

Started on	Tuesday, 21 May 2024, 11:43 AM
State	Finished
Completed on	Wednesday, 5 June 2024, 7:03 AM
Time taken	14 days 19 hours
Grade	300.00 out of 300.00 (100%)

Question **1**
Correct
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Concurrency - Delayed Output

Lengkapi metode static printDelayed yang mengeluarkan suatu string setelah delay sekian milidetik. Metode printDelayed harus bersifat non-blocking, artinya metode harus bisa dieksekusi lagi, walaupun string yang diekspetasikan belum di print.

Misalnya metode printDelayed dipanggil seperti ini:

```
DelayedOutput.printDelayed(200, "Halo Obus");
DelayedOutput.printDelayed(100, "Halo Obus 2");
```

Maka output akan seperti ini:

Halo Obus 2
Halo Obus

Lengkapi dan submit file [DelayedOutput.java](#)

Java 8 ▾

 [DelayedOutput.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.21 sec, 35.50 MB
2	25	Accepted	0.31 sec, 33.92 MB
3	25	Accepted	0.31 sec, 35.35 MB
4	25	Accepted	0.61 sec, 33.42 MB

Question **2**
Correct
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Multithread Array Sum

Terdapat sebuah array berisi bilangan bulat. Implementasikan sebuah program untuk menjumlahkan seluruh elemen di dalam array dengan menggunakan java Multithreading.

- Kelas ArraySum memiliki atribut **nWorkers** yang menyatakan jumlah maksimum threads yang tersedia dan **array** untuk menampung array masukan
- Kelas ArraySum memiliki method **sum()** yang akan membuat sejumlah thread dan memetakan array masukan secara merata ke semua threads yang dapat dibuat. Contohnya:
...

```
array = [1, 2, 3, 4, 5]
nWorkers = 2
```


maka:

```
Thread-0 <- [1, 2]
Thread-1 <- [3, 4, 5]
```


...

masing-masing thread kemudian memanggil method `partialSum(int start, int end)` untuk mendapatkan hasil penjumlahan elemen-elemen-nya
Perhatian:
Jangan membuat kelas baru untuk membuat suatu thread. Gunakanlah deklarasi in-line kelas, seperti berikut
...


```
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
    }
});
```


...

- Kelas ArraySum memiliki method **partialSum(int start, int end)** akan melakukan penjumlahan elemen-elemen array pada index **start** sampai **end-1**

Lengkapilah file **ArraySum.java** pada file [zip](#) ini. Agar tidak membebani server Olympia, gunakanlah file **ArraySumDriver.java** untuk pengujian kompilasi dan solusi awal Anda. Pengujian sebenarnya tetap dengan submit ke Olympia. Kumpulkan file **ArraySum.java** yang telah diimplementasikan.

Java 8 ▾

 [ArraySum.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.08 sec, 29.03 MB
2	20	Accepted	0.08 sec, 33.71 MB
3	20	Accepted	0.07 sec, 29.42 MB

No	Score	Verdict	Description
4	20	Accepted	0.07 sec, 28.04 MB
5	20	Accepted	0.07 sec, 27.95 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Word Count Server

****Problem Statement****

Pada kuliah pengembangan aplikasi sistem terdistribusi, Anda akan belajar beberapa jenis web server. Beberapa yang populer yaitu event-driven and thread-based server system. Contoh event-driven webserver yaitu NodeJS dan thread-based yaitu Apache webserver. Pada thread-based webserver setiap request yang datang dari client akan dibuatkan thread baru dan pemrosesan request dilakukan pada thread tersebut.

Disini anda ditugaskan untuk mensimulasikan dan mengimplementasikan simple thread-based webserver yang akan menghitung jumlah kata yang diberikan oleh client.

Server akan menerima kalimat-kalimat yang perlu dihitung jumlah katanya. Kemudian, untuk setiap worker thread yang tersedia pada server akan memproses setiap kalimat tersebut dan pada akhirnya main thread akan mengembalikan jumlah kata pada setiap kalimat.

****Low Level Design****

- Terdapat kelas `Server` memiliki variabel `workers` yang menyatakan maksimum jumlah threads yang tersedia pada server (disebut juga sebagai thread pool).
- Method `process(String[] requests)` akan memetakan setiap elemen pada array requests ke suatu thread secara rata terlebih dahulu. Jika jumlah thread yang tersedia kurang dari jumlah elemen `requests` maka akan terdapat satu atau lebih thread yang mendapatkan elemen lebih. Contohnya jika `workers = 3` dan `requests.length = 4` maka pemetaan elemen:

```
...
Thread-0 <- requests[0]
Thread-1 <- requests[1]
Thread-2 <- requests[2]
Thread-0 <- requests[3]
...
```

Terlihat bahwa Thread-0 akan mendapatkan 2 elemen.

Perhatian:

Jangan membuat kelas baru untuk membuat suatu worker thread. Gunakanlah deklarasi in-line kelas, seperti berikut

```
...
Thread thread = new Thread(new Runnable() {
    @Override
    public void run() {
    }
});
...
```

- Method `count(String request)` akan menghitung jumlah kata pada kalimat `request`. Method ini akan dipanggil oleh setiap worker thread.

Files

- Lengkapilah file [`Server.java`](#)
- Agar tidak membebani server Olympia, gunakanlah file [`ServerDriver.java`](#) untuk pengujian kompilasi dan solusi awal Anda. Pengujian sebenarnya tetap dengan submit ke Olympia.

Submit kembali file `Server.java` yang telah berisi jawaban Anda.

Java 8 ▾

 [Server.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12.5	Accepted	0.08 sec, 28.82 MB
2	12.5	Accepted	0.07 sec, 28.89 MB
3	12.5	Accepted	0.09 sec, 31.63 MB
4	12.5	Accepted	0.09 sec, 31.54 MB
5	12.5	Accepted	0.10 sec, 31.85 MB
6	12.5	Accepted	0.11 sec, 31.64 MB
7	12.5	Accepted	0.58 sec, 30.86 MB
8	12.5	Accepted	0.92 sec, 33.42 MB

[◀ Praktikum 7](#)

Jump to...

◀ ▶

[Praktikum 8 ▶](#)