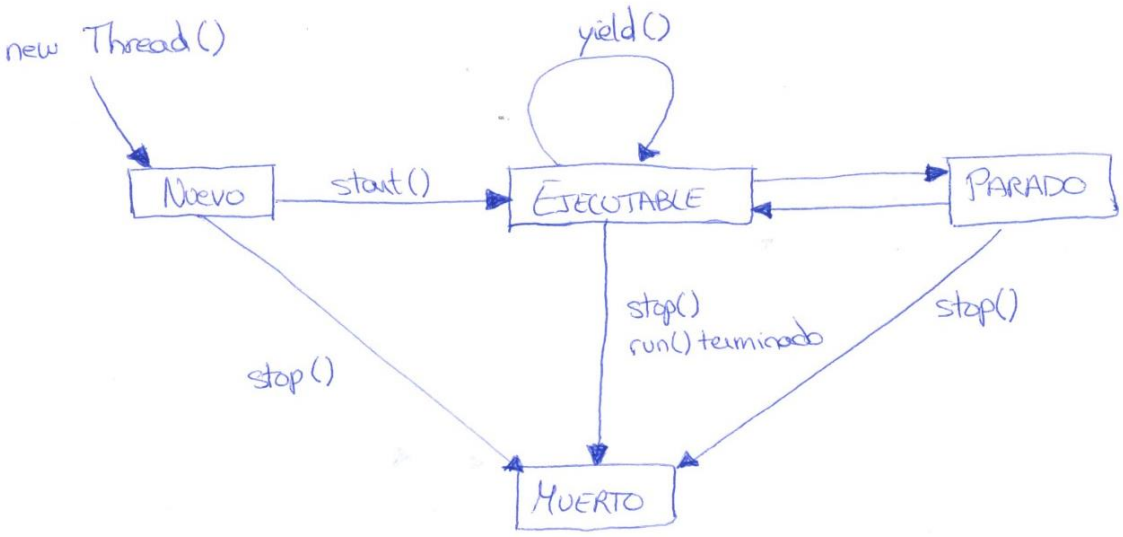
# UF2. [PAC02] Solución

## Actividades

**Parte teórica**

1. Haz un esquema de todos los estados de un hilo, indicando que ocurre en cada uno de ellos.




**NUEVO**: Se crea un nuevo hilo, pero no se arranca. El hilo es un objeto Thread vacío.

**EJECUTABLE**: El hilo está disponible para ponerse en ejecución, pero no necesariamente está en ejecución.

**PARADO**: El hilo podría ejecutarse, pero hay algo que lo evita.

**MUERTO**: La ejecución del hilo ha terminado, ya sea por muerte natural o no.

## Parte práctica

2. Implemente al problema de los filósofos en Java. Ayúdate de las siguientes indicaciones.

Necesitamos tres clases Java:

- **Filósofo**, donde se van a realizar todas las operaciones relacionadas con el filósofo.
- **Tenedor**, donde se van a realizar todas las operaciones relacionadas con el tenedor.
- **Filósofos**, el programa principal que lo lanzará todo.

Delante de los métodos de la clase tenedor deberemos poner **synchronized**, con esto nos aseguramos que solo va a existir un hilo ejecutándose a la vez en ese método.

La clase filósofo debe extender de **Thread**, para poder usar el método run dentro de esta clase cuando se llame al método **start()** desde la clase filósofos.

