PRIORIDADES HILO

1. Diseñar un nuevo proyecto **PSP02Ejer3_2PrioridadHilos** creando una clase implementándola con el interface runneable.

2. Definir los siguientes atributos:
   - un atributo hilo
   - un contador
   - una variable booleana: definir ***volatile boolean nomVarBooleana***;

     Esta variable booleana la utilizaremos para crear un bucle infinito y volatile asegura que el valor de la variable será examinado en cada una de las iteraciones. Sin el uso de volátile, Java es libre para optimizar el bucle de tal manera que el valor de ejecución se lleva a cabo en un registro de la CPU, y no necesariamente es de nuevo examen con cada iteración. El uso de volátile impide esta optimización que hace el Java, diciendo que la ejecución de Java puede cambiar de manera que no está directamente relacionada con la apariencia del código.

   - Crear el constructor pasándole dos parámetros un valor numérico que será la prioridad que tendrá el hilo cuando se cree, y una cadena de caracteres que será el nombre que se le asignará al hilo

   - Hacer el método run() como un bucle infinito – con el atributo booleano-, que se encargará de incrementar el atributo contador en cada iteración. Una vez finalizado el bucle deberá mostrar el siguiente mensaje:

     Hilo …….nomHilo …. Contador= ... valor contador

   - Hacer un método stop(), que permita modificar a false el valor del atributo booleana del bucle infinito anterior y que permitirá la detención del bucle.

   - Implementar el método start() para que lance el proceso del hilo del Thread

3. Diseñar una clase con un método main :

   - Asignarle la máxima prioridad al proceso main

   - Crear dos hilos, de la clase anterior hiloLOW e hiloHIGH el primero con una prioridad normal - 2 y hiloHIGH con una prioridad normal + 2, pasándole también como string para el nombre del hilo el mismo nombre que el del objeto

   - Lanzar los dos hilos

   - Dormir el proceso main durante 10 segundos (sleep)

- Parar los hilos creados invocando al método stop() de cada uno de los objetos runneables, mostrando a continuación si cada uno de los hilos está vivo y su estado.
- Invocar al método join del atributo hilo de los objetos runneables -hiloLOW e hiloHIGH -para que se fuerce el bloqueo de main, dejando el procesador libre y cediendo el control del mismo al hilo del objeto runneable para que se ejecute hasta que finalice el proceso correspondiente a hiloLOW e hiloHIGH; mostrando a continuación si cada uno de los hilos está vivo y su estado.

Comprobar los valores de los contadores de los dos hilos en función de su prioridad.

Jugar con la prioridad de cada uno de ellos y ver los resultados obtenidos.

SOLUCION

```
public class PSP02Ejer3_2PrioridadHilos implements Runnable {
  int cont = 0;
  Thread hilo;
  private volatile boolean running = true;
  public PSP02Ejer3_2PrioridadHilos(int p, String nom) {
    hilo = new Thread(this);
    hilo.setPriority(p);
    hilo.setName(nom);
  }

  public void run() {
    // ejecutamos el bucle de forma infinita
    //hasta que lo obliguemos a parar metodo stop
    while (running) {
      cont++;
    }
    System.out.println("hilo: "+Thread.currentThread().getName()+ " contador="+cont);
  }
  public void stop() {
    running = false;
  }
  public void start() {
    hilo.start();
  }
}
```

--- --- --- ---

```
import java.util.logging.Level;
import java.util.logging.Logger;
public class PSP02Ejer3_2Main {
    public static void main(String[] args) {
        Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
        PSP02Ejer3_2PrioridadHilos hiloHIGH =
                                    new      PSP02Ejer3_2PrioridadHilos(Thread.NORM_PRIORITY      +      2,
                        "hiloHIGH");
        PSP02Ejer3_2PrioridadHilos hiloLOW =
```

```
                                new PSP02Ejer3_2PrioridadHilos(Thread.NORM_PRIORITY - 2, "hiloLOW");
        hiloLOW.start();
        hiloHIGH.start();
        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            System.out.println("Hilo principal interrumpido.");
        }
        hiloLOW.stop();
        hiloHIGH.stop();
        System.out.println("hiloLOW: "+ hiloLOW.hilo.isAlive()+" " +hiloLOW.hilo.getState());
        System.out.println("hiloHIGH: "+ hiloHIGH.hilo.isAlive()+" " +hiloHIGH.hilo.getState());
       //forzamos a que antes de continuar el proceso main se ejecuten
        // y finalicen hiloLOW y hiloHIGH  -- join()
        try {
            hiloLOW.hilo.join();
            hiloHIGH.hilo.join();
        } catch (InterruptedException ex) {
            Logger.getLogger(PSP02Ejer3_2Main.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("hiloLOW: "+ hiloLOW.hilo.isAlive()+" " +hiloLOW.hilo.getState());
        System.out.println("hiloHIGH: "+ hiloHIGH.hilo.isAlive()+" " +hiloHIGH.hilo.getState());
    }
}
```