatro integrantes. En este tipo de carreras el primer integrante se coloca en línea de largada y los otros tres en posiciones más adelantadas. Cuando la carrera comienza el primer integrante debe correr hasta la posición del integrante 2 y entregar la posta; luego el integrante 2 corre hasta la posición del integrante 3 y entrega la posta; luego este hasta el integrante 4 y finalmente, este último, corre hasta la línea de llegada. Es decir, que cada integrante del equipo comienza a correr cuando el anterior le entrega la posta y termina cuando hace entrega al siguiente.

   Para esta simulación cada Corredor extenderá de Thread y finalizará dentro de un rango random de entre 5 y 10 segundos. Solo a partir de que termine podrá comenzar el siguiente corredor.

La impresión en salida debe ser similar a la siguiente:

Inicia Corredor 1
Tiempo: 7 segundos
Inicia Corredor 2
Tiempo: 8 segundos
Inicia Corredor 3
Tiempo: 8 segundos
Inicia Corredor 4
Tiempo: 7 segundos
Tiempo total 30 segundos.

5) Desarrolla una clase Cuenta con un atributo saldo de tipo Float. Está representará a una cuenta bancaria y contendrá el monto monetario que se indique por defecto en el constructor.

   La clase debe poseer un método que permita descontar a ese saldo la cantidad indicada.

   Luego, cree una clase Tarjeta que extienda de Process y reciba en su constructor un número de identificación y una instancia de la clase Cuenta. Es decir, toda tarjeta estará asociada a una cuenta bancaria.

   Implemente el método run de tal manera que ejecute dos veces un descuento de saldo a la cuenta, a fines de prueba este descuento será siempre por $100.

   Cree una instancia de la clase Cuenta con un saldo de $2000. Cree diez instancias de la clase Tarjeta y asocie todas a la cuenta creada previamente. Ejecute el método start de cada instancia de tarjeta(las ejecuciones deben ser en paralelo)

   Al finalizar la ejecución de todos los procesos debe imprimirse el resultado. Este debería de ser cero. Valide ejecutando varias veces el programa asegura que no se obtiene un falso positivo.

Ejemplo de salida

```
Nuevo gasto - Tarjeta: 10 - Monto:100.0
Nuevo gasto - Tarjeta: 10 - Monto:100.0
Nuevo gasto - Tarjeta: 2 - Monto:100.0
Nuevo gasto - Tarjeta: 2 - Monto:100.0
Nuevo gasto - Tarjeta: 3 - Monto:100.0
Nuevo gasto - Tarjeta: 8 - Monto:100.0
Nuevo gasto - Tarjeta: 3 - Monto:100.0
Nuevo gasto - Tarjeta: 5 - Monto:100.0
Nuevo gasto - Tarjeta: 7 - Monto:100.0
Nuevo gasto - Tarjeta: 5 - Monto:100.0
Nuevo gasto - Tarjeta: 9 - Monto:100.0
Nuevo gasto - Tarjeta: 6 - Monto:100.0
Nuevo gasto - Tarjeta: 1 - Monto:100.0
Nuevo gasto - Tarjeta: 6 - Monto:100.0
Nuevo gasto - Tarjeta: 4 - Monto:100.0
Nuevo gasto - Tarjeta: 1 - Monto:100.0
Nuevo gasto - Tarjeta: 9 - Monto:100.0
Nuevo gasto - Tarjeta: 7 - Monto:100.0
Nuevo gasto - Tarjeta: 8 - Monto:100.0
Nuevo gasto - Tarjeta: 4 - Monto:100.0
Saldo final: 0.0
```

6) Identificar en las siguientes porciones de código porque se produce un deadlock y arreglar si se puede:

Python:

a)

```python
from multiprocessing import Process
from multiprocessing import Lock

def task(lock):
    print('Adquiriendo el lock...', flush=True)
    with lock:
        print('Adquiriendo el lock otra vez...', flush=True)
        with lock:
            pass
if __name__ == '__main__':
    lock = Lock()
    process = Process(target=task, args=(lock,))
    process.start()
    process.join()
```

b)

```python
from multiprocessing import parent_process, current_process, Process

def task():
    current = current_process()
    parent = parent_process()
    print(f'[{current.name}] esperando a [{parent.name}]...', flush=True)
    parent.join()

if __name__ == '__main__':
    current = current_process()
    child = Process(target=task)
    child.start()
    print(f'[{current.name}] esperando a [{child.name}]...', flush=True)
    child.join()
```

Java:

```java
public class Main {
    public static Object Lock1 = new Object();
    public static Object Lock2 = new Object();

    public static void main(String[] args) {
        Hilo1 hilo1 = new Hilo1();
        Hilo2 hilo2 = new Hilo2();
        hilo1.start();
        hilo2.start();
    }

    private static class Hilo1 extends Thread {
        public void run() {
            synchronized (Lock1) {
                System.out.println("Hilo 1: Reteniendo a Lock1...");
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                }
                System.out.println("Hilo 1: Esperando a Lock2...");
                synchronized (Lock2) {
                    System.out.println("Hilo 1: Reteniendo a Lock1 y Lock2");
                }
            }
        }
    }
```

```
    private static class Hilo2 extends Thread {
        public void run() {
            synchronized (Lock2) {
                System.out.println("Hilo 2: Reteniendo a Lock2...");
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                }
                System.out.println("Hilo 2: Esperando por Lock1...");
                synchronized (Lock1) {
                    System.out.println("Hilo 2: Reteniendo a Lock1 y Lock2");
                }
            }
        }
    }
}
```

7) Dado el escenario en el cual se encuentran dos personas intentando comer sopa pero se encuentran con que solo tienen una cuchara para hacerlo. ¿Qué tipo de problema con hilos se produce? ¿Cómo se resolvería?