### Clase Filosofo

```
import java.util.Random;

/*
 * Clase filosofo
 * Esta clase es la que debe consumir los recursos, en este caso
 * representados por los tenedores.
 *
 *  Extiende de la clase Thread porque para poder crear procesos concurrentes
 *  necesitamos hacer uso de los hilos que en Java se manejan con esta clase.
 *
 */

public class Filosofo extends Thread {
	private int identificativo;
	private Tenedor izquierda;
	private Tenedor derecha;
	private Random random;
	private boolean izquierdatomado;
	private boolean derechatomado;
	private int elegido;
	private boolean flag;

	//Mediante el constructor le asisgnamos un identificador en forma de
entero
	//Y los dos objetos tenedores que pueden, potencialmente, ser
consmidos por el filosofo.

	Filosofo(int identificativo_, Tenedor izquierda_, Tenedor derecha_) {
		identificativo = identificativo_;
		izquierda = izquierda_;
		derecha = derecha_;
		random = new Random();
	}
```

```
	//Esta función tiene como único cometido detener la ejecución del hilo durante 10000 ms
	//Digamos que el estado Pensando es En espera.
	public void pensar() {
		try {
			Thread.sleep(10000);
		} catch (InterruptedException ie) {
		}
	}
	//Función similar a la anterior, que emula el estado de ejecución.
	public void comer() {
		try {
			System.out.println("El filosofo Nº "+ identificativo+ "está comiendo");

			Thread.sleep(10000);
			//System.out.println("terminando de comer...");
			//System.out.println(identificativo);

		} catch (InterruptedException ie) {
		}
	}

	//Función encargada de gestionar la ejecución del proceso
	public void run() {
		while (true) {
			this.pensar(); //Comenzamos poniendo el proceso en espera, o pensando

			flag = false;

			elegido = random.nextInt(2); //Elegimos de forma aleatoria el primer tenedor
												//Que intentaremos coger.

			//Dependiendo de cual elijamos de fomra aleatoria Intentamos tomarlo

			if (elegido == 0) {
				if (!izquierda.esocupado()) {
					izquierda.tomar();
					izquierdatomado = true;
				} else if (!derecha.esocupado()) {
					derecha.tomar();
					derechatomado = true;
				}
			} else if (elegido == 1) {
				if (!derecha.esocupado()) {
					derecha.tomar();
					derechatomado = true;
				} else if (!izquierda.esocupado()) {
					izquierda.tomar();
					izquierdatomado = true;
				}
			}
```

```
                        //En el caso de que tengamos uno tomado, intentamos
tomar el otro
                        //Si tenemos exito establecemos el flag de
actividad a true
                        //En caso contrario soltamos el tenedor que
teniamos para dejar el recurso
                        //Disponible
                  if (izquierdatomado == true) {
                        if (!derecha.esocupado()) {
                              derecha.tomar();
                              derechatomado = true;
                              flag = true;
                        } else {
                              izquierda.soltar();
                              izquierdatomado = false;
                        }
                  }

                  if (derechatomado == true && !flag) {
                        if (!izquierda.esocupado()) {
                              izquierda.tomar();
                              izquierdatomado = true;
                        } else {
                              derecha.soltar();
                              derechatomado = false;
                        }
                  }

                  //En el caso de que dispongamos de los dos tenedores
podemos empezar la
                  //ejecución del programa, por tanto llamamos a "comer"
                  if (derechatomado && izquierdatomado) {
                        this.comer();

                        elegido = random.nextInt(2);

                        if (elegido == 0) {
                              izquierda.soltar();
                              izquierdatomado = false;
                              derecha.soltar();
                              derechatomado = false;
                        } else {
                              derecha.soltar();
                              derechatomado = false;
                              izquierda.soltar();
                              izquierdatomado = false;
                        }
                  }
            }
      }
}
```

## Clase Tenedor

```
/* Clase Tenedor
 *
 * Representa el recurso del que tiene que hacer uso un proceso
 *
 *
 * */

public class Tenedor {

	private boolean ocupado = false;
	Tenedor(int identificativo_) {
	}

	/* Accion de soltar tenedor */

	synchronized void dejar() {
		ocupado = false;
		notify();
	}

	/* Comprobar si el tenedor esta ocupado */

	synchronized boolean esocupado() {
		if (ocupado) {
			return true;
		} else {
			return false;
		}
	}

	/* Accion de tomar el tenedor */

	synchronized void tomar() {
		ocupado = true;
	}

	void soltar() {
		ocupado = false;
	}
}
```

## Clase Filosofos

```
/*Clase Principal del programa*/

public class Filosofos{

	public static void main( String args[] )
	{

		Tenedor tenedor[]=new Tenedor[5];
```

```
            /*Instancias de los 5 tenedores */
            tenedor[0]=new Tenedor(0);
            tenedor[1]=new Tenedor(1);
            tenedor[2]=new Tenedor(2);
            tenedor[3]=new Tenedor(3);
            tenedor[4]=new Tenedor(4);

            Filosofo filosofo[]=new Filosofo[5];
            /*instancias de los cinco filosofos
             * Cada filosofo cuenta con dos tenedores, los cuals comparte
con el filosofo
             * anterior y posterior respectivamente.
             * */
            filosofo[0]=new Filosofo(0,tenedor[0],tenedor[1]);
            filosofo[1]=new Filosofo(1,tenedor[1],tenedor[2]);
            filosofo[2]=new Filosofo(2,tenedor[2],tenedor[3]);
            filosofo[3]=new Filosofo(3,tenedor[3],tenedor[4]);
            filosofo[4]=new Filosofo(4,tenedor[4],tenedor[0]);

            /*Comienza la ejecucion de los filosofos
             *
             * Esta ejecución no se detendrá, ya que no hemos establecido un
mecanismo de parada
             *
             * */
            filosofo[0].start();
            filosofo[1].start();
            filosofo[2].start();
            filosofo[3].start();
            filosofo[4].start();

      }
}
```