Bab 13

Unit Testing, TDD, dan Refactoring

13.1 Unit Testing dan Siklus TDD

Sub-CPMK yang Dicakup dalam Bab Ini:

- Sub-CPMK 4.1: Menganalisis code smell dan melakukan refactoring

13.1.1 Konsep Unit Testing

Unit testing memastikan setiap method bekerja sesuai harapan. Pengujian yang baik mempercepat deteksi bug.

13.1.2 Test-Driven Development (TDD)

Siklus TDD: **Red** (tulis test gagal), **Green** (buat test lulus), **Refactor** (perbaiki struktur).

Refactoring adalah proses mengubah sistem perangkat lunak sedemikian rupa sehingga tidak mengubah perilaku eksternal kode namun memperbaiki struktur internalnya [1].

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;

class KalkulatorTest {
    @Test
    void testTambah() {
        Kalkulator k = new Kalkulator();
        assertEquals(5, k.tambah(2, 3));
    }
}
```

Kode Program 13.1: Contoh Unit Test JUnit

13.2 Code Quality dan Refactoring

13.2.1 Code Smell yang Umum

Beberapa contoh code smell:

- Method terlalu panjang
- Duplikasi kode
- Nama variabel tidak jelas
- Kelas melakukan terlalu banyak tugas

13.2.2 Teknik Refactoring

Refactoring adalah proses memperbaiki struktur kode tanpa mengubah perilaku.

- Extract Method
- Rename Variable
- Replace Magic Number with Constant
- Move Method

Aktivitas Pembelajaran

1. **Unit Test**: Buat unit test untuk class `RekeningBank`.
2. **TDD**: Terapkan siklus Red-Green-Refactor pada fitur login sederhana.
3. **Refactoring**: Perbaiki class dengan method yang terlalu panjang.
4. **Diskusi**: Identifikasi code smell pada proyek mini.

Latihan dan Refleksi

1. Jelaskan manfaat unit testing bagi tim pengembang.
2. Buat test case untuk method `hitungDiskon()` pada class **`Produk`**.
3. Sebutkan minimal 3 contoh refactoring dan kapan digunakan.
4. Bagaimana TDD membantu desain yang lebih baik?
5. **Refleksi**: Apa code smell yang paling sering Anda temui?

Asesmen (Evaluasi Kinerja)

Instrumen Penilaian untuk Sub-CPMK 4.1

A. Pilihan Ganda

1. Siklus TDD yang benar adalah:
   (a) Green-Red-Refactor
   (b) Red-Green-Refactor
   (c) Refactor-Red-Green
   (d) Red-Refactor-Green
2. Refactoring bertujuan untuk:
   (a) Mengubah perilaku program
   (b) Memperbaiki struktur tanpa mengubah perilaku
   (c) Menambah fitur baru
   (d) Menghapus semua test

B. Tugas Praktik

- Lakukan refactoring pada class **`Order`** agar mengikuti SRP dan tambahkan unit test.

Rubrik Penilaian: Lihat Lampiran A

Checklist Pencapaian Kompetensi

Centang item berikut setelah Anda yakin telah menguasainya:

☐ Saya memahami konsep unit testing

☐ Saya memahami siklus TDD

☐ Saya dapat mengidentifikasi code smell

☐ Saya dapat melakukan refactoring dasar

☐ Saya dapat menulis test untuk method penting

Rangkuman

Unit testing dan TDD meningkatkan kualitas software. Refactoring membantu menjaga struktur kode tetap bersih dan mudah dipelihara, sebagaimana didokumentasikan dalam katalog refactoring [1].

Kata Kunci: ***Unit Testing***, ***TDD***, ***Refactoring***, ***Code Smell***, ***JUnit***

Daftar Pustaka

[1] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition, 2018.