Started on	Sunday, 1 June 2025, 6:25 AM
State	Finished
Completed on	Wednesday, 4 June 2025, 2:24 PM
Time taken	3 days 7 hours
Grade	**500.00** out of 500.00 (**100**%)

Question **1**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Integer List Processor

Spesifikasi

Lengkapi `IntegerListProcessor.java` sehingga program:

- Membaca integer **N**, lalu **N** bilangan ke `List<Integer>`
- Menghapus semua elemen genap
- Jika ukuran list ganjil → urut naik; jika genap → urut turun
- Mengalikan tiap elemen dengan 3
- Memanggil `printList(list)` untuk mencetak hasil

Contoh input:

```
6
7 2 5 8 3 2
```

Contoh output:

```
9 15 21
```

Kumpulkan file `IntegerListProcessor.java`

Java 8

IntegerListProcessor.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.15 sec, 32.29 MB
2	25	Accepted	0.16 sec, 33.67 MB
3	25	Accepted	0.14 sec, 32.14 MB
4	25	Accepted	0.14 sec, 33.46 MB

Question **2**

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 uniqueId 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	0.06 sec, 27.92 MB
2	12	Accepted	0.06 sec, 28.66 MB
3	12	Accepted	0.06 sec, 30.79 MB
4	12	Accepted	0.06 sec, 28.59 MB
5	12	Accepted	0.06 sec, 28.45 MB
6	12	Accepted	0.06 sec, 29.32 MB
7	12	Accepted	0.06 sec, 28.45 MB
8	12	Accepted	0.06 sec, 30.86 MB
9	12	Accepted	0.06 sec, 30.21 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

DND

Dungeon and Dragon atau biasa disingkat DnD adalah sebuah permainan peran (role-playing). DnD dimainkan dengan cara bercerita (story telling) yang melibatkan pemain untuk membuat karakter dalam dunia imaginatif dan melakukan berbagai aksi didalamnya. Anda diminta untuk membuat simulasi DnD sederhana. Untuk membantu implementasi, anda akan diwajibkan menggunakan salah satu design pattern yaitu **Strategy**. Nantinya sebuah karakter akan mempunyai kelas tertentu dengan kemampuan tertentu. Karakter dapat memilih kelas tertentu sesuai dengan gaya bermain masing-masing.

Strategy memungkinkan penggunaan algoritma yang berbeda pada suatu objek atau kelas yang diatur pada waktu runtime, disesuaikan dengan konteks atau kebutuhan yang dihadapi. Dengan **Strategy**, kita dapat memisahkan algoritma kemampuan dari tiap kelas tanpa mempengaruhi implementasi dari kode karakter.

Disediakan DnD.zip yang berisikan

1. Main_Dummy.java **<== Pendukung pengujian**
2. Kelas.java
3. Karakter.java
4. Domba.java **<== Perlu diimplementasikan**
5. Pendekar.java **<== Perlu diimplementasikan**
6. Pembunuh.java **<== Perlu diimplementasikan**
7. Penyihir.java **<== Perlu diimplementasikan**

Kumpulkan kembali **semua file yang perlu diimplementasikan saja** dalam zip dengan nama dan format sama yaitu **DnD.zip**. Disediakan Main_Dummy.java untuk membantu melakukan pengujian.

Lebih lanjut terkait Strategy Pattern cek disini

Java 8

DnD.zip

Score: 80

Blackbox

Score: 80

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.28 sec, 28.24 MB
2	10	Accepted	0.32 sec, 29.32 MB
3	10	Accepted	0.27 sec, 28.38 MB
4	10	Accepted	0.07 sec, 30.52 MB
5	10	Accepted	0.06 sec, 28.38 MB
6	10	Accepted	0.07 sec, 28.91 MB
7	10	Accepted	0.07 sec, 27.95 MB
8	10	Accepted	0.26 sec, 29.25 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Redis

Deskripsi

Redis adalah sistem penyimpanan key-value yang mendukung berbagai tipe data. Redis sering digunakan untuk caching, penyimpanan sesi, dan berbagai aplikasi lainnya. Redis menyimpan data dalam memori berbentuk map sehingga akses data hanya membutuhkan kompleksitas O(1).

