

Started on	Tuesday, 28 March 2023, 4:03 AM
State	Finished
Completed on	Tuesday, 28 March 2023, 4:25 AM
Time taken	21 mins 50 secs
Grade	600.00 out of 600.00 (100%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program robot sederhana yang dapat menerima perintah untuk melakukan aksi. Robot dapat menolak perintah apabila perintah yang diberikan tidak sesuai dengan daftar perintah yang dia ketahui.

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

File tersebut akan memiliki sebuah kelas, yaitu **Robot**.

Kelas Robot memiliki atribut sebagai berikut:

- Atribut **x** bertipe **integer** yang melambangkan lokasi robot di sumbu X dengan nilai awal 0.
- Atribut **y** bertipe **integer** yang melambangkan lokasi robot di sumbu Y dengan nilai awal 0.

Kelas Robot memiliki 2 buah method sebagai berikut:

- walk yang menerima 2 buah parameter, yaitu **x** bertipe **integer** dan **y** bertipe **integer**.
- talk yang menerima sebuah parameter, yaitu **language** bertipe **string**.
- receiveCommand yang menerima sebuah parameter, yaitu **command** bertipe **string**.

Penjelasan masing-masing method sebagai berikut:

Pada method **walk**, robot menerima parameter **x** dan **y** dan menambahkan nilai itu ke atribut **x** dan **y** yang dimilikinya. Robot lalu akan mengeprint pesan berupa **Sedang berjalan menuju (x, y)** dimana **x** dan **y** adalah nilai akhir setelah penambahan. Apabila nilai **x** atau **y** setelah ditambahkan akan melebihi 10, method ini melemparkan sebuah **exception** dengan pesan **Tidak bisa jalan**. Nilai **x** dan **y** tidak ditambahkan pada kasus ini.

Robot hanya dapat berbicara menggunakan bahasa Indonesia dan bahasa Inggris saja. Oleh karena itu, pada pemanggilan method **talk**, apabila parameter **language** bukan **Indonesia** atau **Inggris**, robot akan melemparkan sebuah **exception** dengan pesan **Tidak bisa berbicara dalam bahasa {language}**. Apabila input valid, robot akan mengeprint pesan berupa **Sedang berbicara dalam bahasa {language}**.

Method **receiveCommand** digunakan untuk mengecek apakah input **command** benar, input **command** yang valid adalah **walk** dan **talk**. Selain kedua command tersebut, method akan melemparkan exception dengan pesan **Perintah {command} tidak dikenal**.

Java 8

 [Robot.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	16	Accepted	0.07 sec, 27.79 MB
2	16	Accepted	0.07 sec, 28.87 MB
3	16	Accepted	0.07 sec, 29.07 MB
4	16	Accepted	0.07 sec, 27.77 MB
5	16	Accepted	0.07 sec, 30.21 MB
6	20	Accepted	0.07 sec, 27.82 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 3 buah kelas, yaitu `Calculator`, `InvalidOperationException` yang mengextend kelas `Exception`, dan `InvalidDivisionByZero` yang mengextend kelas `Exception`. `InvalidOperationException` akan menerima input berupa pesan custom ketika diinisiasi. Sedangkan, `InvalidDivisionByZero` akan mengembalikan string mutlak ketika pembagian 0.

Kelas `Calculator.java` yang berisi fungsi `calculate` yang mereturn `double`.

Fungsi tersebut akan menerima 3 buah parameter, yaitu `a` bertipe `double` yang berupa inputan pertama, `b` bertipe `double` yang berupa inputan kedua, dan `c` bertipe `String` berupa operator yang akan di eksekusi oleh kalkulator. (inputan pertama dan kedua berarti `a-b`, `a+b`, `a*b`, dan `a/b`)

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.java` yang berisi pesan dengan format `"Invalid operation: " + operation;`.

Apabila kalkulator menerima input pembagian terhadap 0, maka kalkulator juag akan mengembalikan pesan error dari kelas `InvalidDivisionByZero.java` yang berisi pesan dengan format `Tidak dapat melakukan pembagian terhadap 0.`

Java 8

 [Calculator.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.08 sec, 27.89 MB
2	10	Accepted	0.07 sec, 28.90 MB
3	10	Accepted	0.07 sec, 28.76 MB
4	10	Accepted	0.07 sec, 28.01 MB
5	10	Accepted	0.07 sec, 28.30 MB
6	10	Accepted	0.07 sec, 27.81 MB
7	20	Accepted	0.07 sec, 28.94 MB
8	20	Accepted	0.07 sec, 28.35 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Exception - Calculator (Driver)

Menggunakan `Calculator.java` yang sudah dibuat di soal sebelumnya, buatlah program utama yang menerima perintah untuk melakukan kalkulasi sederhana.

