| | |
|---|---|
| **Started on** | Tuesday, 17 June 2025, 8:39 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 17 June 2025, 10:48 PM |
| **Time taken** | 2 hours 9 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 |

# Number Sum III

## Deskripsi

Anda diberikan suatu array statik bertipe `Number`, untuk setiap elemen di dalamnya Anda ditugaskan untuk menghitung hasil dari elemen-elemen dengan tipe data yang sama dengan melakukan penjumlahan, pengurangan, dan perkalian secara berurutan dan berulang.

Tipe data yang mungkin terdapat pada array tersebut: `Byte, Integer, Double, Float, Short, Long`

Untuk setiap data yang sama, lakukan penjumlahan, pengurangan, dan perkalian secara berurutan dan berulang hingga tipe data yang sama telah berakhir dalam array tersebut. Jika tipe data tersebut telah berakhir, lanjutkan ke tipe data berikutnya dengan mengulang dari penjumlahan kembali. Jika tipe data berikutnya tidak ada, isi dengan `null`. Jika suatu `number` dengan tipe data yang sama terpisah dari kawanannya, operasi tidak akan diulang.

Urutan operasi: `Penjumlahan, Pengurangan, Perkalian`.

**Tidak perlu melakukan print sama sekali.**

*L, f, b, dan S pada angka menunjukkan tipe data dari angka tersebut. Contoh: 2L adalah Long, 4.5f adalah Float, 5b adalah Byte, dan 2S adalah Short. Tidak akan muncul jika di print*

*Tidak perlu memerdulikan floating point precision issue*

## Contoh

## Contoh 1:

**Array masukan:**
[2L, 4.5f, 3.5, 45L, 100S, 5b, 2b] // [Long, Float, Double, Long, Short, Byte, Byte]

**Array keluaran:**
[47L, 4.5f, 3.5, 100S, 7b, null, null] // [Long, Float, Double, Short, Byte, null, null]

**Penjelasan:**
[(2L + 45L), (4.5f), (3.5), (100), (5+2)] [47L, 4.5f, 3.5, 100, 7, null, null] // [Long, Float, Double, Short, Byte, null, null]

Karena tersisa dua tempat kosong pada array keluaran, maka keduanya diisi dengan null

## Contoh 2:

**Array masukan:**
[2, 4, 3, 45, 100, 5, 2] // [Integer, Integer, Integer, Integer, Integer, Integer, Integer]

**Array keluaran:**
[460, null, null, null, null, null, null] // [Integer, null, null, null, null, null, null]

**Penjelasan:**
[((2 + 4 - 3) *45 + 100 - 5)* 2, null, null, null, null, null, null] Operasi dilakukan secara berurutan dan berulang untuk tipe data yang sama hingga tipe data tersebut habis. Karena tipe data habis, maka tempat kosong diisi dengan null

## Contoh 3:

**Array masukan:**
[1, 2, 2, 3, 4, 3.5, 4.5, 5.5, 5L, 6L, 7L, 7S, 8S, 8b, 9b, 10.5f, 11.5f, 12.5f, 5] // [Integer, Integer, Integer, Integer, Integer, Double, Double, Double, Long, Long, Long, Short, Short, Byte, Byte, Float, Float, Float, Integer]

**Array keluaran:**
[2, 2.5, 4L, 15S, 17b, 9.5f, null, null, null, null, null, null, null, null, null, null, null, null] // [Integer, Double, Long, Short, Byte, Float, null, null, null, null, null, null, null, null, null, null, null, null]

**Penjelasan:**
[((1 + 2 - 2) * 3 + 4 - 5), (3.5 + 4.5 - 5.5), (5 + 6 - 7), (7 + 8), (8 + 9), (10.5 + 11.5 - 12.5), null, null, null, null, null, null, null, null, null, null, null, null] -> [2, 2.5, 4L, 15S, 17b, 9.5f, null, null, null, null, null, null, null, null, null, null, null, null]

Perhatikan bahwa urutan tipe data array keluaran haruslah sama dengan array masukan. Perhatikan juga bahwa panjang array masukan haruslah sama dengan array keluaran. Jika terdapat tempat kosong setelah dilakukan operasi, tempat tersebut diisi dengan null

Hint:

- Untuk mengecek apakah dua objek merupakan kelas yang sama bisa dengan: `obj1.getClass().equal(obj2.getClass())`

- Untuk mengecek apakah suatu kelas merupakan subclass dari kelas lain bisa dengan keyword: `instanceof`