Implementasi Redis dalam tutorial ini memiliki fitur sebagai berikut:

1. **Generic Key-Value Pair**: Mendukung berbagai tipe data untuk key dan value melalui generic type `<K, V>`.
2. **Time-to-Live (TTL)**: Setiap pasangan key-value memiliki waktu kedaluwarsa (TTL) dalam satuan detik.
3. **Exception Handling**: Menangani kondisi invalid TTL dan TTL yang sudah kedaluwarsa.
4. **Object Comparison**: Metode untuk membandingkan key dan value antara dua objek Redis.

Struktur Kelas

1. Redis

Kelas utama yang mengimplementasikan key-value store dengan TTL.

2. InvalidTtlException

Exception yang dilempar ketika TTL bernilai negatif atau nol.

3. TtlExpiredException

Exception yang dilempar ketika mencoba mengakses value yang sudah kedaluwarsa (TTL habis).

Konstruktor

- `Redis(K k, V v, long ttl)`: Membuat objek Redis baru dengan key `k`, value `v`, dan TTL `ttl` (dalam detik).
- Jika `ttl` ≤ 0, maka akan dilempar `InvalidTtlException` dengan pesan "TTL must be positive".

Getter dan Setter

- `getKey()`: Mengambil key.
- `getVal(long elapsedSeconds)`: Mengambil value jika TTL belum habis. Jika sudah habis, akan dilempar `TtlExpiredException` dengan pesan `"<key> expired"`.
- `setKey(K k)`: Mengubah key tanpa mereset TTL.
- `setVal(V v)`: Mengubah value tanpa mereset TTL.

TTL Management

- `getRemainingTtl(long elapsedSeconds)`: Mengembalikan sisa TTL (dalam detik). Jika TTL sudah habis, kembalikan 0.
- `refresh(long newTtl)`: Memperbarui TTL dengan nilai baru `newTtl` (dalam detik). Jika `newTtl` ≤ 0, maka akan dilempar `InvalidTtlException` dengan pesan "TTL must be positive".

Object Comparison

- `boolean isEqual(Redis<?, ?> c)`: Membandingkan key dan value dengan objek Redis lain.
- `int nearEQ(Redis<?, ?> c)`: Membandingkan kesamaan dan akan mengembalikan nilai:
  - 3: key dan value sama
  - 2: hanya value sama
  - 1: hanya key sama
  - 0: tidak sama

Penting untuk Diperhatikan

1. TTL harus bernilai positif (> 0).
2. Parameter `elapsedSeconds` pada metode `getVal` dan `getRemainingTtl` mewakili detik yang telah berlalu sejak objek dibuat atau di-refresh.

3. Gunakan `Objects.equals()` untuk membandingkan dua objek.
4. Kumpulkan `InvalidTtlException`, `TtlExpiredException`, dan `Redis.java` dalam satu file `Redis.zip`

Contoh Penggunaan

```
public class Main {

    public static void main(String[] args) {

        /* ---------- 1. Konstruktor & akses dasar ---------- */
        Redis<String, Integer> r1 = new Redis<>("token#A", 111, 5);
        System.out.println("r1 key   : " + r1.getKey());
        System.out.println("r1 value : " + r1.getVal(0));          // elapsed = 0
        System.out.println("r1 TTL   : " + r1.getRemainingTtl(2)); // elapsed = 2  -> 5 - 2 = 3
        System.out.println();

        /* ---------- 2. Exception: TTL habis ---------- */
        try {
            System.out.println("Access r1 after expired:");
            r1.getVal(6); // elapsed = 6 > 5  -> harus throw
        } catch (TtlExpiredException e) {
            System.out.println("[Expired] " + e.getMessage());
        }
        System.out.println();

        /* ---------- 3. refresh & akses lagi ---------- */
        r1.refresh(4); // ttl baru 4 s, anggap elapsed di-reset
        System.out.println("r1 TTL after refresh (elapsed 0): " + r1.getRemainingTtl(0));
        System.out.println("r1 value after refresh (elapsed 3): " + r1.getVal(3));
        System.out.println();

        /* ---------- 4. setKey & setVal ---------- */
        r1.setKey("token#A-updated");
        r1.setVal(222);
        System.out.println("r1 key setelah setKey : " + r1.getKey());
        System.out.println("r1 val (elapsed 1)     : " + r1.getVal(1));
        System.out.println();

        /* ---------- 5. Membuat r2, uji isEqual & nearEQ ---------- */
        Redis<String, Integer> r2 = new Redis<>("token#B", 222, 10);

        System.out.println("isEqual(r1,r2) : " + r1.isEqual(r2));      // false
        System.out.println("nearEQ(r1,r2)  : " + r1.nearEQ(r2));       // 2 (value sama)
        System.out.println();

        /* ---------- 6. Exception: TTL negatif saat konstruktor ---------- */
        try {
            Redis<String, String> bad = new Redis<>("x", "oops", 0);
        } catch (InvalidTtlException e) {
            System.out.println("[Invalid TTL] " + e.getMessage());
        }
    }
}
```

