

[Dashboard](#) / [My courses](#) / [ITB IF2010 2 2425](#) / [\[Java\] Praktikum 6](#) / [Praktikum 6 \(Latihan\)](#).

Started on	Monday, 9 June 2025, 10:57 AM
State	Finished
Completed on	Monday, 9 June 2025, 12:11 PM
Time taken	1 hour 14 mins
Grade	400.00 out of 400.00 (100%)

Question **1**

Correct

Mark 100.00 out of 100.00

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 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

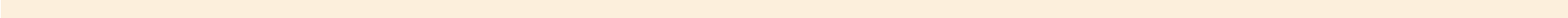
Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.06 sec, 28.38 MB
2	12	Accepted	0.06 sec, 30.35 MB
3	12	Accepted	0.06 sec, 28.87 MB
4	12	Accepted	0.06 sec, 28.91 MB
5	12	Accepted	0.06 sec, 28.85 MB
6	12	Accepted	0.06 sec, 28.92 MB
7	12	Accepted	0.06 sec, 26.90 MB
8	16	Accepted	0.06 sec, 28.94 MB



Question **2**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Pada soal ini, kita akan mencoba mendapatkan data-data yang dimiliki setiap mahasiswa ITB. Setiap Mahasiswa pasti memiliki NIM, IPK dan daftar matkul yang pernah diikuti.. Berikut adalah garis besar kelas Mahasiswa yang dimaksud.

```
import java.util.*;

class Mahasiswa {

    private int NIM;

    private double IPK;

    private List matkul;

    Mahasiswa(int NIM, double IPK) {

        this.NIM = NIM;

        this.IPK = IPK;

        this.matkul = new ArrayList();

    }

    private void addMatkul(String matkul) {

        this.matkul.add(matkul);

    }

}
```

Sebagai mahasiswa yang iseng, kita ingin melihat NIM, IPK, dan menambahkan matkul-matkul susah ke daftar matkul teman kita. Sayangnya atribut NIM, IPK dan method untuk menambahkan matkul bersifat private di kelas Mahasiswa, sehingga kita harus menggunakan reflection untuk mengaksesnya.

Buatlah kelas MahasiswaDecoder yang memiliki interface sebagai berikut

```
import java.lang.reflect.*;

public class MahasiswaDecoder {

    MahasiswaDecoder(Mahasiswa mahasiswa) {

    }

    public void addMatkul(String name) throws Exception {

    }

    public int getNIM() throws Exception {

    }

    public double getIPK() throws Exception {

    }

}

}
```

Hint: perhatikan kalau Field/Method private, bagaimana cara mengaksesnya? (gunakan method `setAccessible`)

Kumpulkan **MahasiswaDecoder.java**

Java 8

 [MahasiswaDecoder.java](#)

Score: 30

Blackbox

Score: 30

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.46 sec, 28.04 MB
2	10	Accepted	0.78 sec, 28.42 MB
3	10	Accepted	0.65 sec, 28.56 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Terdapat sebuah class Reflection.java dengan isi sebagai berikut

```
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class Reflection{
    //TIDAK BOLEH MENGUBAH NAMA METHOD YANG SUDAH ADA DAN PARAMS SERTA INPUT TYPENYA
    //BOLEH MENAMBAHKAN PRIVATE / PROTECTED METHOD SESUAI DENGAN KEBUTUHAN
    //DI LUAR SANA ADA KELAS YANG NAMANYA Ghost DAN Secret.

    public ArrayList<String> ghostMethods(){

        return new ArrayList<>();
    }

    public Integer sumGhost() throws Exception{

        return 0;
    }

    public String letterGhost() throws Exception{

        return "";
    }

    public String findSecretId(List<Secret> sl, String email) throws Exception{

        return "NaN";
    }
}
```

Tugas anda adalah implementasi ke empat method tersebut dengan ketentuan sebagai berikut

