Tasarım Örüntüleri
Template Method

Örüntülerin Temel Prensipleri

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

Template Method Örüntüsü

- Template Method
  - **Davranış encapsulation** yapan diğer bir örüntü
  - **Strategy** örüntüsü ile işlevsel olarak çok benzerdir
  - Her ikisi de **algoritma encapsulation** yapar
  - İşlevsel olarak aynı, **yapısal olarak farklı**dırlar

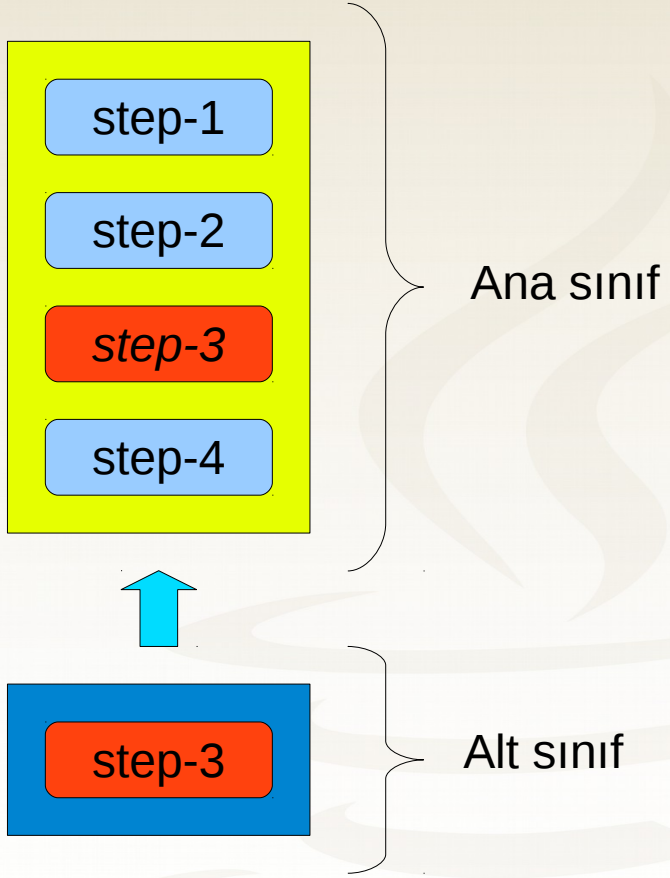
Template Method Örüntüsü

- Bir algoritma **genel hatları ile** birden fazla sınıf için aynı olabilir
- Ancak bu sınıfların her birinde algoritmanın adımlarından **bir kaçı değişiklik** gösterebilir
- Burada **ana akış sabit**, yani değişmezdir
- Ancak ana akış içerisindeki **bazı adımlar değişir**

