Started on	Thursday, 28 April 2022, 11:01 AM
State	Finished
Completed on	Thursday, 28 April 2022, 12:33 PM
Time taken	1 hour 32 mins
Marks	400.00/400.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Karena ini praktikum terakhir, kamu akan mencoba belajar membuat soal praktikum! Di soal praktikum ini, mahasiswa akan membuat kelas yang:

- bernama Laptop
- hanya boleh memiliki 1 method: Integer getRamSize()
- hanya boleh memiliki 1 field: String serialNumber

Lengkapilah kelas LaptopTest yang memeriksa apakah kelas yang diberikan sudah memenuhi syarat di atas!

Contoh Implementasi Laptop	Contoh Output

`public class Laptop {`
`  private String serialNumber = "abc";`
`  private boolean debug = true;`
 	
`  public Laptop() {`
`  //`
`  }`
`  public Integer getRamSize() {`
`  //`
`  }`
`  public Integer getDebug() {`
`  //`
`  }`
`}`
	`// testFields:`
	`Banyaknya field hanya boleh 1`
	`// testMethods:`
	`Banyaknya method hanya boleh 1`

`public class Laptop {`
`  private Integer serial_number = "abc";`
 	
`  public Laptop() {`
`  //`
`  }`
`  public String get_ram_size() {`
`  //`
`  }`
`}`
	`// testFields:`
	`Nama field harus serialNumber`
	`Tipe field harus String`
	`// testMethods:`
	`Nama method harus getRamSize`
	`Tipe return method harus Integer`

Hint: jangan lupa (1) perbedaan getMethods dan getDeclaredMethods, dan (2) semua kelas merupakan turunan dari kelas Object.

Kumpulkan **LaptopTest.java**

Java 8

LaptopTest.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.37 sec, 28.50 MB
2	10	Accepted	0.32 sec, 27.83 MB
3	10	Accepted	0.56 sec, 28.83 MB
4	10	Accepted	0.89 sec, 28.61 MB
5	10	Accepted	0.74 sec, 28.19 MB

No	Score	Verdict	Description
6	10	Accepted	0.46 sec, 27.99 MB
7	10	Accepted	0.25 sec, 28.48 MB
8	10	Accepted	0.18 sec, 28.46 MB
9	10	Accepted	0.22 sec, 27.87 MB
10	10	Accepted	0.14 sec, 27.83 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 asisten OOP IF. Setiap asisten pasti memiliki data gaji yang didapatkan dan juga daftar mahasiswa yang direkomendasikan untuk menjadi asisten Labpro selanjutnya. Berikut adalah garis besar kelas asisten yang dimaksud.

```
import java.util.*;

class Assistant {
  private int salary;
  private List recommendations;

  Assistant(int salary) {
    this.salary = salary;
    recommendations = new ArrayList();
  }

  private void addPersonToRecommendations(String name) {
    this.recommendations.add(name);
  }
}
```

Sebagai detektif kita ingin melihat berapa gaji tiap asisten dan juga menambahkan nama kita ke dalam daftar rekomendasi asisten Labpro tahun depan. Sayangnya atribut salary dan method untuk menambahkan nama kita bersifat private di kelas Assistant, sehingga kita harus menggunakan reflection untuk mengaksesnya.

Buatlah kelas AssistantDecoder yang memiliki interface sebagai berikut

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

public class AssistantDecoder {

  AssistantDecoder(Assistant assistant) {

  }

  public void addPersonToRecommendations(String name) throws Exception {

  }

  public int getSalary() throws Exception {

  }
}
```

Hint: perhatikan kalau Field/Method private, bagaimana cara mengaksesnya? (baca slide)

Kumpulkan **AssistantDecoder.java**

Java 8

AssistantDecoder.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	50	Accepted	0.52 sec, 28.19 MB
2	50	Accepted	0.31 sec, 29.04 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

JSON (JavaScript Object Notation) adalah format penulisan objek javascript yang sering digunakan untuk mengubah objek menjadi string, atau sebaliknya. Pada soal ini, Anda diminta membuat JSONWriter yang menuliskan kelas menjadi format JSON. Untuk itu, gunakan anotasi JSONField.java.

Lengkapilah kelas JSONWriter berikut. Contoh penggunaan JSONWriter akan seperti ini:

```
class Person {
    @JSONField
    private String name;

    @JSONField
    private String address;

    private String gender;

    public Person(String name, String address, String gender) {
        this.name = name;
        this.address = address;
        this.gender = gender;
    }

    public static void main(String[] args) {
        Person p1 = new Person("Hojun", "Cisitu", "Male");
        Person p2 = new Person("Qila", "Tubis", "Male");
        System.out.println(new JSONWriter(p1).toString());
        System.out.println(new JSONWriter(p2).toString());
        // menuliskan
        // {"address":"Cisitu","name":"Hojun"}
        // {"address":"Tubis","name":"Qila"}
        // perhatikan kalau:
        // - gender tidak dituliskan, karena tidak ada anotasi @JSONField
        // - kedua field private, tapi bisa diakses
        // asumsikan semua field merupakan string
    }
}
```

Kumpulkan **JSONWriter.java** yang sudah dilengkapi

Java 8

JSONWriter.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	33	Accepted	0.26 sec, 35.37 MB
2	33	Accepted	0.33 sec, 35.34 MB
3	34	Accepted	0.62 sec, 34.02 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Salah satu teknik yang cukup sering dilakukan dalam pemrograman Object Oriented adalah Dependency Injection (DI). Dalam menggunakan teknik DI, dependency yang terdapat pada kelas A tidak diinstasiasikan oleh kelas A sendiri, tetapi di-inject oleh kelas lain. Untuk itu kita akan membuat sebuah kelas **Injector** dengan definisi sebagai berikut

```
public class Injector {
  // Menerima sebuah definisi dependencies
  // yang berupa sebuah instansiasi kelas
  // Catatan: Injector bisa menampung lebih dari 1 dependency,
  //          jadi injector bisa menginject banyak sekaligus.
  //          Jika ada 2 objek dengan kelas yang sama,
  //          inject dengan object terakhir yang di add ke daftar dependency
  void addDependencies(Object object);
  /*
    Menginjeksi dependencies yang telah dibuat ke dalam objek
    Injeksi dilakukan dengan mengacu pada dependencies yang ditambah
    pada method addDependencies.

    Setiap field di object yang diinject, bila kelasnya
    sudah ditambahkan sebagai dependency, maka akan diset sebagai
    object yang sudah di add didependency.
  */
  void inject(Object object) throws Exception {

  }
}
```

Misal terdapat sebuah kelas Card yang bergantung pada kelas GameState.

```
class Player {
  public GameState gameState; // null
  public CardList cards; // null
}
```

Lalu kita menggunakan kelas Injector untuk melakukan injeksi dependency.

```
Injector injector = new Injector();

// Asumsi gameState dan cards sudah terinisialisasi
injector.addDependencies(gameState1);
injector.addDependencies(gameState2);
injector.addDependencies(cards);

// Setelah melakukan inject, nilai gameState pada card tidak bernilai null
injector.inject(player);

player.gameState == gameState2 // true
player.cards == cards // true
```

Buatlah implementasi kelas **Injector.java**.

Java 8

Injector.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.59 sec, 28.38 MB
2	20	Accepted	0.63 sec, 28.23 MB
3	20	Accepted	0.53 sec, 29.38 MB

No	Score	Verdict	Description
4	20	Accepted	0.59 sec, 28.12 MB
5	20	Accepted	0.38 sec, 27.89 MB

◄ Slide Minggu 15

Jump to...