Started on	Wednesday, 17 May 2023, 12:33 PM
State	Finished
Completed on	Wednesday, 17 May 2023, 3:29 PM
Time taken	2 hours 56 mins
Marks	341.00/350.00
Grade	**97.43** out of 100.00

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

BoxOperator.java merupakan kelas ajaib yang dapat melakukan filtering isi dari Box.java berdasarkan tipe kelas dari object-object yang berada di dalam Box.java. Silahkan implementasi method getContents yang berada di BoxOperator.java.

Notes: **Jangan ubah-ubah kelas Box.java**, cukup **implementasikan getContents** yang ada di dalam BoxOperator.java dan **kumpulkan BoxOperator.java**.

Contoh Input dan Output:

Input (Pseudocode):	Output (Pseudocode):
BoxOperator.getContents([21, 3.4f, true, "TEST", 'c'], Integer.class)	[21]
BoxOperator.getContents([21, 3.4f, true, "TEST", 'c'], Object.class)	[21, 3.4f, true, "TEST", 'c']

Hint:

- Gunakan fungsi getClass() dari kelas object untuk mendapat Class<?> dari suatu object.
- Tipe Class<T> memiliki fungsi public boolean isAssignableFrom(Class<?> cls) yang dapat digunakan untuk memeriksa apakah sebuah Class<?> bisa di assign ke Class<T>.
- Tipe Class<T> memiliki fungsi `public T cast(Object obj)` yang dapat digunakan untuk cast object ke T dengan kelas Class<T>

Java 8

BoxOperator.java

Score: 60

Blackbox

Score: 60

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.49 sec, 28.99 MB
2	10	Accepted	0.33 sec, 27.91 MB
3	10	Accepted	0.19 sec, 28.73 MB
4	10	Accepted	0.16 sec, 27.86 MB
5	10	Accepted	0.19 sec, 27.81 MB
6	10	Accepted	0.16 sec, 28.00 MB

Question **2**

Partially correct

Mark 91.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Buatlah sebuah program untuk sebuah sistem e-commerce untuk mengelola produk-produk yang dijual. Program sebagai berikut:

- Kelas abstrak `Product` yang memiliki atribut `id` (integer), `name` (string), `price` (integer), dan `description` (string). Kelas `Product` memiliki method abstrak dengan nama `getInfo` yang mengembalikan sebuah `string`. Untuk setiap atribut, terdapat method `setter` dan `getter`. Catatan: Dipastikan setiap `Product` memiliki ID yang beda (Unique ID).
- Kelas `ElectronicProduct` yang merupakan turunan dari `Product` yang memiliki atribut tambahan `warrantyPeriod` (string) yaitu periode garansi. Kelas ini memiliki method tambahan `getWarrantyPeriod` yang mengembalikan `string` berupa `warrantyPeriod`. Method `getInfo` di-override untuk mengembalikan `string` dengan format `{id}: {name} - {price} - {warrantyPeriod} - {description}`.
- Kelas `FashionProduct` yang merupakan turunan dari `Product` yang memiliki atribut tambahan `size` (string) yaitu periode garansi. Kelas ini memiliki method tambahan `getSize` yang mengembalikan `string` berupa `size`. Method `getInfo` di-override untuk mengembalikan `string` dengan format `{id}: {name} - {price} - {size} - {description}`.
- Kelas `FoodProduct` yang merupakan turunan dari `Product` yang memiliki atribut tambahan `expiryDate` (string) yaitu tanggal kadaluwarsa. Kelas ini memiliki method tambahan `getExpiryDate` yang mengembalikan `string` berupa `expiryDate`. Method `getInfo` di-override untuk mengembalikan `string` dengan format `{id}: {name} - {price} - {expiryDate} - {description}`.
- Interface `ProductManagement` yang memiliki beberapa method sebagai berikut:

  1. `addProduct` bertipe void menerima parameter bertipe `Product`.
  2. `removeProduct` bertipe `Product` menerima parameter bertipe `id` berupa `integer`.
  3. `updateProduct` bertipe void menerima parameter bertipe `id` berupa `integer`, `name` bertipe `string`, `price` bertipe `integer`, dan `description` bertipe `string`.
  4. `getAllProduct` bertipe `List<Product>` dan tidak menerima parameter apapun.
  5. `getProduct` bertipe `Product` dan menerima parameter `id` berupa `integer`.

