Tasarım Örüntüleri
Command

Örüntülerin Temel Prensipleri

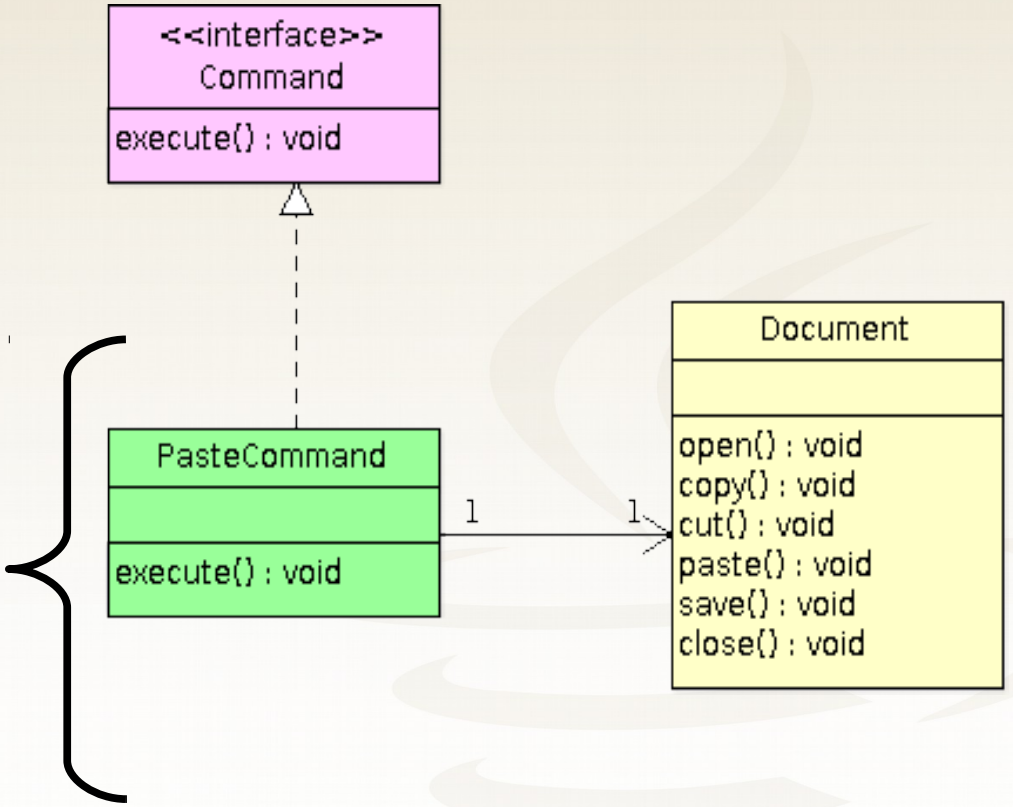
- GoF tasarım örüntülerinin altında yatan temel prensipler
  - Encapsulation
  - Composition
  - Abstract Data Types

Command Örüntüsü

- **Metot invokasyonu**nu encapsule eden bir örüntüdür
- Bir isteğin veya metot invokasyonunun **nesne olarak** ifade edilmesini ve ele alınmasını sağlar

Command Sınıf Diagramı

```
public void execute() {
        document.paste();
}
```



Command Örüntüsünün Java İçerisinde Kullanımı

- Java içerisinde Command örüntüsünün kullanıldığı iki çok bilinen örnek **javax.swing.Action** ve **java.lang.Runnable** arayüzleridir

Action Command

- **Action** arayüzü GUI üzerinden invoke edilebilecek kullanıcı komutlarını encapsule etmek için kullanılır

```
public class DocumentSaveAction extends AbstractAction {
      private Document document;

      public DocumentSaveAction(Document document) {
            this.document = document;
      }

      @Override
      public void actionPerformed(ActionEvent e) {
            this.document.save();
      }
}
```

```
Document document = new Document();
...
JButton btn = new JButton(new DocumentSaveAction(document));
btn.setText("Save");
...
```

Runnable Command

- **Runnable** arayüzü ise ayrı bir **Thread** vasıtası ile çalıştırılacak task'ları nesne olarak encapsule etmemizi sağlar

```
Runnable task = new Runnable() {
     @Override
     public void run() {
          System.out.println("Hello world!");
          try {
               Thread.sleep(1000);
          } catch (InterruptedException e) {}
          }
     };
```

```
Thread t = new Thread(task);

t.start();
```

Command Örüntüsünün Spring İçerisinde Kullanımı

- Spring'deki **JdbcTemplate** ile özdeşleşen programlama pratiğinde ki **Callback**'ler **Command** örüntüsüne karşılık gelmektedir

RowMapper (Callback) Command

```
Collection<Document> result = jdbcTemplate.query(
    "select * from T_DOCUMENT",
    new RowMapper<Document>() {

        @Override
        public Document mapRow(
                            ResultSet rs, int rowNum)
                throws SQLException {
            Document doc = new Document();
            doc.setName(rs.getString("doc_name"));
            doc.setType(rs.getInt("doc_type"));
            return doc;
        }
    }
);
```

Command Örüntüsünün Sonuçları

- Command örüntüsü sayesinde **metot invokasyonlar tekrar tekrar kullanılabilir**, bir yerde saklanabilir veya başka bir yere transfer edilebilir hale gelmektedir
- Command nesnelerini bir araya getirip **composite command nesneleri** oluşturarak daha kompleks bir takım işlemleri yeniden kullanılabilir biçimde ifade etmek de mümkündür

İletişim

JAVA Eğitimleri

Kurumsal Java Eğitimleri

www.java-egitimleri.com

info@java-egitimleri.com

@javaegitimleri

youtube.com/c/
KurumsalJavaEğitimleri