Taller 1 – Threads

El propósito de este taller es entender la forma como se crean y corren los threads para implementar aplicaciones concurrentes en Java. La primera parte del taller presenta diferentes formas de creación y la segunda parte ilustra su ejecución.

Parte 1 - Creación de Threads

En Java existen dos formas de realizar la implementación de los threads, bien sea mediante la extensión de la clase **`Thread`** o la implementación de la interface **`Runnable`**.

Ejemplo 1: Creación de threads como extensión de la clase `Thread`

Revise el código de la clase **`EjThreads01`**.

```
public class EjThreads01 extends Thread {

	public void run() {
		System.out.println("Extendiendo la clase Thread.");
	}

	public static void main(String[] args) {
		EjThreads01 t = new EjThreads01();

		t.start();
	}
}
```

Ejemplo 2: Creación de threads como implementación de la interfaz `Runnable`

Revise el código de la clase **`EjThreads02`**.

```
public class EjThreads02 implements Runnable {

	public void run() {
		System.out.println("Implementando la interfaz Runnable.");
	}

	public static void main(String[] args) {

		Thread t = new Thread(new EjThreads02());

		t.start();
	}
}
```

Carlos Eduardo Gómez Montoya
Jairo Emilio Bautista Mora
Geovanny Andrés González Rodríguez

Ejemplo 3: Creación de threads con un valor inicial – como extensión de la clase `Thread`

Revise el código de la clase **`EjThreads01a`**.

```
public class EjThreads01a extends Thread {

	private int n;

	public EjThreads01a(int n) {
		System.out.println("Extendiendo la clase Thread.");
		this.n = n;
	}

	public void run() {
		System.out.println("El valor inicial es: " + n);
	}

	public static void main(String[] args) {
		EjThreads01a t = new EjThreads01a(5);

		t.start();
	}
}
```

Ejemplo 4: Creación de threads con un valor inicial – como implementación de la interface `Runnable`

Revise el código de la clase **`EjThreads02a`**.

```
public class EjThreads02a implements Runnable {

	private int n;

	public EjThreads02a(int n) {
		System.out.println("Implementando la interfaz Runnable.");
		this.n = n;
	}

	public void run() {
		System.out.println("El valor inicial es: " + n);
	}

	public static void main(String[] args) {

		Thread t = new Thread(new EjThreads02a(5));

		t.start();
	}
}
```

Ejemplo 5: Creación de threads con nombre – como extensión de la clase `Thread`

Revise el código de la clase **`EjThreads01b`**.

```
public class EjThreads01b extends Thread {

	private String name;

	public EjThreads01b(String name) {
		System.out.println("Extendiendo la clase Thread.");
		this.name = name;
	}

	public void run() {
		System.out.println("El nombre es: " + name);
	}

	public static void main(String[] args) {
		EjThreads01b t0 = new EjThreads01b("Thread"+0);
		EjThreads01b t1 = new EjThreads01b("Thread"+1);
		EjThreads01b t2 = new EjThreads01b("Thread"+2);

		t0.start();
		t1.start();
		t2.start();
	}
}
```

Ejemplo 6: Creación de threads con nombre – como implementación de la interface `Runnable`

Revise el código de la clase **`EjThreads02b`**.

```
public class EjThreads02b implements Runnable {

	private String name;

	public EjThreads02b(String name) {
		System.out.println("Implementando la interfaz Runnable.");
		this.name = name;
	}

	public void run() {
		System.out.println("El nombre es: " + name);
	}

	public static void main(String[] args) {

		Thread t0 = new Thread(new EjThreads02b("Thread"+0));
		Thread t1 = new Thread(new EjThreads02b("Thread"+1));
		Thread t2 = new Thread(new EjThreads02b("Thread"+2));

		t0.start();
		t1.start();
		t2.start();
	}
}
```

Ejemplo 7: Creación de threads con nombre usando ciclos – como extensión de la clase `Thread` – `Uso con arreglos`

Revise el código de la clase **`EjThreads01c`**.

```
public class EjThreads01c extends Thread {

	private final static int MAX = 3;

	private String name;

	public EjThreads01c(String name) {
		System.out.println("Extendiendo la clase Thread.");
		this.name = name;
	}

	public void run() {
		System.out.println("El nombre es: " + name);
	}

	public static void main(String[] args) {
		EjThreads01c [] ta = new EjThreads01c[MAX];

		for (int i = 0; i < ta.length; i++) {
			ta[i] = new EjThreads01c("Thread"+i);
		}

		for (int i = 0; i < ta.length; i++) {
			ta[i].start();
		}
	}
}
```