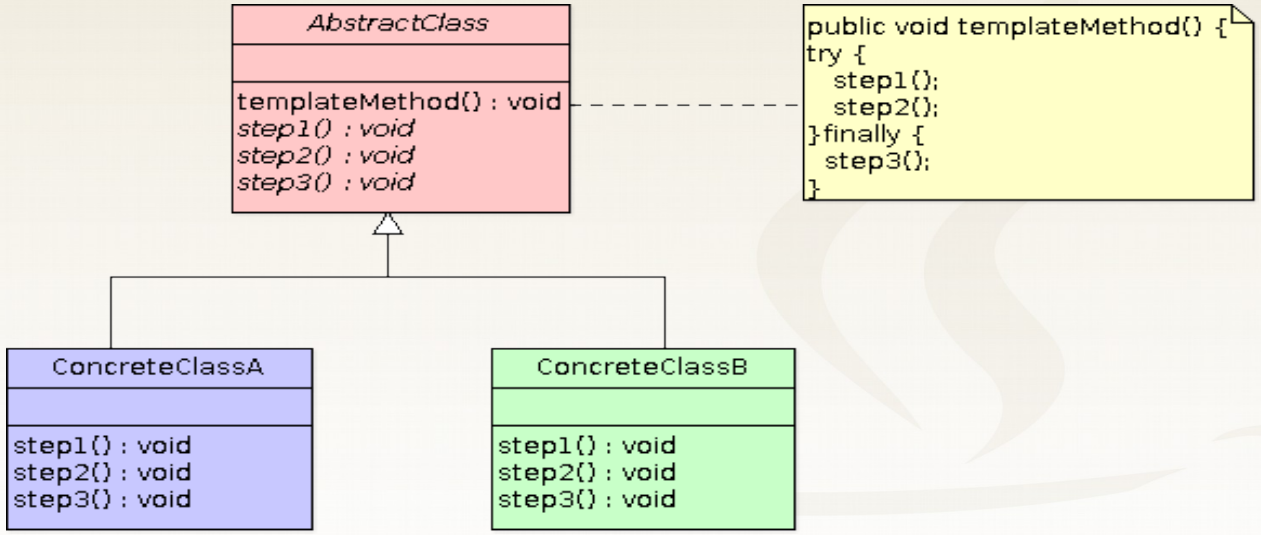
Template Method Örüntüsü

- Template Method ile algoritma encapsulation probleminde **çözüm olarak inheritance** kullanılır
- **Ana akış**, **üst sınıf**ın bir metodu içerisinde encapsule edilir
- **Değişen adımlar** ise **alt sınıflar** tarafından ilgili metotların override edilmesi ile tanımlanır

Template Method Örüntüsü



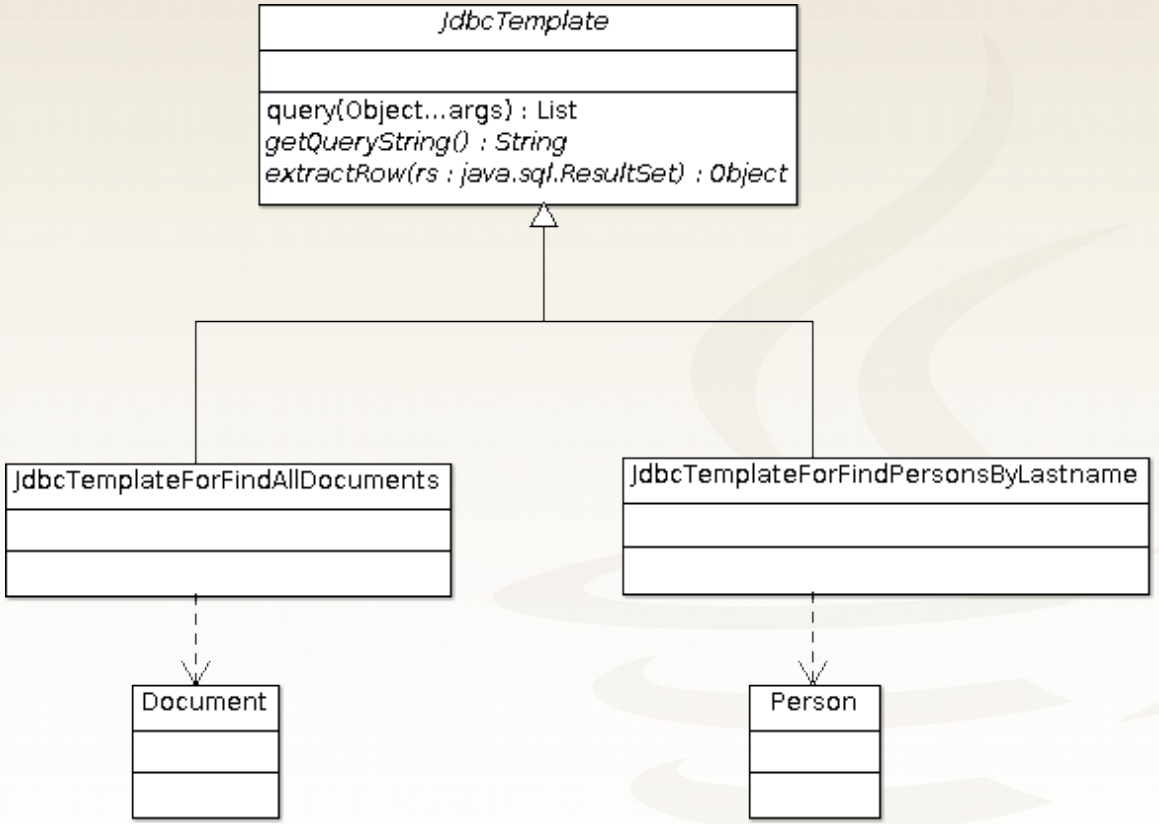
Template Method Sınıf Diagramı



Örnek Problem: JdbcTemplate

- JDBC API ile çalışma adımları:
  Veritabanı bağlantısı kurulması
  Statement nesnesinin oluşturulması
  SQL ifadesinin yazılması
  SQL ifadesinin çalıştırılması
  Dönen ResultSet üzerinde iterate edilerek işlem yapılması
  İşi biten statement ve veritabanı bağlantı nesnelerinin kapatılması
- Bu işlem adımlarından **sadece ikisi** senaryoya göre farklılık gösterir

