İnönü Üniversitesi
Bilgisayar Mühendisliği Bölümü
2024 Bahar Nesneye Yönelik Yazılım Mühendisliği (376)
Proje Analiz Rapor

Structural (Yapısal) Tasarım Kalıplarından : Facade

Facade tasarım kalıbı, karmaşık bir alt sisteme basitleştirilmiş bir arayüz sağlayan bir yapısal tasarım kalıbıdır. Bu kalıp, genellikle bir dizi alt sistemle etkileşim kurmayı ve bu alt sistemlerin karmaşıklığını gizlemeyi amaçlar.

Projede, SQLiteJDBC sınıfı bir facade tasarım kalıbı olarak tasarlanmıştır.. Bu sınıf, SQLite veritabanı ile etkileşimi basitleştirir ve yeni tablo oluşturma, veritabanı bağlantısı, tabloya veri ekleme, verileri tablodan çekme, tablodaki verileri silme gibi görevleri yerine getiren metotlar sağlar. Bu metotlar, veritabanı ile etkileşimde bulunurken gereken karmaşık işlemleri (bağlantı kurma, ifade oluşturma, ifadeyi çalıştırma ve hataları yönetme) kapsülleştirir.

```
public interface ISQLiteJDBC {
    Connection connect();
    void createNewTable();
    void insert(Sentence sentence);
    List<Sentence> getTranslations();
    void deleteAll();

    static ISQLiteJDBC getInstance() {
        return null;
    }
}
```

LanguageTranslator sınıfı, Facademodelinin güzel bir örneğidir. Bu sınıf, DeepL API'nin karmaşık alt sistemine basitleştirilmiş bir arayüz sağlar. Çeviri sürecinin karmaşıklığını gizler ve istemcinin kullanımı için basit bir TranslateText yöntemi sağlar.

```
public interface ILanguageTranslator {
    Sentence translateText(String source_text, String sourceLanguage,
String targetLanguage) throws DeepLException, InterruptedException;
    List<Language> getTargetLanguages();
    List<Language> getSourceLanguages();
    static ILanguageTranslator getInstance() {
        return null;
    }
}
```

Behavioral (Davranışsal) Tasarım Kalıplarından : Command

Bir isteği nesneye dönüştürerek isteğin kullanıcı sınıfları tarafından rahatça erişilebilmesi sağlar. Yapılmak istenen işlemi bir nesneye dönüştürerek alıcı nesne tarafından işlemin yerine getirilmesi sağlar. Bu tasarım kalıbı kullanıcı arayüzü elemanları için kullanılabilir.

Bu projede, belirli düğmelere tıklandığında gerçekleştirilmesi gereken eylemleri özetlemek için Komut Modeli kullanılır. DeleteTextAreaCommand, CopyTextCommand ve SwapLanguagesCommand sınıfları komut örnekleridir. Bu sınıfların her biri bir execute() yöntemi içeren ortak bir arayüz uygular.

```
public interface ICommand {
    void execute();
}
```

DeleteTextAreaCommand: Bu komut, kaynak ve hedef metin alanlarındaki metni temizleyen bir eylemle ilişkilidir. Bu komutun execute() yöntemi çağrıldığında ButtonClickedReceiver sınıfının deleteTextArea() yöntemini tetikler.

CopyTextCommand: Bu komut, metni hedef metin alanından sistem panosuna kopyalayan bir eylemle ilişkilidir. Bu komutun execute() yöntemi çağrıldığında ButtonClickedReceiver sınıfının copyText() yöntemini tetikler.

SwapLanguagesCommand: Bu komut, kaynak ve hedef dilleri ve bunlara karşılık gelen metinleri değiştiren bir eylemle ilişkilidir. Bu komutun execute() yöntemi çağrıldığında ButtonClickedReceiver sınıfının swapLanguages() yöntemini tetikler.

TranslationViewController sınıfında bu komutlar, belirli düğmeler tıklatıldığında yürütülecek şekilde ayarlanmıştır. Örneğin aşağıda gösterilen kod, copyTextButton tıklandığında yürütülecek CopyTextCommand'ı ayarlar. Bu, GUI düğmelerinin belirli komutları tetiklediği Komut Modeli'nin klasik bir kullanımıdır.

```
// initialize commands
buttonClickedReceiver = new ButtonClickedReceiver();
copyTextCommand = new CopyTextCommand(buttonClickedReceiver,
                                      targetTextArea,
                                      rootVBox);
// set commands
copyTextButton.setOnAction(e -> copyTextCommand.execute());
```