Ejemplo 8: Creación de threads con nombre usando ciclos – como implementación de la interface `Runnable` – Uso de `arreglos`

Revise el código de la clase **`EjThreads02c`**.

```
public class EjThreads02c implements Runnable {

	private final static int MAX = 3;
	private String name;

	public EjThreads02c(String name) {
		System.out.println("Implementando la interfaz Runnable.");
		this.name = name;
	}


	public void run() {
		System.out.println("El nombre es: " + name);
	}

	public static void main(String[] args) {
		Thread [] ta = new Thread[MAX];

		for (int i = 0; i < ta.length; i++) {
			ta[i] = new Thread(new EjThreads02c("Thread"+i));
		}

		for (int i = 0; i < ta.length; i++) {
			ta[i].start();
		}
	}
}
```

Parte 2 - Ejecución

Ejemplo 9: Aplicación multithread para encontrar el elemento mayor de una matriz de enteros

El ejemplo a continuación muestra cómo utilizar threads para que de manera concurrente se pueda encontrar el mayor de los elementos de una matriz de enteros.

```java
import java.util.concurrent.ThreadLocalRandom;

public class MaximoMatriz extends Thread {
    //Vamos a generar los numeros aleatorios en un intervalo amplio
    private final static int INT_MAX = 105345;

    //Dimensiones cuadradas
    private final static int DIM = 3;

    //Matriz
    private static int[][] matriz = new int[DIM][DIM];

    //Mayor global
    private static int mayor = -1;

    //Mayor local
    private int mayorFila = -1;

    //ID Thread
    private int idThread;

    //Fila a registrar
    private int fila;

    //Constructor
    public MaximoMatriz(int pIdThread, int pFila) {
        this.idThread = pIdThread;
        this.fila = pFila;
    }
```

```java
//Generar la matriz con números aleatorios
public static void crearMatriz() {
    for (int i = 0; i < DIM; i++) {
        for(int j = 0; j < DIM; j++) {
            matriz[i][j] = ThreadLocalRandom.current().nextInt(0, INT_MAX);
        }
    }
    //Imprimir la matriz
    System.out.println("Matriz:");
    System.out.println("============================");
    imprimirMatriz();
}

//Imprimir la matriz en consola
private static void imprimirMatriz() {
    for (int i = 0; i < DIM; i++) {
        for (int j = 0; j < DIM; j++) {
            System.out.print(matriz[i][j] + "\t");
        }
        System.out.println();
    }
}
```

```
    @Override
    public void run() {
        for (int j = 0; j < DIM; j++) {
            if (this.mayorFila < matriz[this.fila][j]) {
                this.mayorFila = matriz[this.fila][j];
            }
        }
        if (this.mayorFila > mayor) {
            try {
                Thread.sleep(250);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            mayor = this.mayorFila;
            String warn = String.format(
                    "============ Nuevo maximo encontrado =========== \n " +
                    "ID Thread: %d - Maximo local actual: %d - Maximo global: %d \n" +
                    "\n",
                    this.idThread,
                    mayor,
                    this.mayorFila
            );
            System.out.println(warn);
        }
        //Resultados
        String msg = String.format("ID Thread: %d - Maximo Local: %d - Maximo Global: %d",
                this.idThread,
                this.mayorFila,
                mayor);
        System.out.println(msg);
    }

    //Main
    public static void main(String[] args) {
        System.out.println("Busqueda concurrente por una matriz");

        //Iniciar la matriz
        MaximoMatriz.crearMatriz();
        System.out.println();
        System.out.println("Iniciando la busqueda por la matriz \n");

        //Iniciar busqueda
        MaximoMatriz[] bThreads = new MaximoMatriz[DIM];
        for (int i = 0; i < DIM; i++) {
            bThreads[i] = new MaximoMatriz(i, i);
            bThreads[i].start();
        }
    }
}
```

Responda:

1. Ejecute cinco veces el programa y escriba el resultado obtenido en cada ejecución.

Ejecución	Valor obtenido	Valor esperado
1		
2		
3		
4		
5		

2. ¿Hay acceso concurrente a alguna variable compartida? Si es así, diga en dónde.

3. ¿Cómo explica el comportamiento observado?