- Kelas `ProductCatalog` yang mengimplementasikan interface `ProductManagement`. Kelas ini memiliki member `productList` bertipe `List<Product>` untuk menyimpan produk-produk yang ada. Ketentuan implementasi method dari interface `ProductManagement` sebagai berikut:

  1. `addProduct` menambahkan `Product` ke `productList`.
  2. `removeProduct` menghilangkan `Product` dari `productList` berdasarkan `id` dan mengembalikannya sebagai return value. Apabila tidak ditemukan, kembalikan `null`.
  3. `updateProduct` mengubah atribut `name`, `price`, dan `description` dari sebuah `Product` berdasarkan `id`. Apabila tidak ditemukan, print pesan `Produk dengan id {id} tidak ditemukan`.
  4. `getAllProduct` mengembalikan seluruh daftar `Product`.
  5. `getProduct` mengembalikan sebuah `Product` berdasarkan `id`. Apabila tidak ditemukan, kembalikan `null`.

Kumpulkan `Product.java`, `ElectronicProduct.java`, `FashionProduct.java`, `FoodProduct.java`, `ProductManagement.java`, `ProductCatalog.java` sebagai `Product.zip`.

Java 8

Product.zip

Score: 91

Blackbox

Score: 91

Verdict: Wrong answer

Evaluator: Exact

No	Score	Verdict	Description
1	9	Accepted	0.33 sec, 28.38 MB

No	Score	Verdict	Description
2	9	Accepted	0.25 sec, 28.70 MB
3	9	Accepted	0.19 sec, 28.97 MB
4	9	Accepted	0.51 sec, 28.40 MB
5	9	Accepted	0.56 sec, 27.90 MB
6	9	Accepted	0.49 sec, 28.43 MB
7	9	Accepted	0.47 sec, 28.21 MB
8	0	Wrong answer	0.43 sec, 28.85 MB
9	9	Accepted	0.60 sec, 28.43 MB
10	9	Accepted	0.25 sec, 28.01 MB
11	10	Accepted	0.21 sec, 28.95 MB

Question **3**

Correct

Mark 150.00 out of 150.00

Time limit	1 s
Memory limit	64 MB

Detektif

Terdapat sebuah kelas rahasia (**method dan attribute-nya tidak diketahui**) yang akan kita sebut sebagai kelas **MysteriousCase**. Meskipun rahasia, **MysteriousCase** ternyata memiliki beberapa aturan yaitu.

1. Merupakan kelas yang "**concrete**" artinya tidak akan meng-inherit maupun di-inherit (tidak merupakan kelas child maupun kelas parent).
2. Semua attribute-nya bertipe **String atau Integer**.
3. Semua methodnya:
    1. Memiliki return value **String atau Integer**
    2. BIsa menerima parameter ataupun tidak dan apabila menerima parameter maka parameternya dipastikan **consecutive Integer** atau **consecutive String**, parameter Integer dan String tidak akan bercampur. Consecutive = 1..n
4. **Access modifier** pada class, attribute, maupun method **tidak dibatasi** (namun tetap sesuai aturan), contohnya public, private, protected, static, final dst...

Buatlah kelas **Detective** (yang pada dasarnya adalah **reflection class**) yang bertujuan untuk membongkar isi dari MysteriousCase. Detective mempunyai 8 method utama yaitu

1. **investigateSecretAttribute()** <- Sudah diimplementasikan
2. **investigateSecretMethod()** <- Sudah diimplementasikan
3. **investigateSecretMethodWithParam()**
4. **getNumberOfIntUsed()** <- Sudah diimplementasikan
5. **getNumberOfStringUsed()**
6. **printAllMethodSignature()**
7. **printAllAttributeSignature()** <- Sudah diimplementasikan

Detail parameter dan tujuan fungsi dapat dibaca pada file yang disediakan. Disediakan Detective.java yang sudah diimplementasikan beberapa (**4**) method utamanya, lengkapilah implementasi method utama yang lainnya dan kumpulkan kembali Detective.java . Perhatikan bahwa

1. Disediakan juga MainTest.java dan MysteriousCase1.java yang berguna untuk membantu melakukan pengujian
2. Anda boleh membuat method intermediate/helper selama tidak merubah kode yang dari awal sudah ada dan tidak menambah import baru.

Java 8

Detective.java

Score: 120

Blackbox

Score: 120

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.07 sec, 28.24 MB
2	10	Accepted	0.07 sec, 28.90 MB
3	10	Accepted	0.07 sec, 28.38 MB
4	10	Accepted	0.07 sec, 28.52 MB

No	Score	Verdict	Description
5	10	Accepted	0.07 sec, 27.85 MB
6	10	Accepted	0.07 sec, 28.93 MB
7	10	Accepted	0.08 sec, 27.86 MB
8	10	Accepted	0.07 sec, 27.88 MB
9	10	Accepted	0.08 sec, 28.94 MB
10	10	Accepted	0.08 sec, 28.73 MB
11	10	Accepted	0.18 sec, 34.05 MB
12	10	Accepted	0.18 sec, 33.33 MB

◄ Announcements

Jump to...

Bagian 2 - Ujian Praktikum 2 ►