```java
package tarea2;

import java.util.logging.Level;
import java.util.logging.Logger;

public class SumatoriaThread extends Thread {

   static int pos = 0;//voy recorriendo el array
   NumerosCompartidos recurso;
   double sumatoria = 0.0;//guarda el valor de la sumatoria
   static boolean par = true;//selecciono que hilo hace la suma

   public SumatoriaThread(NumerosCompartidos recurso) {
      this.recurso = recurso;
      sumatoria = 0.0;
   }
   /*
 Este método se ejecuta solo cuando el atributo par es TRUE Así aseguro la alternancia de hilos
   */
   synchronized void entraHilo1(NumerosCompartidos recurso) {
      sumatoria += recurso.valores[pos];//voy acumulando la suma
      par = false;//cambio el valor para que entre el otro hilo
   }
   /*
   Este hilo se ejecuta solo cuando el atributo par es FALSE Así aseguro la alternancia de hilos.
   */
   synchronized void entraHilo2(NumerosCompartidos recurso) {
      sumatoria += recurso.valores[pos];//voy acumulando la otra suma

      par = true;//cambio el valor para que entre el otro hilo
   }

   @Override
   public void run() {
      /*
      El primer hilo que entre en el metodo run() se sincroniza con el recurso
      y bloquea el acceso de cualquier otro hilo que se lo quiera quedar
       */
      synchronized (recurso) {
         /*
         el bucle FOR va a ejecutar los hilos tantas veces como elementos haya en el array. Como hay varios hilos, utilizo una variable estática como atributo de clase para que todos los objetos pueden acceder a la posición de memoria (solo hay una pos_mem y no una por cada objeto por eso es estática)
          */
         for (pos = 0; pos < recurso.numValores; pos++) {
            /*
            como solo hay dos hilos, los vamos a diferenciar por un atributo booleano para distinguir entre uno u otro. Este booleano es el que determina que hilo entra
 */
```

```
            if (par) {
                entraHilo1(recurso);

                try {
                    Thread.sleep(500);// retraso
                } catch (InterruptedException ex) {
                    Logger.getLogger(SumatoriaThread.class.getName()).log(Level.SEVERE, null, ex);
                }

            } else {
                entraHilo2(recurso);

                try {
                    Thread.sleep(500);//retraso
                } catch (InterruptedException ex) {
                    Logger.getLogger(SumatoriaThread.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            recurso.notify();//una vez hecha la la tarea, el hilo notifica que deja libre el recurso...

            try {
                recurso.wait();// ....y luego el mismo hilo que libera el recurso se queda a la espera
            } catch (InterruptedException ex) {
                Logger.getLogger(SumatoriaThread.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        recurso.notify();//me aseguro que no dejo ningún hilo en espera
    }
  }
}
```

```java
package tarea2;

import java.util.logging.Level;
import java.util.logging.Logger;

public class GeneradorThread extends Thread {

   String nombreGenerador = null;//nombre del hilo
   int numTerminos = 0;//numero máximo de terminos que se van a generar
   int tiempo = 0;//tiempo de retraso del hilo en generar y añadir nuevos números
   NumerosCompartidos recurso = null;

   static String textoGenerado = "";//atributo estatico que va a guardar el resultado de TODOS LOS NUMEROS QUE GENEREN TODOS LOS HILOS
   static String textoHiloId = "";//atributo estatico que va a guardar TODOS LOS NUMEROS QUE GENEREN LOS HILOS CON SU ID
   String textoHilo = null;//atributo que va a guardar LOS NUMEROS QUE GENEREN CADA HILO
   double numeroGenerado;//Guarda el valor que se va a gardar en el Array

   public GeneradorThread(String nombreGenerador, int numTerminos, int tiempo, NumerosCompartidos recurso) {
      this.nombreGenerador = nombreGenerador;
      this.numTerminos = numTerminos;
      this.tiempo = tiempo;
      this.recurso = recurso;

      textoHilo = "";

   }

   public String getNombreGenerador() {
      return nombreGenerador;
   }

   /*
   CUIDADO DANIELITO QUE ESTO TIENE TRUCO!!!!
