VERSION 3.0

FEBRUARI, 2021

PEMROGRAMAN LANJUT

MODUL 3 - SIMPLE REFACTORING

TIM PENYUSUN :
- HARDIANTO WIBOWO, S.KOM., MT.
- DINDA ARINAWATI WIYONO
- NUR SYAHFEI

PRESENTED BY : LAB. TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

PEMROGRAMAN LANJUT

CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu memahami refactor.
2. Mahasiswa mampu memahami teknik-teknik refactor.
3. Mahasiswa mampu melakukan/mengimplementasi refactor.

KEBUTUHAN HARDWARE & SOFTWARE

- Laptop/PC
- Netbeans/Intellij/Eclipse

MATERI POKOK

Refactoring adalah teknik untuk meningkatkan kualitas kode yang ada. Ia bekerja dengan menerapkan serangkaian langkah-langkah kecil, yang masing-masing mengubah struktur internal kode, sambil mempertahankan perilaku eksternal. Anda mulai dengan program yang berjalan dengan benar, tetapi tidak terstruktur dengan baik, refactoring meningkatkan strukturnya, membuatnya lebih mudah untuk dipertahankan dan diperluas.

Kelebihannya, antara lain :

a. Source code lebih sedikit
b. Mudah dipahami dan diubah
c. Mudah menemukan bug atau error pada program

Adapun contoh refactoring, antara lain :

a. Mengganti nama (mengganti nama variable, class, method, atau nama item lain yang ada di struktur code dengan nama baru).
b. Memindahkan class (memindahkan class ke package lain, serta mengimport referensi terkait class tersebut).
c. Membuat method baru (memisahkan method lama dan/atau membuat method baru yang digunakan untuk meningkatkan perawatan dan keterbacaan pada code).

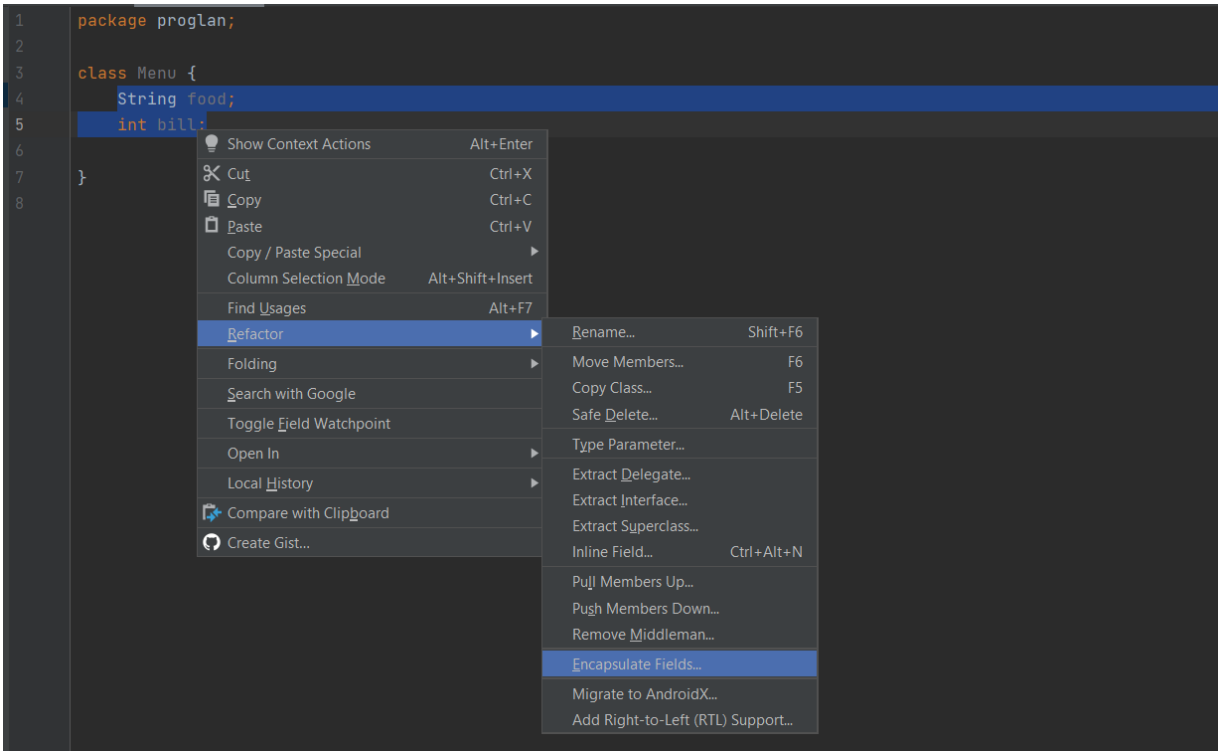
Pada refactoring, programmer juga dapat melakukan enkapsulasi. Enkapsulasi adalah suatu pembungkusan pemrograman agar tidak dapat diakses secara sembarangan atau bisa juga disebut dengan penyembunyian suatu data. Konsep kerja dari enkapsulasi ini melindungi suatu program dari akses program lain yang mempengaruhinya, manfaatnya adalah menjaga keutuhan program yang telah dibuat dengan konsep dan rencana yang sudah ditentukan dari awal. Umumnya dengan cara menambahkan method getter dan setter untuk masing-masing variable dan menjadikan variable tersebut bermodifier private.