Output

```
r1 key   : token#A
r1 value : 111
r1 TTL   : 3

Access r1 after expired:
[Expired] token#A expired

r1 TTL after refresh (elapsed 0): 4
r1 value after refresh (elapsed 3): 111

r1 key setelah setKey : token#A-updated
r1 val (elapsed 1)     : 222

isEqual(r1,r2) : false
nearEQ(r1,r2)  : 2

[Invalid TTL] TTL must be positive
```

Contoh Pembuatan Exception

```
public class SomeException extends RuntimeException {

    public SomeException(String msg) {
        super(msg);
    }
}
```

Java 8

Redis.zip

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	10	Accepted	0.06 sec, 28.93 MB
2	10	Accepted	0.06 sec, 28.43 MB
3	10	Accepted	0.06 sec, 28.40 MB
4	10	Accepted	0.06 sec, 30.92 MB
5	10	Accepted	0.06 sec, 28.62 MB
6	10	Accepted	0.06 sec, 27.90 MB
7	10	Accepted	0.06 sec, 27.86 MB
8	10	Accepted	0.06 sec, 28.48 MB
9	10	Accepted	0.06 sec, 26.72 MB
10	10	Accepted	0.06 sec, 28.05 MB

Question **5**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Penjumlahan Paralel dengan Java Thread

Deskripsi Singkat

Dengan memanfaatkan `Thread`, tugas penjumlahan elemen array dibagi ke beberapa unit kerja agar dapat dieksekusi secara paralel.

Spesifikasi Soal

Diberikan file `Main.java` berisi kerangka program dengan bagian `// TODO` yang belum diisi

Tugas Anda

Lengkapi dua bagian yang masih ber-`TODO` di `Main.java`:

1. Pada **`main` (`// 4. BAGIAN PENGERJAAN`)**
   - Bagi array `data` ke dalam **T** segmen sedemikian rupa sehingga selisih jumlah elemen antar segmen ≤ 1.
     Jika `N % T ≠ 0`, maka segmen dengan indeks lebih kecil mendapat 1 elemen ekstra.
     Contoh: N=8, T=3 → segmen [3, 3, 2].
   - Untuk setiap segmen ke-i (0 ≤ i < T):
     - Buat satu `SumThread` yang menghitung jumlah elemen di segmen tersebut.
     - `SumThread` menulis hasilnya ke `sums[i]`.
   - Jalankan semua `SumThread` lalu tunggu sampai semuanya selesai.
2. Pada **`SumThread.run()`**
   - Hitung `sum` dari `data[dataStartIndex]` hingga `data[dataEndIndex-1]`
   - Simpan `sum` tersebut ke `result[threadIndex]`

Contoh input:

```
8
1 2 3 4 5 6 7 8
3
```

Contoh output:

```
Sum0 = 6
Sum1 = 15
Sum2 = 15
Total = 36
```

Kumpulkan file `Main.java` yang sudah menyelesaikan kedua TODO di atas.

Java 8

Main.java

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	20	Accepted	0.06 sec, 30.07 MB
2	20	Accepted	0.06 sec, 29.01 MB

No	Score	Verdict	Description
3	20	Accepted	0.06 sec, 29.91 MB
4	20	Accepted	0.07 sec, 28.14 MB
5	20	Accepted	0.06 sec, 28.83 MB

◄ Praktikum 5 (Latihan)

Jump to...

Link Slides Tutorial 6 ►