Lengkapilah metode-metode pada file `NumberSumIII.java`. Kemudian, Submit file `NumberSumIII.java`

Java 8

NumberSumIII.java

## Score: 100

## Blackbox

### Score: 100

### Verdict: Accepted

### Evaluator: Exact

| No | Score | Verdict | Description |
|---|---|---|---|
| 1 | 10 | Accepted | 0.06 sec, 28.53 MB |
| 2 | 10 | Accepted | 0.05 sec, 26.66 MB |
| 3 | 10 | Accepted | 0.06 sec, 28.70 MB |
| 4 | 10 | Accepted | 0.06 sec, 28.39 MB |
| 5 | 10 | Accepted | 0.06 sec, 28.16 MB |
| 6 | 10 | Accepted | 0.05 sec, 28.94 MB |
| 7 | 10 | Accepted | 0.05 sec, 29.14 MB |
| 8 | 10 | Accepted | 0.07 sec, 28.92 MB |
| 9 | 10 | Accepted | 0.05 sec, 28.05 MB |
| 10 | 10 | Accepted | 0.06 sec, 28.30 MB |

Question **2**

Correct

Mark 100.00 out of 100.00

| | |
|---|---|
| Time limit | 1 s |
| Memory limit | 64 MB |

# Torrent

Torrent adalah protokol berbagi file yang memungkinkan pengguna untuk mengunduh file dalam bentuk chunk dari berbagai sumber (peer) sekaligus. Anda diminta untuk membuat sistem unduhan torrent sederhana. Disediakan file TorrentPeer.java yang mengimplementasikan Peer pada torrent. lengkapi file TorrentDriver.java yang merupakan file yang memiliki method main. untuk menyederhanakan, isi dari masing-masing chunk cukup dengan "[Chunk-i]" dengan i menandakan nomor chunk.
implementasi dengan menggunakan input dan output sesuai dengan contoh berikut:

Input:

```
arsip_fufafufa.zip
4
2 3 1 4
5 9 6 7
1 8
7 5 2
```

Output:

```
Peer1 memiliki chunk: 2, 3, 1, 4
Peer2 memiliki chunk: 5, 9, 6, 7
Peer3 memiliki chunk: 1, 8
Peer4 memiliki chunk: 7, 5, 2
------------
Mengunduh chunk 1 dari Peer1...
Mengunduh chunk 2 dari Peer1...
Mengunduh chunk 3 dari Peer1...
Mengunduh chunk 4 dari Peer1...
Mengunduh chunk 5 dari Peer2...
Mengunduh chunk 6 dari Peer2...
Mengunduh chunk 7 dari Peer2...
Mengunduh chunk 8 dari Peer3...
Mengunduh chunk 9 dari Peer2...
------------
Semua chunk berhasil diunduh!
Menyusun kembali file...
File arsip_fufafufa.zip berhasil direkonstruksi dari 9 chunk.
Isi rekontruksi file: [Chunk-1][Chunk-2][Chunk-3][Chunk-4][Chunk-5][Chunk-6][Chunk-7][Chunk-8][Chunk-9]
```

**Baris pertama** merupakan masukan nama file
**Baris kedua** Merupakan masukan jumlah peer (n)
**Baris ketiga hingga ke-(3+n)** Merupakan masukan elemen masing-masing peer [1, n]

**Catatan:**

- Asumsi bahwa maksimal chunk unik yang dimiliki seluruh peer adalah 100 dan indeks chunk maksimal yang bisa dibuat yaitu 99.
- Petunjuk hal-hal yang harus dilakukan ditulis pada komentar di kode sumber
- Anda tidak perlu membuat implementasi TorrentPeer karena sudah disediakan pada TorrentPeer.java. Anda hanya perlu mengimplementasi TorrentDriver.java saja

Java 8

TorrentDriver.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

| No | Score | Verdict | Description |
|---|---|---|---|
| 1 | 10 | Accepted | 0.06 sec, 27.95 MB |
| 2 | 10 | Accepted | 0.06 sec, 28.08 MB |
| 3 | 10 | Accepted | 0.07 sec, 26.77 MB |
| 4 | 10 | Accepted | 0.06 sec, 28.77 MB |
| 5 | 10 | Accepted | 0.07 sec, 28.61 MB |
| 6 | 10 | Accepted | 0.07 sec, 30.84 MB |
| 7 | 10 | Accepted | 0.06 sec, 30.79 MB |
| 8 | 10 | Accepted | 0.06 sec, 28.48 MB |
| 9 | 10 | Accepted | 0.06 sec, 28.54 MB |
| 10 | 10 | Accepted | 0.06 sec, 28.43 MB |