1. Terdapat sebuah kelas **Ghost** yang memiliki N method, dan semua method tersebut adalah private. Sesuai namanya, yaitu **Ghost**, maka isi kelas tersebut adalah misterius dan tidak diketahui baik nama method maupun return dari method tersebut (dipastikan return dari Ghost antara Integer dan String saja).
2. Implementasi method **getMethod()** dimana akan membaca seluruh method yang dimiliki oleh kelas **Ghost** dan menyimpan nama method tersebut ke dalam ArrayList
3. Implementasi method **sumGhost()** dimana akan menjumlahkan seluruh return dari method dari kelas **Ghost** yang memiliki kembalian Integer
4. Implementasi method **letterGhost()** dimana akan mengembalikan concat dari seluruh method di kelas **Ghost** yang memiliki kembalian String. (Ketika concat langsung concat aja, gausah di kasih spasi atau pemisah lainnya).
5. Terdapat kelas lain yang bernama **Secret** dengan isi sebagai berikut

```
public class Secret{
    private String email;
    private String nama;
    public static Integer counter = 0;
    private String uniqueId;

    //KELAS SUDAH PATEN, TIDAK BOLEH DIEDIT EDIT


    public Secret(String email, String nama){
        //MAGIC HAPPENED THERE.
        //Intinya konstruktor biasa, cuma ada randomisasi buat uniqueId, dan jangan lupa
        //counter++ pada bagan akhir konstruktor ini.
    }

    public boolean isThis(String email){
        //GUNAKAN FUNGSI INI UNTUK MENCOCOKAN EMAIL DENGAN EMAIL YANG DIMILIKI KELAS
        return this.email.equalsIgnoreCase(email);
    }
}
```

1. Implementasi method `findSecretId(List<Secret> sl, String email)` yang berfungsi akan mencari `Secret` sesuai dengan email yang hendak dicari. Apabila `Secret` dengan email yang dimaksud ada, maka kembalikan `uniqueId` dari email tersebut, jika tidak kembalikan `NaN` sesuai dengan template yang ada diatas.
2. Tips, gunakan `setAccessible(true)` untuk mengakses method maupun attribut private / protected.
3. Untuk implementasi no. 6, manfaatkan method `isThis(String email)`
4. Untuk debugging di local teman-teman bisa membuat kelas `Secret` sesuai dengan template diatas, lalu memberikan uniqueld dengan String random, dan kelas `Ghost` dengan membuat beberapa method private yang mengembalikan Integer dan String.

kumpulkan `Reflection.java`

Java 8

 [Reflection.java](#)

Score: 108

Blackbox

Score: 108

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.05 sec, 28.57 MB
2	12	Accepted	0.06 sec, 28.82 MB
3	12	Accepted	0.06 sec, 28.42 MB
4	12	Accepted	0.06 sec, 28.51 MB
5	12	Accepted	0.06 sec, 29.00 MB
6	12	Accepted	0.06 sec, 28.39 MB
7	12	Accepted	0.06 sec, 27.11 MB
8	12	Accepted	0.05 sec, 28.49 MB
9	12	Accepted	0.06 sec, 28.81 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Long running task merupakan sebuah perintah eksekusi yang memakan waktu lama. Jika perintah eksekusi ini dilakukan di *main thread* program, maka akan mengakibatkan program utama *hang* selama proses eksekusi berjalan. Oleh karena itu dibutuhkan pemanfaatan **multithreading** agar *long running task* tetap dapat dieksekusi namun tidak membuat program utama *hang*.

Diberikan [Main.java](#), [Client.java](#), [Task.java](#), dan [Worker.java](#) untuk menjadi dasar pengetahuan kalian bagaimana **multithreading** bekerja di Java. Sayangnya implementasi Worker masih kurang efisien (gimana sih asistennya>:(), perhatikan bahwa untuk setiap task diperlukan worker baru untuk menjalankan task tersebut. Tugas kalian adalah menunjukkan bahwa kalian lebih OP dari asisten dengan cara mengimplementasikan [WorkerV2.java](#).

*Jangan ubah isi dari class-class yang telah ada kecuali WorkerV2.java dan Main.java (untuk keperluan test)
**File yang dikumpul hanya WorkerV2.java

Java 8 ▾

 [WorkerV2.java](#)

Score: 50

Blackbox

Score: 50

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.29 sec, 26.88 MB
2	10	Accepted	0.86 sec, 30.43 MB
3	10	Accepted	0.67 sec, 26.26 MB
4	10	Accepted	0.55 sec, 27.85 MB
5	10	Accepted	0.43 sec, 28.31 MB

[◀ Praktikum 6](#)

Jump to...

[Ujian Praktikum - UAS ▶](#)