Question **4**

Tries remaining: 10

Marked out of 100.00

Time limit	1 s
Memory limit	64 MB

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** (Buatlah sendiri) 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

Maximum size for new files: 512MB, maximum attachments: 1

Files

You can drag and drop files here to add them.

Run Check

Question **5**

Tries remaining: 10

Marked out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program Java yang menerima dua baris `string` yang dipisahkan oleh spasi kemudian mengembalikan satu list baru yang merupakan elemen yang sama yang dimiliki oleh kedua baris string tersebut

Aturan masukan yang diterima:

- List input dapat berupa list kosong
- List input dapat mengandung kata yang duplikat

Asumsi:

- List input pasti berupa list string yang dipisahkan oleh spasi (contoh: saya senang praktikum oop)

Contoh:

Input:

apple banana cherry date elderberry
date elderberry fig grape honeydew

Output: [date, elderberry]

Silahkan lengkapi file Main.java

Hint:

- Gunakan konsep collection List dan Set
- Gunakan fungsi .split untuk memisahkan elemen dalam string
- Format output adalah format print ArrayList. Silahkan langsung print variabel bertipe ArrayList.
- Jika terdapat lebih dari satu elemen dengan nilai yang sama dari kedua string, maka cukup ambil salah satu saja
- Urutan output terurut berdasarkan elemen pertama yang memiliki nilai yang sama pada kedua baris string

Java 8

Maximum size for new files: 512MB, maximum attachments: 1

Files

You can drag and drop files here to add them.