Berikut contoh enkapsulasi dengan IntelliJ IDEA Community Edition 2020.3 :

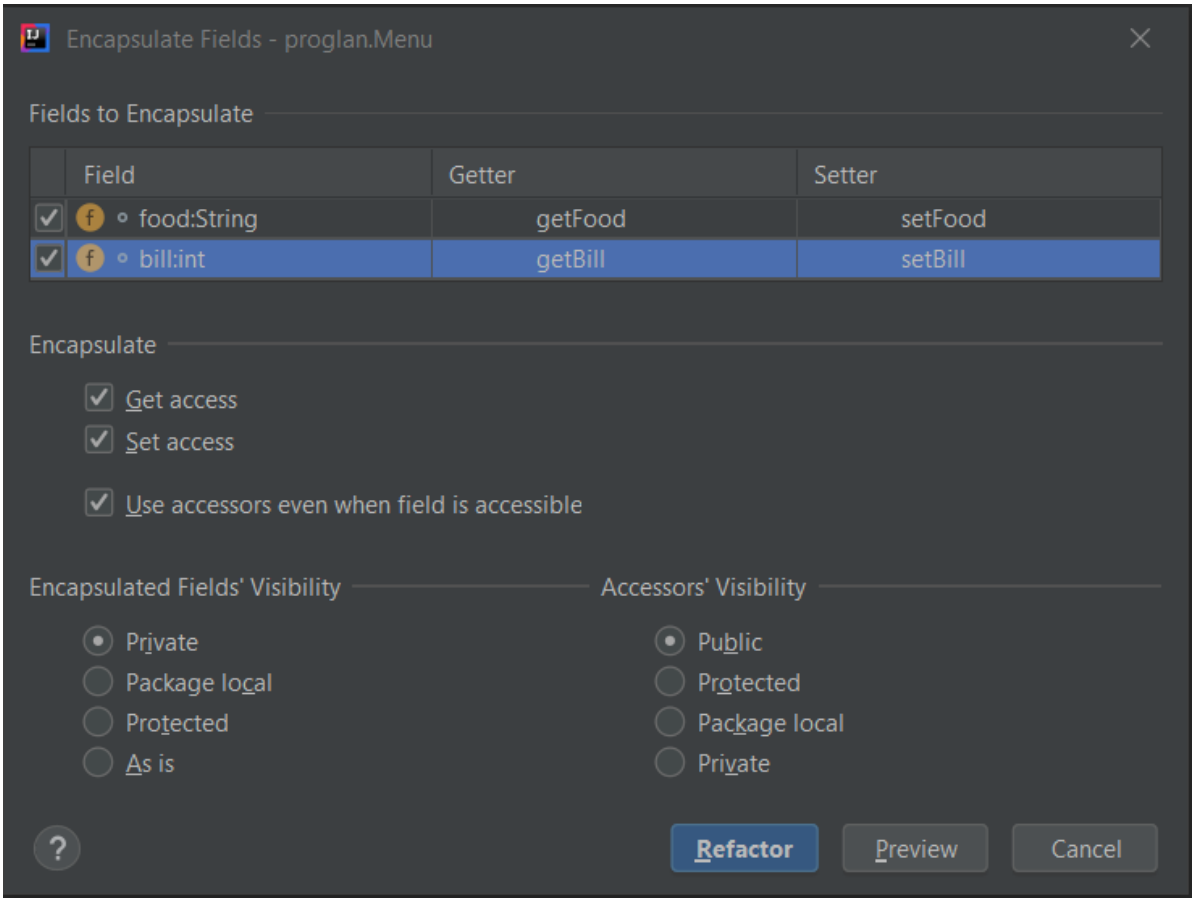
1. Siapkan sebuah class dan beberapa variable

```
package proglan;

class Menu {
    String food;
    int bill;

}
```

2. Seleksi variabelnya, kemudian klik kanan > Refactor > Encapsulate Fields



3. Maka akan muncul pop up seperti berikut, centang variable yg ingin kamu buat setter-getter nya. Kemudian untuk yang lainnya bisa sesuaikan dengan kebutuhanmu. Jangan lupa klik refactornya ya!



