Praktikum 14

Analisis dan Desain Berorientasi Objek

Dependency Inversion Principle

Pada praktikum kali ini, kita akan mencoba prinsip terakhir dari SOLID Principles, yaitu Dependency Inversion Principle (DIP). Kata kunci dari DIP adalah:

A. High-level modules should not depend on low-level modules. Both should depend on abstractions.
B. Abstractions should not depend upon details. Details should depend upon abstractions.

Code Example

```
public class Product {
      public void AdjustPrice(ProductDiscount productDiscount) {
      }
}

public class ProductDiscount {
}

public class ProductRepository {
      public IEnumerable<Product> FindAll() {
            return new List<Product>();
      }
}

public class ProductService {
      private ProductDiscount _productDiscount;
      private ProductRepository _productRepository;

      public ProductService() {
            _productDiscount = new ProductDiscount();
            _productRepository = new ProductRepository();
      }

      public IEnumerable<Product> GetProducts() {
            IEnumerable<Product> productsFromDataStore =
      _productRepository.FindAll();
            foreach (Product p in productsFromDataStore) {
                  p.AdjustPrice(_productDiscount);
            }
            return productsFromDataStore;
      }
}
```

Potongan kode tersebut belum memenuhi prinsip Dependency Inversion Principle, karena high level modules akan memanggil low level modules sehingga akan terdapat dependensi diantara mereka. Pada potongan kode tersebut, ProductService memanggil ProductRepository dan ProductDiscount, tetapi, sebelumnya harus dilakukan instansiasi objek baru dengan 'new' sehingga kelas Product Service akan memiliki dua dependensi yaitu dengan ProductRepository dan ProductDiscount. Dengan kata lain, ProductService akan memiliki ikatan yang kuat terhadap dua kelas lainnya. Apabila kode tersebut diterapkan, akan sulit apabila akan menggunakan strategi diskon yang bervariasi seperti diskon natal, diskon tahun baru, diskon akhir tahun, dll. Selain itu, apabila database yang digunakan oleh setiap outlet berbeda, akan sulit pula untuk menentukan teknik pengambilan data. Misalnya

outlet-outlet menggunakan database seperti SQL database, MySql database, MongoDb database, file storage, memory storage , dan lain sebagainya, maka teknik pembacaan datanya pun harus disesuaikan. Akan jauh lebih baik jika kita menggunakan suatu strategi dimana kelas ProductService tidak perlu selalu diubah implementasinya setiap ada perubahan, tapi kita gunakan prinsip desain DIP agar data yang diinput kesana adalah data yang sesuai (pelajari slide kuliah).

Untuk tugasnya, Anda diminta untuk menerapkan prinsip Dependency Inversion Principle untuk membuat program penghitungan diskon yang dapat dijalankan di console. Buatlah kelas tester yang akan menguji program anda. Tambahkan kelas-kelas lain sesuai kebutuhan. Gunakan testcase beberapa skema diskon seperti diskon natal 50%, diskon tahun baru 30%, dan diskon akhir tahun 20%. Gunakan testcase berupa file .txt agar memudahkan pengujian.

Contoh input:

```
Harga awal : 100000
Natal
Tahunbaru
Akhirtahun
```

Contoh output:

```
50000
30000
20000
```

Kumpulkan file P14xxyyy.jar atau P14xxyyy.zip yang mengandung kode program javanya. (jangan lupa untuk menghapus nilai "Exclude From JAR File" di Project Properties -> Build -> Packaging.)
Sertakan file .txt yang Anda gunakan.