Started on	Sunday, 25 May 2025, 9:17 PM
State	Finished
Completed on	Sunday, 25 May 2025, 10:50 PM
Time taken	1 hour 33 mins
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

Kobopedia

Kamu sedang Doomscrolling di Instagram, tiba-tiba kamu ketemu video AI antara Kobo dan Zeta.

"Zeta~ Zeta~ cara bikin Tokopedia itu gimana sih?"
"Ohh~ gini Kobo, pertama coba kamu bikin fitur sorting duluu."
"Gimana caranya Zeta?"
"Nah, kamu coba buat dulu nih comparator di Java..."

Terinspirasi dari reels tersebut, kamu ingin mencoba untuk membuat Kobopedia menggunakan collections! Diberikan file **Kobopedia.java** untuk membuat hal tersebut.

Submit kembali Kobopedia.java

Java 8

Kobopedia.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.06 sec, 28.66 MB
2	10	Accepted	0.07 sec, 29.02 MB
3	10	Accepted	0.07 sec, 27.83 MB
4	10	Accepted	0.07 sec, 28.20 MB
5	10	Accepted	0.07 sec, 28.96 MB

No	Score	Verdict	Description
6	10	Accepted	0.06 sec, 30.35 MB
7	10	Accepted	0.07 sec, 28.77 MB
8	10	Accepted	0.07 sec, 28.72 MB
9	10	Accepted	0.06 sec, 28.31 MB
10	10	Accepted	0.07 sec, 27.98 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Java Collection - TreeSet

Set adalah sebuah struktur data yang memiliki kemampuan untuk menyimpan elemen-elemen unik. **Set** merupakan implementasi himpunan terbatas pada matematika. **Set** sebagai struktur data memiliki berbagai cara untuk diimplementasi, salah satu implementasi **Set** dalam bahasa Java adalah **TreeSet**.

TreeSet merupakan implementasi **Set** yang juga menyimpan informasi urutan dari elemen-elemen yang dimiliki. Karena urutan yang dipertahankan, pengambilan nilai minimal dan maksimal pada **TreeSet** memiliki kompleksitas waktu O(1).

Buatlah sebuah program **Main.java** yang memanfaatkan **TreeSet** dari **Collection Java** untuk memenuhi spesifikasi program sebagai berikut:

- Program menerima input sebagai berikut:
  - Pada baris pertama, berisi bilangan bulat positif **Q** (**1 ≤ Q ≤ 100**) menyatakan banyaknya operasi yang akan dilakukan
  - **Q** baris berikutnya berisi salah satu dari operasi berikut
    - `add X` → Masukkan bilangan bulat `X` kedalam **TreeSet**
    - `remove X`
      - Apabila terdapat bilangan bulat `X` pada **TreeSet**, maka hapus element tersebut
      - Apabila tidak ada, keluarkan pada layar `"Element {X} is not in The TreeSet"` (Tanpa tanda petik)
    - `first`
      - Mengeluarkan nilai bilangan minimal dari **TreeSet** ke layar
      - Apabila **TreeSet** kosong, keluarkan pada layar `"EMPTY"` (Tanpa tanda petik)
    - `last`
      - Mengeluarkan nilai bilangan maksimal dari **TreeSet** ke layar
      - Apabila **TreeSet** kosong, keluarkan pada layar `"EMPTY"` (Tanpa tanda petik)
- Program mengeluarkan output sebagai berikut
  - Untuk setiap operasi `remove`, `first`, dan `last` keluarkan sesuai yang diharapkan.

Contoh Input

```
8
add 10
remove 9
remove 10
add 5
add 7
add 8
first
last
```

Contoh Output

```
Element 9 is not in The TreeSet
5
8
```

Submit file **Main.java**.

Hints:

- Anda dapat menggunakan **TreeSet** sebagai berikut, `TreeSet<Integer> TreeSet = new TreeSet<Integer>()`
- Untuk melakukan operasi yang dibutuhkan, gunakan *method* yang ada pada format input.
- Untuk melihat apakah terdapat element `X` pada **TreeSet** gunakan *method* `contains(X)` pada **TreeSet**.

Java 8

Main.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.27 sec, 27.96 MB
2	10	Accepted	0.33 sec, 28.84 MB
3	10	Accepted	0.34 sec, 28.32 MB
4	10	Accepted	0.32 sec, 29.50 MB
5	10	Accepted	0.34 sec, 28.88 MB
6	10	Accepted	0.30 sec, 28.18 MB
7	10	Accepted	0.35 sec, 30.68 MB
8	10	Accepted	0.14 sec, 28.38 MB
9	10	Accepted	0.09 sec, 28.40 MB
10	10	Accepted	0.42 sec, 28.60 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Student Sorter

Buatlah sebuah program Java yang mengelola informasi mengenai siswa dan nilai-nilai mereka dalam beberapa mata pelajaran. Program ini akan menghitung nilai rata-rata (GPA) dari setiap siswa dan mengurutkan mereka berdasarkan GPA.

Program harus memenuhi persyaratan berikut:

Dalam kelas StudentSorter:

- Terdapat kelas Student dengan atribut name (String) dan courseGrades (Map<String, Integer>). Kelas Student memiliki method `addCourseGrade` untuk menambahkan nilai mata pelajaran dan `getGPA` menghitung GPA.
- Terdapat kelas StudentComparator yang mengimplementasikan Comparator. Kelas ini harus mengurutkan objek Student berdasarkan GPA. Kelas ini meng-override method `compare` yang melakukan komparasi siswa s1 dan s2. Cara kerja komparasi adalah jika s1 < s2 method ini akan mengembalikan bilangan negatif, nol jika sama dan positif jika s1 > s2. Bilangan tersebut bebas.
- Terdapat static method `sortStudentsByGPA` yang mengembalikan list objek Student yang sudah diurutkan dengan StudentComparator

Lengkapilah file `StudentSorter.java`.

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

Java 8

StudentSorter.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.06 sec, 28.73 MB
2	20	Accepted	0.07 sec, 27.78 MB
3	20	Accepted	0.06 sec, 28.91 MB
4	20	Accepted	0.06 sec, 28.95 MB
5	20	Accepted	0.07 sec, 28.99 MB

◄ Pasca Praktikum 7

Jump to...

Praktikum 8 ►