4. IDE akan secara otomatis membuat method untuk setter-getter. Kamu bisa mengeditnya jika memang diperlukan, sesuaikan dengan kebutuhan kamu.

```java
package proglan;

class Menu {
    private String food;
    private int bill;

    public String getFood() {
        return food;
    }

    public void setFood(String food) {
        this.food = food;
    }

    public int getBill() {
        return bill;
    }

    public void setBill(int bill) {
        this.bill = bill;
    }
}
```

Sebagian besar refactoring dikhususkan untuk metode penulisan yang benar. Dalam kebanyakan kasus, metode yang terlalu panjang adalah akar dari semua masalah. Keanehan kode di dalam metode ini adalah menyembunyikan logika eksekusi dan membuat metode ini sangat sulit untuk dipahami, dan bahkan lebih sulit untuk diubah.

Berikut adalah beberapa teknik refactoring untuk meminimalisir metode, menghapus duplikasi kode, dan membuka jalan untuk perbaikan di masa mendatang :

1. Extract Method
   - **Problem :** Kamu memiliki fragmen kode yang dapat dikelompokkan bersama.
   - **Solution :** Pindahkan kode ini ke metode (atau fungsi) baru yang terpisah dan ganti kode lama dengan panggilan ke metode tersebut.
2. Extract Variable
   - **Problem :** Kamu memiliki ekspresi yang sulit untuk dipahami.
   - **Solution :** Buat variable terpisah yang berisi ekspresi tersebut.
3. Inline Temp
   - **Problem :** Kamu memiliki variable sementara yang merupakan hasil ekspresi sederhana.
   - **Solution :** Ganti variable sementara tersebut ke hasil ekspresi itu sendiri.

REFERENSI

http://www.cs.unc.edu/~stotts/COMP204/refactor/chap1.html

https://refactoring.guru/refactoring/

LEMBAR KERJA

TUGAS 1

Buatlah sebuah class balok, tambahkan variabel panjang,lebar,dan tinggi. Lalu buatlah setter dan getter ke 3 variabel tersebut, kemudian tambahkan method hasil seperti berikut !

```
protected void hasil(){
    Perhitungan ph = new Perhitungan();
    System.out.println("Hasil luas balok : " + ph.luas( sisi: this));
    System.out.println("Hasil luas balok : " + ph.volume( sisi: this));
}
```

TUGAS 2

Buat class perhitungan, lalu buatlah method untuk menghitung volume dan luas seperti berikut :

```
void perhitungan(){
    int panjang = sisi.getPanjang();
    int lebar = sisi.getLebar();
    int tinggi = sisi.getTinggi();
    luas = panjang*lebar*tinggi;

    int panjangV = sisi.getPanjang();
    int lebarV = sisi.getLebar();
    int tinggiV = sisi.getTinggi();
    volume = 4*panjangV*lebarV*tinggiV;
}
```

Penulisan method hitung volume dan luas masih kurang benar. Refactor method hitung di atas dengan metode **inline temp** dan **extract method** !

TUGAS 3

Penulisan method berikut sulit untuk dipahami, refactor method tersebut dengan metode **extract variable**. Kemudian jelaskan ke asisten!

```
static boolean isCube(long input) {
    return (Math.round(Math.cbrt(input))*Math.round(Math.cbrt(input))
            *Math.round(Math.cbrt(input))) == input;
}
```

TUGAS 4

Buatlah sebuah driver class yang berisi fungsi main. Inisialisasi panjang,lebar, dan tinggi dengan menggunakan setter, lalu panggil method total dari hasil balok !

Kerjakan dengan rapi ya, agar minim terjadi kesalahan syntax dan mudah dibaca.

RUBRIK PENILAIAN

Kriteria	Nilai
Tugas 1	10
Tugas 2	25
Tugas 3 (Optional)	20
Tugas 4	15
Pemahaman	30
Total	**100**

Silahkan dikerjakan tanpa copas dan jangan lupa berdoa sebelum praktikum! Good Luck!!