

[Dashboard](#) / [My courses](#) / [ITB IF2212 2 2324](#) / [Praktikum 6: Exception](#) / [Praktikum 6 - Latihan](#)

<b>Started on</b>	Wednesday, 5 June 2024, 4:58 AM
<b>State</b>	Finished
<b>Completed on</b>	Wednesday, 5 June 2024, 5:18 AM
<b>Time taken</b>	20 mins 13 secs
<b>Grade</b>	<b>300.00</b> out of 300.00 ( <b>100%</b> )

Question **1**  
Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Menggunakan [Email.java](#), buatlah program utama yang menerima sebuah string email lalu memvalidasinya.

Format input:

- 1. Berisi string email lengkap *instance* `String` langsung.

Format output:

- 1. Apabila validasi email berhasil, maka cetak boolean hasil validasi.
- 2. Apabila validasi email gagal/program karena *exception*, maka cetak pesan dalam format `<<nama exception>!` diikuti dengan isi message exception tersebut.
  - 1. Program bisa gagal di tahap apapun, tidak hanya pada tahap validasi email saja.
  - 2. Apabila `exception` yang dikeluarkan merupakan instance *Exception* selain `InvalidEmailException` ataupun `InvalidDomainException`, maka cetak nama kelas exception yang dipanggil (bisa dengan getName() saja).
- 3. Berdasarkan hasil validasi email:
  - 1. Apabila email valid maka cetak `Email validated.` pada baris yang berbeda dari output sebelumnya ke layar.
  - 2. Apabila email tidak valid, maka cetak `Email string error!` pada baris yang berbeda dari output sebelumnya ke layar.
- 4. Setelah validasi email selesai (apapun hasilnya), tutup scanner yang dibuka dan cetak `Operation finished.` pada baris yang berbeda dari output sebelumnya ke layar. Penutupan scanner bisa dilakukan dengan pemanggilan method `close()` dari scanner.

Contoh:

Input	Output
praktikum@oop.com	true Email validated. Operation finished.
oop.com	InvalidEmailException! Email harus mengandung tepat satu buah @ Email string error! Operation finished.

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

Java 8 ▾

 [Main.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.06 sec, 28.03 MB
2	20	Accepted	0.06 sec, 28.54 MB
3	20	Accepted	0.06 sec, 29.25 MB
4	20	Accepted	0.06 sec, 26.90 MB

No	Score	Verdict	Description
5	20	Accepted	0.06 sec, 28.05 MB

Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program kalkulator yang bisa menerima kalkulasi sederhana. Meskipun sederhana, kalkulator akan menerapkan konsep Exception dimana kalkulator akan mengeluarkan pesan error ketika input tidak benar.

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

File tersebut akan memiliki 4 buah kelas, yaitu:

- 1. Kelas `Calculator`
- 2. Kelas `CalculatorException` yang mengextend kelas `Exception`
- 3. Kelas `InvalidOperationException` yang mengextend kelas `CalculatorException`
- 4. Kelas `InvalidDivisionException` yang mengextend kelas `CalculatorException`

Kelas `Calculator.java` memiliki method `calculate` yang mengembalikan `double`. Method tersebut akan menerima 3 buah parameter, yaitu `a` bertipe `double` yang berupa masukan pertama, `b` bertipe `double` yang berupa masukan kedua, dan `operation` bertipe `char` yang berupa operator yang akan dieksekusi oleh kalkulator.

Pertama, Kalkulator akan memeriksa jenis operasi yang dimasukkan pada parameter. Apabila operasi yang diinput pengguna bukan `+`, `-`, `*`, atau `/`, maka kalkulator akan mengembalikan pesan error dari kelas `InvalidOperationException` yang berisi pesan dengan format `Invalid Operation: <operasi yang dimasukan>`. Contoh: `Invalid Operation: a` ketika kalkulator menerima 'a' sebagai masukan operasi.

Apabila kalkulator menerima input pembagian terhadap 0, maka kalkulator akan mengembalikan pesan error dari kelas `InvalidDivisionException` yang berisi pesan dengan format `Invalid Division: Tidak dapat melakukan pembagian terhadap 0!`.

Java 8 ▾

 [Calculator.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12.5	Accepted	0.06 sec, 28.17 MB
2	12.5	Accepted	0.06 sec, 29.11 MB
3	12.5	Accepted	0.06 sec, 28.91 MB
4	12.5	Accepted	0.06 sec, 28.64 MB
5	12.5	Accepted	0.07 sec, 28.35 MB
6	12.5	Accepted	0.06 sec, 27.80 MB
7	12.5	Accepted	0.06 sec, 28.37 MB
8	12.5	Accepted	0.06 sec, 28.03 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program yang dapat mengelola keranjang belanja sederhana. Program ini akan memiliki 3 kelas, yaitu **Item**, **Account**, dan **Cart**.