8) Haciendo uso de Pool de multiprocessing, realice una sumatoria de dos matrices de gran tamaño con números random. Realice la comparativa de realizar el mismo cálculo de forma secuencial.

9) El sistema nacional de datos del mar cuenta con 2 boyas ubicadas en el golfo San Jorge, llamadas CIDMAR-1 y CIDMAR-2 estas cuentan con una gran cantidad de sensores pero solo se pedirán los siguientes:

   - Temperatura ambiente (Termómetro): los valores promedio rondan entre 5.0 a 30.0 grados centígrados
   - Velocidad del viento (Anemómetro): los valores promedio rondan entre 0 a 60km/h Las clases Termómetro y anemómetro implementan la interfaz sensor, que es la siguiente:

```
public interface Sensor {

    public Double sensar();

}
```

   La boya debe generar 5 paquetes de datos y enviarlos al servidor, el cual se encarga de almacenarlos en su cola de paquetes.

El envío de paquete tiene una probabilidad del 40% de fallar, cuando eso pase debe imprimir en consola el mensaje:

"ERROR: CIDMAR-1 Hubo una falla en la comunicación, se perdió el paquete"

con la ayuda de una excepción creada por usted mismo.

El proceso de enviar un paquete consume 5000 milisegundos.

Al momento de almacenar un paquete y consultar el servidor se encarga de imprimir por pantalla la acción.

El científico se encarga de consultar los datos al servidor y anotarlos en su cuaderno (colaPaquetes), por último, realiza un informe con los promedios de temperatura y velocidad de viento para cada boya.

El programa no se debe bloquear, debe respetar el orden de producción y consumición y finalizar su ejecución correctamente.

Ejemplo de simulación:

```
ERROR: CIDMAR-1 Hubo una falla en la comunicacion, se perdio el paquete
ERROR: CIDMAR-2 Hubo una falla en la comunicacion, se perdio el paquete
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=28.963019670989993, velocidadViento=12.942493033888622, tiempo=1447209983960]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=28.963019670989993, velocidadViento=12.942493033888622, tiempo=1447209983960]
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=21.018825872773693, velocidadViento=3.6021974587659344, tiempo=1447209983961]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=21.018825872773693, velocidadViento=3.6021974587659344, tiempo=1447209983961]
ERROR: CIDMAR-1 Hubo una falla en la comunicacion, se perdio el paquete
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=23.782904328052485, velocidadViento=34.39246431883105, tiempo=1447209988964]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=23.782904328052485, velocidadViento=34.39246431883105, tiempo=1447209988964]
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=29.87252576542097, velocidadViento=27.52670261447227, tiempo=1447209993965]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=29.87252576542097, velocidadViento=27.52670261447227, tiempo=1447209993965]
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=29.867145290651052, velocidadViento=46.405842464006085, tiempo=1447209993965]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=29.867145290651052, velocidadViento=46.405842464006085, tiempo=1447209993965]
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=29.336484407874224, velocidadViento=23.59196656452306, tiempo=1447209998967]
ALMACENO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=13.365413434521496, velocidadViento=36.76964612361764, tiempo=1447209998968]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-2, temperatura=29.336484407874224, velocidadViento=23.59196656452306, tiempo=1447209998967]
CONSULTO: PaqueteDatos [nombreBoya=CIDMAR-1, temperatura=13.365413434521496, velocidadViento=36.76964612361764, tiempo=1447209998968]

 INFORME DE PROMEDIOS:
El promedio de temperatura para la Boya CIDMAR-1 es :24.065192798720847
El promedio de velocidad de viento para la Boya CIDMAR-1 es :24.065192798720847
El promedio de temperatura para la Boya CIDMAR-2 es :26.002685093530342
El promedio de velocidad de viento para la Boya CIDMAR-2 es :26.002685093530342
```

10) Implemente un programa que sea capaz de abrir un archivo de imagen para aplicarle un filtro de color gris y guarde el nuevo archivo generado.

    Para ello debera:

    1.
        a. Aplicar el filtro a toda la imagen en una unica operación.
        b. Guardar el archivo modificado
    2.
        a. Determinar cuantos nucleos de procesamiento tiene su computadora.
        b. Dividir la imagen en N partes de igual tamaño (Donde N es la cantidad de nucleos), para luego aplicar el filtro a cada parte de la imagen, utilizando un hilo distinto para cada una, es decir, un hilo por cada nucleo disponible.
        c. Unir las partes de la imagen nuevamente y guardarla

Tanto para el procesamiento secuencial como para el procesamiento multi hilado, deberá tomar el tiempo que toma aplicar el filtro a la imagen para comparar rendimientos.

Puede utilizar la siguiente implementacion de Run para aplicar el filtro sobre la imágen.

```
@Override
public void run() {
   for (int y = startY; y < endY; y++) {
       for (int x = startX; x < endX; x++) {
           int rgb = img.getRGB(x, y);
           int r = (rgb >> 16) & 0xFF;
           int g = (rgb >> 8) & 0xFF;
           int b = rgb & 0xFF;
           int gray = (r + g + b) / 3;
           int newPixel = (gray << 16) | (gray << 8) | gray;
           img.setRGB(x, y, newPixel);
       }
   }
}
```