   El texto que genera el mensaje de salida de EJECUCION DE LOS HILOS GENERADORES DE VALORES
   se hace añadiendo cadenas a un String, pero ese String lo usan varios hilos a la vez, con lo que tenemos una condición de carrera (como lo de los jardines). Asi que hay que sincronizar el String para que entren los hilos de uno en uno
    */
   static synchronized String textoGen(GeneradorThread gen, double numeroGenerado) {
      textoGenerado = ("\nEl generador " + " "
            + gen.nombreGenerador
            + " ha generado el valor: "
            + numeroGenerado).concat(textoGenerado);
      return (textoGenerado);
   }

   /*
```

```
   CUIDADO DANIELITO QUE ESTO TIENE TRUCO!!!!
   El texto que genera el mensaje de salida de EJECUCION DE LOS HILOS GENERADORES DE VALORES se hace añadiendo cadenas a un String, pero ese String lo usan varios hilos a la vez, con lo que tenemos una condición de carrera (como lo de los jardines). Asi que hay que sincronizar el String para que entren los hilos de uno en uno
    */
   static synchronized String textoId(GeneradorThread gen, double numeroGenerado) {
      textoHiloId = ("\n" + gen.getId() + ": " + numeroGenerado).concat(textoHiloId);
      return textoHiloId;
   }

   @Override
   public void run() {

      for (int i = 0; i < numTerminos; i++) {

         if (recurso.numValores < recurso.maxValores) {
            this.numeroGenerado = (Math.random() * 100);//Aqui genero el número que el hilo va a guardar en el vector
            textoGen(this, this.numeroGenerado);//llamo al metodo que sincroniza el String

            this.textoHilo = ("\nEl hilo " + this.nombreGenerador + " ha generado el valor: " + this.numeroGenerado).concat(textoHilo);
            textoId(this, this.numeroGenerado);//llamo al metodo que sincroniza el String
         }
         /*
         La sincronización me asegura que cuando entra un hilo (el que sea que entre primero), el recurso queda bloqueado para cualquier otro hilo que quiera entrar.
          */
         synchronized (recurso) {
            /*
            con el recurso sincronizado, el hilo inserta el valor dentro del recurso (el array estático). Solo el hilo que tiene acceso recurso puede insertar el valor y después, tendrá que liberarlo
             */

            recurso.insertarValor(numeroGenerado);

         }

         try {
            Thread.sleep(this.tiempo);// * 1000);//indico el tiempo que se tarda en generar el siguiente numero
         } catch (InterruptedException ex) {
            Logger.getLogger(GeneradorThread.class.getName()).log(Level.SEVERE, null, ex);
         }
      }
   }
}
```

```java
package tarea2;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Daniel González Fernández (33.374.9888-W)
 */
public class Tarea2 {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {

//      NumerosCompartidos recurso = new NumerosCompartidos(20);//USAR SOLO DENTRO DE NETBEANS
      NumerosCompartidos recurso = new NumerosCompartidos(Integer.parseInt(args[0]));//recibe parametro a traves de la consola msdos
      File fichero = new File("Operaciones.txt");
      Scanner hilo = null;
      ArrayList<GeneradorThread> listaHilos = new ArrayList<GeneradorThread>();//este arraylist va a guardar los hilos que quiero crear

      try {
         hilo = new Scanner(fichero);

         /*
         Este bucle while nos va a crear tantos hilos como filas tenga el
         fichero "Operaciones.txt". Vamos a ir sacando fila a fila para despues
         extraer los parametros con los que vamos a instanciar los hilos.
          */
         while (hilo.hasNext()) {
            String hiloLeido = hilo.nextLine();
            String[] paramSeparados = hiloLeido.split(";");//separo los parametros de cada fila

            String param1 = paramSeparados[0];//nombre del hilo
            int param2 = Integer.valueOf(paramSeparados[1]);//cuantos numeros va a generar el hilo
            int param3 = Integer.valueOf(paramSeparados[2]);//segundos que espera el hilo para generar un nuevo número
            listaHilos.add((new GeneradorThread(param1, param2, param3, recurso)));// añado al arraylist los hilos con los parametros necesarios
         }

         /*
```

```
        En este bucle FOR voy a lanzar los hilos que he creado y estan
        guardados en el ArrayList
         */
        for (int i = 0; i < listaHilos.size(); i++) {
            listaHilos.get(i).start();// recorro el arraylist dondeestan los hilos para iniciarlos

        }

    } catch (FileNotFoundException ex) {
        Logger.getLogger(Tarea2.class.getName()).log(Level.SEVERE, null, ex);
    }

    /*
    Este bucle FOR espera a que todos los hilos hayan muerto.