Question **3**

Correct

Mark 100.00 out of 100.00

| | |
|---|---|
| Time limit | 1 s |
| Memory limit | 64 MB |

# Hexaesar

Pak Dengklek habis belajar kriptografi ngiranya bakal dapat ilmu buat main kripto. Ternyata dia kena scam (padahal belum invest) dan ternyata kriptografi tuh main matematika banget. Dia udah dikasih tugas sama dosen buat bikin program yang bisa enkripsi dan dekripsi pesan dengan sebuah metode bernama Hexaesar. Metode ini mirip Caesar Cipher but with a twist, yaitu make hexadecimal. Pak Dengklek panik karena dia ga suka matematika dan PRS udah ditutup jadi dia gabisa drop. Akhirnya dia minta tolong kamu buat bantuin bikin programnya. Ayo bantu Pak Dengklek!

Pada attachment, kamu diberikan 2 file, yaitu `Hexaesar.java` dan `Main.java`. Lengkapilah metode-metode pada file `Hexaesar.java`.

Selain itu, implementasikan metode `main` pada file `Main.java` yang menerima 3 masukan yaitu mode, pesan, dan key. Mode berisi sebuah integer 1 sampai 4 yang merepresentasikan 4 mode berikut:

1. Enkripsi String Hexadecimal dengan Hexaesar
2. Dekripsi String Hexadecimal dengan Hexaesar
3. Enkripsi Integer dengan Hexaesar
4. Dekripsi Integer dengan Hexaesar

Jika mode adalah 1 atau 2, maka pesan berisi sebuah string yang akan dienkripsi atau didekripsi. Jika mode adalah 3 atau 4, maka pesan berisi sebuah integer yang akan dienkripsi atau didekripsi.

Key berisi sebuah integer yang merepresentasikan jumlah pergeseran karakter dalam metode Hexaesar.

Diberikan juga file `HexCalculator.java` untuk membantu kamu dalam mengubah desimal ke hexadecimal dan sebaliknya.

## Attachments

Attachments: attachments.zip

## Notes

- Hexadecimal adalah sistem bilangan yang menggunakan 16 simbol, yaitu 0-9 dan a-f, dimana a-f merepresentasikan nilai 10-15. Contohnya adalah `a` merepresentasikan nilai 10, `b` merepresentasikan nilai 11, dan seterusnya.
- Caesar Cipher bekerja dengan menggeser setiap karakter dalam pesan sebesar `key` karakter. Jika `key` bernilai 1, maka karakter `a` akan menjadi `b`, karakter `b` akan menjadi `c`, dan seterusnya. Contohnya adalah `abc` dengan `key` 1 akan menjadi `bcd`.
- Dekripsi adalah proses membalikkan enkripsi. Jika pesan `bcd` didekripsi dengan `key` 1, maka hasilnya adalah `abc`. Jika pesan `abc` dienkripsi dengan `key` 1, maka hasilnya adalah `bcd`.

## Contoh Masukan dan Keluaran

**Contoh 1**

Masukan: 1 abc 1

Keluaran: bcd

**Contoh 2**

Masukan: 2 bcd 1

Keluaran: abc

**Contoh 3**

Masukan: 3 10 1

Keluaran: 11

**Contoh 4**

Masukan: 4 11 1

Keluaran: 10

Java 8

3.zip

Score: 100

## Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

| No | Score | Verdict | Description |
|---|---|---|---|
| 1 | 10 | Accepted | 0.06 sec, 28.55 MB |
| 2 | 10 | Accepted | 0.06 sec, 28.36 MB |
| 3 | 10 | Accepted | 0.06 sec, 28.91 MB |
| 4 | 10 | Accepted | 0.06 sec, 30.25 MB |
| 5 | 10 | Accepted | 0.06 sec, 28.89 MB |
| 6 | 10 | Accepted | 0.06 sec, 27.82 MB |
| 7 | 10 | Accepted | 0.06 sec, 28.46 MB |
| 8 | 10 | Accepted | 0.06 sec, 28.89 MB |
| 9 | 10 | Accepted | 0.07 sec, 29.39 MB |
| 10 | 10 | Accepted | 0.06 sec, 30.20 MB |

◄ Feedback Form

Jump to...

Pra-kuis 1 ►