Lengkapi **Item.java**, **Account.java**, dan **Cart.java** yang telah diberikan dalam [zip](#) ini. Kumpulkan ketiga file dalam sebuah zip file bernama keranjang.zip

Kelas **Item** merupakan representasi barang yang memiliki atribut sebagai berikut:

- **name** yang bertipe String yang merepresentasikan nama barang.
- **price** yang bertipe integer yang merepresentasikan harga barang.
- **discount** yang bertipe integer yang merepresentasikan diskon/potongan harga pada barang.

Kelas **Item** akan memiliki method sebagai berikut:

- Konstruktor yang menerima parameter **name**, **price**, dan **discount**. Nama barang harus terdiri dari 3-20 karakter (inklusif), harga barang tidak boleh bernilai negatif, dan diskon tidak boleh bernilai negatif dan kurang dari harga barang.
- Method **getName** yang mengembalikan nama barang.
- Method **getPrice** yang mengembalikan harga barang.
- Method **getDiscount** yang mengembalikan diskon barang.
- Method **getTotalPrice** yang mengembalikan harga barang setelah diskon.

Kelas **Account** merupakan representasi akun yang memiliki atribut sebagai berikut:

- **name** yang bertipe String yang merepresentasikan nama pengguna.
- **balance** yang bertipe integer yang merepresentasikan saldo pengguna.

Kelas **Account** akan memiliki method sebagai berikut:

- Konstruktor yang menerima parameter **name** dan **balance**. Nama pengguna harus terdiri dari 3-20 karakter (inklusif) dan saldo pengguna tidak boleh bernilai negatif.
- Method **getName** yang mengembalikan nama pengguna.
- Method **getSaldo** yang mengembalikan saldo pengguna.
- Method **addSaldo** yang menerima parameter **amount** bertipe integer dan menambahkan saldo pengguna sebesar **amount**. **amount** harus lebih dari 0.
- Method **reduceSaldo** yang menerima parameter **amount** bertipe integer dan mengurangi saldo pengguna sebesar **amount**. **amount** harus lebih dari 0 dan saldo pengguna setelah dikurangi **amount** tidak boleh kurang dari 0.

Kelas **Cart** merupakan representasi keranjang belanja yang memiliki atribut sebagai berikut:


- **items** yang bertipe **List<Item>** yang merepresentasikan barang-barang yang ada di keranjang belanja.
- **account** yang bertipe Account yang merepresentasikan akun pengguna.

Kelas **Cart** akan memiliki method sebagai berikut:

- Konstruktor yang menerima parameter **account** bertipe Account.
- Method **getAccount** yang mengembalikan akun pengguna.
- Method **getItems** yang mengembalikan barang-barang yang ada di keranjang belanja.
- Method **addItem** yang menerima parameter **item** bertipe Item dan menambahkan barang ke keranjang belanja.
- Method **removeItem** yang menerima parameter **name** bertipe String dan menghapus seluruh barang dengan nama ``name`` dari keranjang belanja.

- Method `getTotalPrice` yang mengembalikan total harga barang dalam keranjang belanja.
- Method `checkout` yang mengurangi saldo pengguna sebesar total harga barang dalam keranjang belanja. Jika saldo pengguna tidak cukup, maka method ini akan mengembalikan exception. Jika saldo pengguna cukup, maka method ini akan mengurangi saldo akun dan menghapus seluruh barang dalam keranjang belanja.

Java 8 ▾

 [keranjang.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	4	Accepted	0.06 sec, 27.88 MB
2	4	Accepted	0.06 sec, 28.91 MB
3	4	Accepted	0.06 sec, 28.75 MB
4	4	Accepted	0.06 sec, 28.34 MB
5	4	Accepted	0.07 sec, 28.90 MB
6	5	Accepted	0.06 sec, 28.91 MB
7	5	Accepted	0.07 sec, 28.67 MB
8	5	Accepted	0.06 sec, 28.51 MB
9	5	Accepted	0.06 sec, 28.35 MB
10	5	Accepted	0.06 sec, 27.80 MB
11	5	Accepted	0.06 sec, 28.89 MB
12	5	Accepted	0.06 sec, 27.87 MB
13	5	Accepted	0.06 sec, 27.89 MB
14	5	Accepted	0.07 sec, 28.07 MB
15	5	Accepted	0.06 sec, 27.87 MB
16	5	Accepted	0.06 sec, 28.73 MB
17	5	Accepted	0.07 sec, 28.52 MB
18	5	Accepted	0.07 sec, 27.89 MB
19	5	Accepted	0.06 sec, 29.05 MB
20	5	Accepted	0.06 sec, 27.87 MB
21	5	Accepted	0.06 sec, 29.96 MB

◀ [Praktikum 6](#)

Jump to...



