Started on	Thursday, 21 April 2022, 11:08 AM
State	Finished
Completed on	Thursday, 21 April 2022, 12:49 PM
Time taken	1 hour 41 mins
Marks	400.00/400.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Dr. Michel Fang, S.T. merupakan seorang botanis di Kota Gotham. Ia membantu istrinya, Dr. Pamela Lillian Isley dalam meneliti bunga anggrek yang ada di sekitar Kota Gotham. Ia ingin membuat program yang menerima masukan lebar kelopak, panjang kelopak, serta tinggi batang dari sebuah bunga anggrek dan menghasilkan keluaran jenis dari bunga anggrek tersebut.

Ia mengambil sampel data ribuan jenis anggrek di sekitar Kota Gotham, yang masing-masing jenis diambil ratus-ribuan sampel untuk dikumpulkan. Ia berencana untuk mengambil rata-rata lebar kelopak, panjang kelopak, dan tinggi batang setiap jenis anggrek menjadi satu vektor tiga dimensi. Jadi, data masukan program akan dicocokan ke jenis anggrek yang memiliki jarak euclidean paling kecil ke rata-rata data setiap jenis anggrek yang telah dikumpulkan dalam bentuk *comma separated values* (csv).

Namun apabila program Dr. Michel Fang, S.T. menghitung rata-rata jutaan data secara serial, maka akan memakan waktu yang besar. Oleh karena itu, ia membuat program yang mengaplikasikan multithreading dalam menghitung rata-rata data setiap jenis anggrek. Apabila ada kesalahan pembacaan berkas, maka rata-rata setiap aspek dibuat menjadi `-1` untuk mengindikasikan kesalahan. Setiap jenis anggrek akan ditangani oleh satu thread, sehingga komputasi berlangsung paralel.

Bantu Dr. Michel Fang, S.T. dalam membuat program tersebut. Dalam program yang dibuat, terdapat:

- Kelas OrchidMeanDatum untuk menyimpan rata-rata aspek data suatu jenis anggrek
- Kelas CSVReader untuk membaca berkas eksternal
- Kelas OrchidDataAvgProcessor untuk menghitung rata-rata dari pembacaan berkas setiap jenis anggrek
- Program `Main` **kasaran** dengan bentuk seperti ini

Lengkapi kelas OrchidDataAvgProcessor yang mengimplementasikan *interface* `Runnable`.
Kumpulkan **OrchidDataAvgProcessor.java**.

Java 8

OrchidDataAvgProcessor.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	14	Accepted	0.75 sec, 27.89 MB
2	14	Accepted	0.53 sec, 26.34 MB
3	14	Accepted	0.31 sec, 28.79 MB
4	14	Accepted	0.87 sec, 27.93 MB
5	14	Accepted	0.94 sec, 27.95 MB
6	14	Accepted	0.71 sec, 27.98 MB
7	16	Accepted	0.72 sec, 29.92 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Seharusnya kalian tidak asing lagi dengan NIM Finder ITB. Mulai dari yang dibuat oleh Napoleon hingga NIM finder yang digeprek, ada banyak variasi NIM Finder yang beredar di ITB. Pertanyaannya; bagaimana caranya memperoleh data untuk NIM Finder?

Di ITB, ada suatu website yaitu nic.itb.ac.id. Dahulu kala, website ini tidak dibatasi untuk jumlah *request* yang bisa dilakukan oleh seorang user. Bayangkan, untuk mengambil 1 data mahasiswa diperlukan paling tidak 500ms (bergantung beberapa faktor). Bagaimana caranya mengambil data mahasiswa dari berbagai angkatan dan jurusan?

Untuk mengatasi hal tersebut, dilakukanlah *multithreading*. Dengan *multithreading*, *request* bisa dilakukan secara paralel. Tentunya, tidak asal membuat N thread untuk N mahasiswa; sebaiknya akan dilakukan *batching* karena thread mengonsumsi *resource* dalam jumlah yang cukup banyak. Biasanya akan dibuat beberapa thread saja.