     */
    for (int i = 0; i < listaHilos.size(); i++) {
        try {
            listaHilos.get(i).join();//Espero a que todos los hilos mueran

        } catch (InterruptedException ex) {
            Logger.getLogger(Tarea2.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /*
    Aqui empiezo a imprimir los resultados por pantalla
     */
    System.out.println("\nEJECUCIÓN DE LOS HILOS GENERADORES DE VALORES");
    System.out.println("-----------------------------------------");
    System.out.println(GeneradorThread.textoGenerado);
    if (!NumerosCompartidos.sigo) {
        System.out.println("El generador " + " no puede generar mas valores");
    }

    System.out.println("\nResumen final del elementos generados");
    System.out.println("-------------------------------------");
    for (int i = 0; i < listaHilos.size(); i++) {
        System.out.println("\n" + listaHilos.get(i).textoHilo);
    }

    System.out.println("\nEstado del recurso compartido");
    System.out.println("----------------------------");
    System.out.print("[");
    for (int i = 0; i < recurso.numValores - 1; i++) {
        System.out.println(recurso.valores[i] + (", "));

    }
    System.out.print(recurso.valores[recurso.numValores - 1]);
    System.out.println("]");

    System.out.println("\nEJECUCIÓN DE LOS DOS HILOS QUE REALIZAN LA SUMATORIA");
    System.out.println("---------------------------------------------------");
```

```
        System.out.println("(Id_hilo): (Elemento)");
        System.out.println(GeneradorThread.textoHiloId);

        System.out.println("\nRESUMEN DEL EJERCICIO");
        System.out.println("--------------------\n");
        System.out.println("Estado del recurso compartido");
        System.out.println("-----------------------------");
        System.out.print("[");
        for (int i = 0; i < recurso.numValores - 1; i++) {
            System.out.println(recurso.valores[i] + (", "));
        }
        System.out.print(recurso.valores[recurso.numValores - 1]);
        System.out.println("]");;

        /*
        Aqui comienzan la segunda parte, la de los hilos que suman.
        COmo sé que voy a usar dos hilos que van a sumar, los instancio y arranco
         */
        SumatoriaThread sumatorio1 = new SumatoriaThread(recurso);//creo el hilo1
        SumatoriaThread sumatorio2 = new SumatoriaThread(recurso);//creo el hilo2

        sumatorio1.start();//arranco el hilo1
        sumatorio2.start();//arrando el hilo2

        try {
            sumatorio1.join();//espero a que el hilo1 muera
            sumatorio2.join();//espero a que el hilo2 muera
        } catch (InterruptedException ex) {
            Logger.getLogger(Tarea2.class.getName()).log(Level.SEVERE, null, ex);
        }

        /*
        Aqui muestro los resultados de la sumatoria
         */
        System.out.println("\nDatos de la sumatoria");
        System.out.println("---------------------");
        System.out.println("Sumatoria: "
                + (sumatorio1.sumatoria + sumatorio2.sumatoria)
                + " ("
                + sumatorio1.sumatoria
                + ", "
                + sumatorio2.sumatoria
                + ")");
    }
}
```

```java
package tarea2;

public class NumerosCompartidos {

   int maxValores;//numero de valores máximos que se pueden guardar en el vector
   double valores[] = null; //valor metido en el vector
   int numValores;//numero de valores efectivamente metidos en el vector
   static boolean sigo = true;// controla que el array esta lleno

   public NumerosCompartidos(int maxValores) {
      this.maxValores = maxValores;
      this.numValores = 0;
      valores = new double[maxValores];
   }

   public int getMaxValores() { return maxValores; }
   public void setMaxValores(int maxValores) {this.maxValores = maxValores;}
   public double[] getValores() {return valores;}
   public void setValores(double[] valores) { this.valores = valores; }
   public int getNumValores() { return numValores; }
   public void setNumValores(int numValores) { this.numValores = numValores; }

   /*
         El método insertarValor inserta el double que se pasa por parámetro dentro del array
estático. Se comprueba que quepan los números que los hilos vayan generando dentro del
array. Cuando se llena el array, no se añaden mas números
    */
   public synchronized void insertarValor(double numeroGenerado) {
      if (numValores < maxValores) {
         valores[numValores] = numeroGenerado;
         numValores++;
      } else {
         sigo = false;
      }
   }

   @Override
   public String toString() {

      return "NumerosCompartidos{" + "maxValores=" + maxValores + ", valores=" +
valores[numValores] + ", numValores=" + numValores + '}';
   }
}
```