Format input:

- 1. Baris pertama berisi input angka pertama.
- 2. Baris kedua berisi input angka kedua.
- 3. Baris ketiga berisi operasi kalkulasi apa yang akan dilakukan oleh angka kedua kepada angka pertama.

Format output:

- 1. Apabila kalkulasi berhasil, maka cetak angka hasil kalkulasi.
- 2. Apabila kalkulasi gagal/program gagal 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 kalkulasi saja.
 - 2. Apabila *exception* yang dikeluarkan merupakan instance *Exception* selain *InvalidOperationException* ataupun *InvalidDivisionByZero*, maka cetak `Unknown exception!`
- 3. Setelah operasi kalkulasi selesai (apapun hasilnya), tutup scanner yang dibuka dan cetak `Calculated.` ke layar.
 - 1. Penutupan scanner bisa dilakukan dengan pemanggilan method `close()` dari scanner.

Contoh input **benar**:

```
20.0
4.0
*
```

Contoh output:

```
80.0
Calculated.
```

Contoh input **salah** (*InvalidOperationException*):

```
20.0
4.0
#
```

Contoh output:

```
InvalidOperationException! Invalid operation: #
Calculated.
```

Contoh input **salah** (*Unknown Exception*):

```
a
```

Contoh output:

```
Unknown exception!
Calculated.
```

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	0.09 sec, 29.86 MB
2	20	Accepted	0.09 sec, 30.13 MB
3	20	Accepted	0.09 sec, 31.69 MB
4	20	Accepted	0.08 sec, 30.91 MB
5	20	Accepted	0.08 sec, 29.79 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program bank yang dapat mensimulasikan cara kerja bank secara sederhana.

Lengkapi [Bank.java](#) dan [Account.java](#). Submit sebagai **Bank.zip**.

File tersebut akan memiliki 2 buah kelas, yaitu **Bank** dan **Account**.

Kelas **Bank** memiliki atribut sebagai berikut:

- **name** yang bertipe string yang melambangkan nama bank.

Kelas **Bank** sebuah method sebagai berikut:

- Konstruktor yang menerima parameter **name** bertipe **string**.
- **transfer** yaitu method untuk melakukan transfer dari satu akun ke akun lain. Menerima 3 buah parameter yaitu **accountFrom** dan **accountTo** yang dua-duanya bertipe **Account** beserta **jumlah** bertipe **integer**. Transfer antar bank (akun dengan nama bank berbeda) akan dikenakan biaya admin sebesar 2500 yang dikurangi dari saldo pengirim. Biaya admin ini tidak ikut ditransfer ke akun tujuan. Apabila **accountFrom** memiliki nama bank yang tidak sama dengan bank yang melakukan transfer, lempar exception dengan pesan **Bukan akun milik bank {bankName}**.
- **createAccount** yaitu method untuk membuat sebuah **account** yang menerima 2 buah parameter **saldo** bertipe **integer** dan **accountName** bertipe **string**. Akun yang dibuat akan memiliki atribut **bankName** dengan nama bank yang membuat akun. Method ini akan melanjutkan pelemparan exception dari kelas **Account**.

Kelas **Account** memiliki atribut sebagai berikut:

- **name** yang bertipe **string** melambangkan nama pemilik akun.
- **bankName** yang bertipe **string** melambangkan nama bank pencipta akun.
- **saldo** yang bertipe **integer** melambangkan saldo akun. Jumlah saldo tidak bisa dibawah 0.

Kelas **Account** memiliki method sebagai berikut:

