Práctica 1
Threads

Objetivo
Crear objetos derivados de la clase Thread.

Actividad
Compile, ejecute y explique (utilice gráficas) cómo se lleva a cabo la calendarización de los threads en los siguientes programas:

Programa 1

```
public class P1SimpleThread extends Thread {

  private int countDown = 5;
  private static int threadCount = 0;
  private int threadNumber = ++threadCount;
  public P1SimpleThread() {
    System.out.println("Construyendo " + threadNumber);
  }

  public void run() {
    while(true) {
      System.out.println("Thread " + threadNumber + "(" + countDown + ")");
      if(--countDown == 0) return;
    }
  }

 public static void main(String[] args) {
    for(int i = 0; i < 5; i++)
       new P1SimpleThread().start();
    System.out.println("Todos los threads comenzaron");
  }
}
```

Programa 2

```
public class P2RunnableThread implements Runnable {

  private int countDown = 5;
  public String toString() {
    return "Thread #" + Thread.currentThread().getName() +
      ": " + countDown;
  }

  public void run() {
    while(true) {
      System.out.println(this);
      if(--countDown == 0) return;
    }
  }
```

```
  public static void main(String[] args) {
    for(int i = 1; i <= 5; i++)
      new Thread(new P2RunnableThread(), "" + i).start();
  }
}
```

Programa 3

```
public class PrimeGenerator extends Thread{

	public void run() {
		long number=1L;

		while (true) {
			if (isPrime(number)) {
				System.out.printf("Number %d is Prime\n",number);
			}

			if (isInterrupted()) {
				System.out.printf("El generador de numeros primos
ha sido interrumpido\n");
				return;
			}
			number++;
		}
	}

	private boolean isPrime(long number) {
		if (number <=2) {
			return true;
		}
		for (long i=2; i<number; i++){
			if ((number % i)==0) {
				return false;
			}
		}
		return true;
	}

}


public class Main2 {

	public static void main(String[] args) {

		Thread task=new PrimeGenerator();
		task.start();

		try {
```

```
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            task.interrupt();
        }
}
```

Programa 4

```
class Sleeper extends Thread {

  private int duration;
  public Sleeper(String name, int sleepTime) {
    super(name);
    duration = sleepTime;
    start();
  }
  public void run() {
    System.out.println(getName() + " inicia su run");
    try {
      sleep(duration);
    } catch (InterruptedException e) {
      throw new RuntimeException(e);
    }
    System.out.println(getName() + " ha despertado");
  }
}

class Joiner extends Thread {

  private Sleeper sleeper;
  public Joiner(String name, Sleeper sleeper) {
    super(name);
    this.sleeper = sleeper;
    start();
  }

  public void run() {
    System.out.println(getName() + " inicia su run");
   try {
      sleeper.join();
    } catch (InterruptedException e) {
      throw new RuntimeException(e);
    }
    System.out.println(getName() + " join completado");
  }
}

public class P3Joining {
  public static void main(String[] args) {
    Sleeper sleepy = new Sleeper("Sleepy", 1500);
    Joiner dopey = new Joiner("Dopey", sleepy);
  }
}
```