Aufgabe zum keyword synchronized

In der Vorlesung haben wir am Anfang angeschaut, wie man theoretisch mit Hilfe von Threads die Summe von einer gewissen Anzahl Integern berechnen könnte.

Erstelle ein Programm, welches die Summe von 100 Integern berechnet, in dem 4 Threads verwendet werden. Es sollen die Teilsummen und am Ende die ganze Summe ausgegeben werden. Dabei soll jeder Eintrag des Arrays zuerst *2 gerechnet werden (muss nicht gemacht werden, ist einfach in der Lösung so).

Tipps:

- Verwende einen Counter, der zählt wieviele Threads fertig sind.
- Überlege dir, wo du synchronized Blocks benötigst.
- Du kannst die Länge des Arrays auch variabel wählen.

Auf der nächsten Seite findest du meine kommentierte Lösung.

```java
public class Main {
	//Dieses Array wird unter den 4 Threads aufgeteilt.
	public static int[] array = new int[100];

	public static void main(String[] args) {
		for (int i = 0; i < array.length; i++) {
			array[i]=i;
		}
		//kreiere die Worker
		int k=array.length;
		Worker worker1 = new Worker(array, 0, (k/4)-1, 1);
		Worker worker2 = new Worker(array, (k/4), (k/2)-1,
		2);
		Worker worker3 = new Worker(array, k/2, (3*k/4)-1,
		3);
		Worker worker4 = new Worker(array, 3*k/4, k-1, 4);
		//Starte die Worker
		worker1.start();
		worker2.start();
		worker3.start();
		worker4.start();
		while(Worker.counter<4){
			//warten bis alle Worker fertig sind
		}
		System.out.println("Gesammtsumme: " + Worker.sum);
	}
}
```

Output:

```
Thread 3 ist fertig und hat berechnet: 3100
Thread 2 ist fertig und hat berechnet: 1850
Thread 4 ist fertig und hat berechnet: 4350
Thread 1 ist fertig und hat berechnet: 600
Gesammtsumme: 9900
```

Oder

```
Thread 1 ist fertig und hat berechnet: 600
Thread 3 ist fertig und hat berechnet: 3100
Thread 2 ist fertig und hat berechnet: 1850
Thread 4 ist fertig und hat berechnet: 4350
Gesammtsumme: 9900
```

```java
public class Worker extends Thread {
	//Attribute
	public static int sum=0; //Gesamtsumme
	private int[] array;	//Array mit den zu bearbeitenden
	Zahlen
	private int lower;	//Untergrenze des zu
	bearbeitenden Stücks
	private int upper;	//Obergrenze
	private int id;		//ID des Threads (für
	Output)
	public static int counter=0;  //Der Counter zählt die
	Threads, die fertig sind.

	//Konstruktor
	public Worker(int[] a, int l, int u, int id) {
		array=a;
		lower=l;
		upper=u;
		this.id = id;
	}

	//überschreiben der run Methode von Thread
	public void run() {
		int psum = 0;	//Teilsumme
		for (int i = lower; i <= upper; i++) {
			psum+=2*array[i];   //dasselbe wie sum= sum +
			(2*array[i]);
		}
		System.out.println("Thread " + id + " ist fertig und
		hat berechnet: " + psum);
		/*Dieser Teil ist heikel, es könnte sein, dass
		mehrere Threads zur selben Zeit auf sum zugreifen
		möchten. Darum müssen wir diesen Block
		synchronisieren.*/
		synchronized (Worker.class) {
			sum+=psum;
			counter++;
		}
	}
}
```