Implementasikan `ScraperThread.java` yang mengimplementasikan apa yang dijalankan oleh tiap thread yang menangani suatu batch. Baca juga file-file lainnya untuk memahami konteks permasalahan; pahami bagaimana thread dibuat, dipanggil, dan dihentikan.

- `Scraper.java` - kelas yang mengemulasikan pembuatan thread untuk mock web scraping
- `Website.java` - mock website yang memberi informasi mahasiswa
- `ThreadListener.java` - interface yang digunakan pada oleh kelas yang listen ke thread
- `Student.java` - class yang menampung informasi mahasiswa

Java 8

ScraperThread.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	33	Accepted	0.41 sec, 31.12 MB
2	33	Accepted	0.70 sec, 37.73 MB
3	34	Accepted	0.60 sec, 41.08 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Dalam game online, tidak jarang pemain login lalu idle (tidak melakukan apa-apa) untuk waktu lama. Anda diminta membuat game yang jika pemain idle untuk waktu yang cukup lama, pemain akan otomatis logout. Lengkapi kelas Player berikut.

Kumpulkan **Player.java**

Java 8

Player.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.16 sec, 28.34 MB
2	25	Accepted	0.25 sec, 28.54 MB
3	25	Accepted	0.57 sec, 27.88 MB
4	25	Accepted	0.34 sec, 27.97 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Anda diminta membuat sebuah website pemilu untuk sebuah himpunan di ITB. Website Pemilu yang Anda buat kurang lebih seperti ini. Cobalah jalankan kode tersebut dengan kode Main berikut.

```
import java.util.ArrayList;
import java.util.List;

public class Main {
 public static void main(String[] args) {
   int nVotes = 10000;
   List<Thread> threads = new ArrayList<Thread>();
   WebPemilu web = new WebPemilu();

   long startTime = System.nanoTime();
   for (int i = 0; i < nVotes; i++) {
     Thread t = web.receiveVote("candidate-1");
     if (t != null) {
       threads.add(t);
     }
   }
   long endTime = System.nanoTime();
   long durationInNanoSeconds = (endTime - startTime);
   System.out.println("waktu: " + (durationInNanoSeconds / (1000 * 1000)) + " milisekon");
   System.out.flush();

   // memastikan semua vote sudah masuk
   for (Thread t : threads) {
     t.join();
   }
   web.printResult();
 }
}
```

Seharusnya, waktu yang dibutuhkan cukup lama. Silakan ubah variabel `nVotes` hingga waktu yang dibutuhkan sekitar 5.000 milisekon (5 detik), supaya perbedaan nantinya dapat terlihat.

Dalam pembuatan website, hal ini sangat buruk. Karena saat satu vote diterima, vote lain harus menunggu vote ini selesai diproses. Untuk itu, perlu dibuat perubahan:

- Buatlah supaya verifikasi dan `addVote` dipanggil didalam sebuah thread, sehingga pemanggilan method tidak membutuhkan waktu lama, tapi proses voting dikerjakan sebagai background task.
- Thread yang dibuat untuk melakukan verifikasi dan `addVote` di-return, supaya Main program dapat menunggu semua vote masuk sebelum memanggil `printResult()`
- Jalankan di program Anda, dan bandingkan hasilnya! Apakah sudah lebih cepat (kalau tidak, kemungkinan ada yang salah...)
- Perhatikan hasil yang dituliskan di layar. **Pastikan vote candidate-1 sesuai dengan jumlah nVote**. Jika tidak, kenapa? Apa yang salah? Benarkan dalam kode Anda. (Hint: perhatikan method `addVote`)

Perhatikan kalau:

- Tidak boleh membuat variabel atau method tambahan di kelas ini. Jika dibuat, maka seluruh testcase akan mengembalikan wrong answer
- Method verify dan printResult tidak perlu diubah (jika diubah, grader akan mengembalikan seperti semula)

Kumpulkan **WebPemilu.java** yang sudah dimodifikasi

Java 8

WebPemilu.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	50	Accepted	0.07 sec, 27.98 MB
2	50	Accepted	0.07 sec, 30.39 MB

◄ Slide Minggu 14

Jump to...