Run Check

Question **6**

Tries remaining: 10

Marked out of 100.00

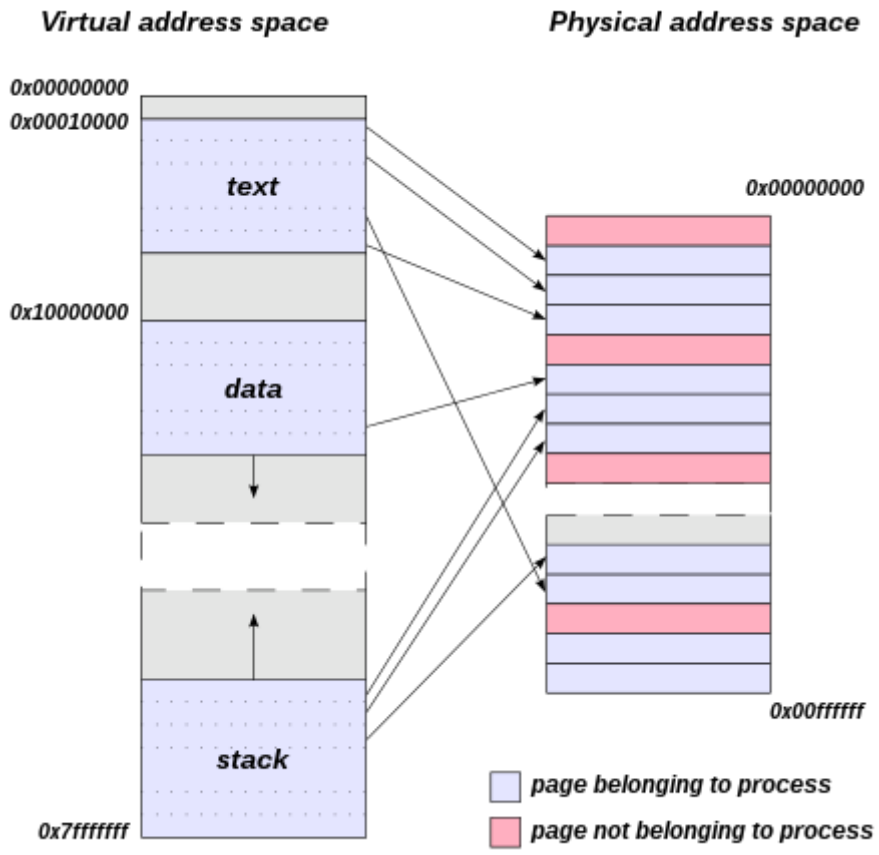
Time limit	1 s
Memory limit	64 MB

Page Replacement LRU

Latar Belakang

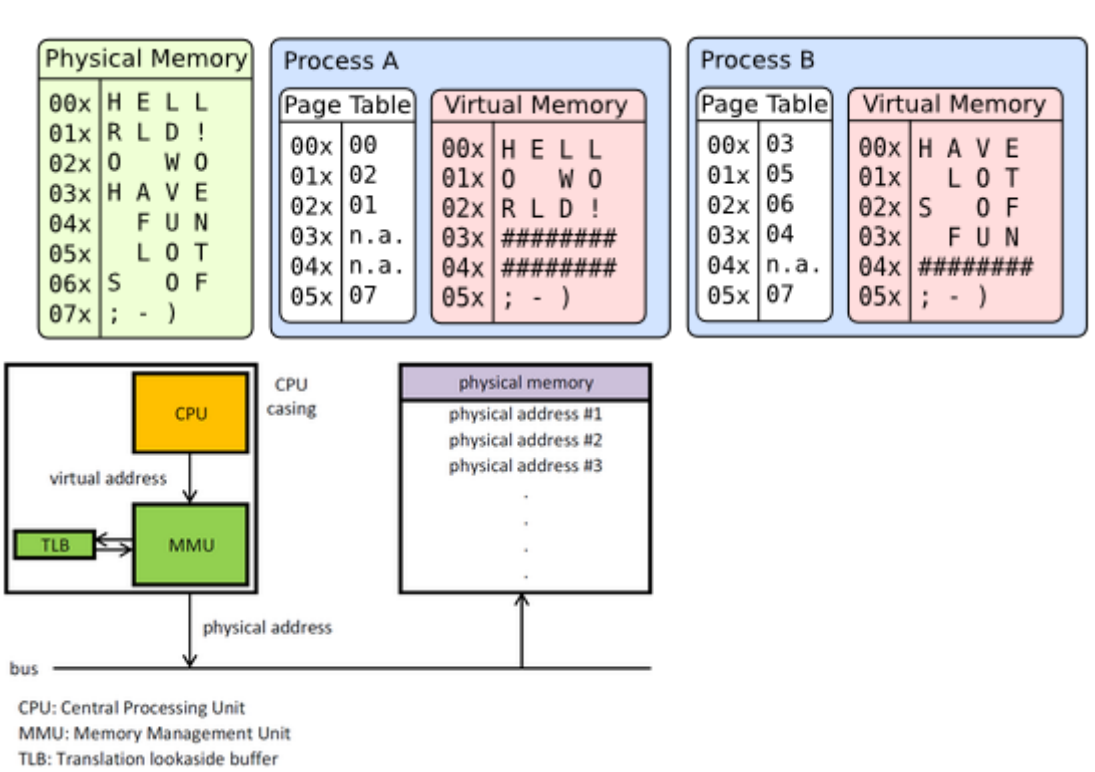
"Process" merupakan suatu program yang sedang berjalan. Pada OS Linux, setiap process mendapatkan "virtual memory address space" tersendiri. Maksudnya bagaimana? Misalnya pada PC dengan memory/RAM 4GB memiliki process Chrome dan VScode, dari kacamata masing-masing process tersebut, masing-masing "merasa" memiliki akses penuh RAM 4GB atau bisa lebih! Tetapi kenyataannya, hanya terdapat akses terbatas pada RAM. OS dan hardware bersama-sama membangun ilusi tersebut dengan beberapa mekanisme seperti mapping virtual address ke physical address, paging, swapping, etc.

"virtual memory" suatu process secara kasarnya akan dipotong-potong, potongan tersebut bernama "page" yang kemudian ditempatkan pada physical memory. Mekanisme tersebut dikenal dengan "paging", berikut ilustrasi paging suatu process



(Sumber: Wikipedia)

Disini tugas OS pada dasarnya melakukan "tracking" page-page untuk setiap process pada "page table", sehingga ketika ada permintaan untuk akses suatu alamat, MMU dapat melakukan mapping address dengan benar



(Sumber: OSdev, Wikipedia)

Seperti yang dikatakan sebelumnya, setiap proses memiliki virtual memori sendiri sehingga ada kemungkinan physical memori **tidak dapat menampung semua page processes**, saat kondisi ini terjadi OS umumnya perlu mekanisme/algoritma "**Page Replacement" contohnya LRU** agar page lain dapat masuk ke physical memori dengan me-replace page yang jarang diakses, page tersebut kemudian dipindahkan ke disk. Skema ini sering disebut dengan

"swapping". Kasus lain yang perlu diperhatikan adalah ketika suatu page ingin diakses tetapi page tersebut tidak ada di physical memori sehingga akan terjadi "page fault" . OS kemudian perlu memuat page tersebut dari disk ke memori lagi serta memperbarui "page table" process tersebut.

Problem Statement

Tugas Anda disini adalah mengimplementasikan algoritma "Page Replacement" LRU (Least Recently Used) dengan menggunakan HashMap dan Doubly linked list. HashMap bertugas untuk penyimpanan page dan linked list untuk tracking posisi page.

Detil Implementasi:

- Saat adanya penambahan page baru (addPage) dan ukuran HashMap/LinkedList = maxPages, page paling belakang pada LinkedList akan dihapus dan page baru ditempatkan di depan LinkedList. Jika ukurannya masih kurang dari maxPages page baru tetap ditempatkan paling depan tanpa adanya penghapusan
- Saat adanya pengaksesan suatu page (accessPage) dan page tersebut tidak ada di LinkedList, akan dikeluarkan pesan "page fault!" ke stdout lalu accessPage mengembalikan -1. Jika page yang diakses ada di LinkedList, ambil page tersebut di HashMap, pindahkan page tersebut ke depan LinkedList dan kembalikan physicalAddress

Agar lebih jelas, langsung saja dengan contoh:

Contoh kasus:
Misalnya maxPages = 3
// accessPage(<PID, pageID>, physicalAddress)
addPage(<3, 5>, 1)
addPage(<1, 2>, 4)
accessPage(<1, 2>)
addPage(<1, 3>, 7)
addPage(<2, 5>, 7)
accessPage(<3, 5>, 1)
accessPage(<2, 5>, 7)

Jika setiap kembalian dari fungsi accessPage di-outputkan:
4
page fault!
-1
7

State LinkedList setiap tahap akses:
saat accessPage(<1, 2>) :
(1,2) <-> (3,5) <-> null
saat accessPage(<3, 5>):
(2, 5) <-> (1, 3) <-> (1, 2)
karena page (3, 5) sudah tidak ada di memori maka akan dikeluarkan "page fault!" dan kembalian -1
saat accessPage(<2, 5>):
(2, 5) <-> (1, 3) <-> (1, 2)

Files

Lengkapilah file PageReplacementLRU.java, disediakan juga Pair.java dan PageReplacementLRUDriver.java untuk menjalankan dan mengecek kebenaran awal program Anda

Submit kembali file **PageReplacementLRU.java** yang telah berisi jawaban Anda.

Selamat bekerja :)

Java 8

Maximum size for new files: 512MB, maximum attachments: 1

Files

You can drag and drop files here to add them.

Run Check

◄ Feedback Praktikum 9

Jump to...