- Konstruktor yang menerima 3 buah parameter yaitu **accountName** bertipe **string**, **bankName** bertipe **string**, dan **saldo** bertipe **integer**. Apabila **accountName** memiliki panjang kurang dari 3 karakter, dilempar sebuah exception dengan pesan **Nama akun harus memiliki panjang minimal 3 karakter**. Apabila **saldo** yang dimasukkan kurang dari 0, akan dilempar sebuah exception dengan pesan **Tidak dapat membuat akun dengan saldo dibawah 0**.
- **decreaseSaldo** yaitu method untuk mengurangi jumlah saldo pada akun. Menerima sebuah parameter **jumlah** bertipe **integer**. Apabila jumlah saldo tidak mencukupi untuk melakukan operasi, dilemparkan exception dengan pesan **Saldo tidak mencukupi**. Apabila parameter **jumlah** kurang dari 0, lempar exception dengan pesan **Jumlah pengurangan tidak boleh minus**.
- **increaseSaldo** yaitu method untuk menambah jumlah saldo pada akun. Menerima sebuah parameter **jumlah** bertipe **integer**. Apabila parameter **jumlah** kurang dari 0, lempar exception dengan pesan **Jumlah penambahan tidak boleh minus**.
- **getSaldo** yaitu method untuk mendapatkan jumlah saldo akun.
- **getBankName** yaitu method untuk mendapatkan atribut **bankName**.
- **setName** yaitu method untuk mengubah atribut **name**. Menerima sebuah parameter **name** bertipe **string**. Sebuah **name** memiliki ketentuan yaitu memiliki panjang minimal 3 karakter. Apabila tidak memenuhi ketentuan ini, akan dilempar sebuah exception dengan pesan **Nama akun harus memiliki panjang minimal 3 karakter**.
- **getName** yaitu method untuk mendapatkan atribut **name** dari akun.

Java 8

 [Bank.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	8	Accepted	0.07 sec, 27.79 MB
2	8	Accepted	0.07 sec, 28.11 MB
3	8	Accepted	0.07 sec, 30.98 MB
4	8	Accepted	0.08 sec, 28.14 MB
5	8	Accepted	0.07 sec, 27.86 MB
6	8	Accepted	0.07 sec, 28.71 MB
7	8	Accepted	0.07 sec, 27.81 MB
8	8	Accepted	0.07 sec, 28.01 MB
9	8	Accepted	0.07 sec, 29.85 MB
10	8	Accepted	0.07 sec, 28.76 MB
11	8	Accepted	0.07 sec, 28.11 MB
12	12	Accepted	0.07 sec, 29.10 MB

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program untuk melakukan validasi email. Email dianggap valid apabila memenuhi aturan berikut

- Email harus mengandung tepat 1 karakter @ ditengah email
- Bagian sebelum @ tidak boleh kosong
- Bagian email setelah @ harus memiliki tepat 1 buah titik (.)
- Email hanya akan mengandung huruf, angka, (@), dan (.). Karakter lain tidak perlu diperiksa
- `"john@example.com"` => Email Valid
- `"jane.doe@gmail.com"` => Email Valid

Sebagai contoh email tidak valid apabila:

- `""` => mengembalikan pesan "Email tidak boleh kosong"
- `"example.com"` => mengembalikan pesan "Email harus mengandung tepat satu buah @"
- `"@example.com"` => mengembalikan pesan "@ tidak boleh di awal email"

Domain email merupakan bagian setelah @. Domain email tidak valid apabila:

- `"john@com"` => mengembalikan pesan "Email harus memiliki domain yang valid"
- `"john@.com"` => mengembalikan pesan "Email harus memiliki domain yang valid"
- `"john@com."` => mengembalikan pesan "Email harus memiliki domain yang valid"

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

File tersebut akan memiliki 3 buah kelas, yaitu Email, InvalidEmailException yang mengextend kelas Exception, dan InvalidDomainException yang mengextend kelas Exception. InvalidEmailException akan menerima input berupa pesan custom ketika diinisiasi. Sedangkan, InvalidDomainException akan mengembalikan string mutlak ketika domain email tidak valid.
Dilarang menambahkan fungsi baru kedalam file `Email.java`

Java 8

 [Email.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.07 sec, 27.99 MB
2	10	Accepted	0.07 sec, 28.72 MB
3	10	Accepted	0.07 sec, 28.22 MB
4	20	Accepted	0.08 sec, 28.00 MB
5	10	Accepted	0.07 sec, 28.34 MB
6	10	Accepted	0.07 sec, 27.96 MB
7	20	Accepted	0.07 sec, 28.59 MB
8	10	Accepted	0.07 sec, 29.13 MB

Question **6**
Correct
Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Menggunakan `Email.java` yang sudah dibuat di soal sebelumnya, 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.
- 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.
 - 1. Penutupan scanner bisa dilakukan dengan pemanggilan method `close()` dari scanner.

Contoh input **benar**:

praktikum@oop.com

Contoh output:

true
Email validated.
Operation finished.

Contoh input **salah** (InvalidEmailException):

oop.com

Contoh output:

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	33	Accepted	0.07 sec, 27.95 MB

No	Score	Verdict	Description
2	33	Accepted	0.07 sec, 28.85 MB
3	34	Accepted	0.06 sec, 28.56 MB

[◀ Feedback Praktikum 7](#)

Jump to...