Örnek Problem: Sınıf Diagramı



Örnek Problem


```
public abstract class JdbcTemplate {
      public final List<?> query() {
            Connection connection = null;
            Statement statement = null;
            try {
                  connection = DriverManager.getConnection(
                        "jdbc:h2:tcp://localhost/~/test","sa","");
                  statement = connection.createStatement();
                  ResultSet rs =
                        statement.executeQuery(getQueryString());
                  List<Object> result = new ArrayList<>();
                  while(rs.next()) {
                        result.add(extractRow(rs));
                  }
                  return result;
            } catch(Exception ex) {
                  throw new RuntimeException(ex);
            } finally {
                  try {
                        statement.close();
                        connection.close();
                  } catch (Exception e) {
                  }
            }
      }
      protected abstract Object extractRow(ResultSet rs)
                                                throws Exception;
      protected abstract String getQueryString();
}
```

Örnek Problem

```
public class JdbcTemplateForFindAllDocuments
                                extends JdbcTemplate {

     @Override
     protected Object extractRow(ResultSet rs)
                                throws Exception {
          Document doc = new Document();
          doc.setId(rs.getLong("id"));
          doc.setName(rs.getString("name"));
          return doc;
     }

     @Override
     protected String getQueryString() {
          return "select * from t_document";
     }
}
```

Örnek Problem

```
public class JdbcTemplateForFindPersonsByLastname
                                extends JdbcTemplate {

     @Override
     protected Object extractRow(ResultSet rs)
                                     throws Exception {
          Person p = new Person();
          p.setId(rs.getLong("id"));
          p.setFirstName(rs.getString("first_name"));
          p.setLastName(rs.getString("last_name"));
          return p;
     }

     @Override
     protected String getQueryString() {
          return "select * from t_person where last_name = ?";
     }
}
```

Java Servlets ve Template Method

```
public abstract class HttpServlet extends GenericServlet {
protected void service(HttpServletRequest req,
HttpServletResponse resp)
        throws ServletException, IOException
    {
        String method = req.getMethod();

        if (method.equals(METHOD_GET)) {
            long lastModified = getLastModified(req);
            if (lastModified == -1) {
                doGet(req, resp);
            } else {
                long ifModifiedSince =
req.getDateHeader(HEADER_IFMODSINCE);
                if (ifModifiedSince < lastModified) {
                    maybeSetLastModified(resp, lastModified);
                    doGet(req, resp);
                } else {

resp.setStatus(HttpServletResponse.SC_NOT_MODIFIED);
                }
            }

        } else if (method.equals(METHOD_HEAD)) {
            long lastModified = getLastModified(req);
            maybeSetLastModified(resp, lastModified);
            doHead(req, resp);

        } else if (method.equals(METHOD_POST)) {
            doPost(req, resp);
```

Java Servlets ve Template Method

```
            else if (method.equals(METHOD_PUT)) {
                doPut(req, resp);

            } else if (method.equals(METHOD_DELETE)) {
                doDelete(req, resp);

            } else if (method.equals(METHOD_OPTIONS)) {
                doOptions(req,resp);

            } else if (method.equals(METHOD_TRACE)) {
                doTrace(req,resp);

            } else {
                String errMsg =
lStrings.getString("http.method_not_implemented");
                Object[] errArgs = new Object[1];
                errArgs[0] = method;
                errMsg = MessageFormat.format(errMsg,
errArgs);


resp.sendError(HttpServletResponse.SC_NOT_IMPLEMENTED,
errMsg);
            }
        }
}
```

Spring Framework ve Template Method

- **Spring ekosistemi**nde Template Method örüntüsü **yaygın** biçimde kullanılır
- Değişken kısımları ana metoda **callback yöntemi** ile metot input parametresi şeklinde geçen bir **varyasyon** tercih edilir

Spring Framework ve Template Method

JAVA Eğitimleri

```
public class JdbcTemplate extends JdbcAccessor implements
JdbcOperations {

     @Override
     public <T> List<T> query(
          String sql, RowMapper<T> rowMapper, Object... args)
          throws DataAccessException {
          ...

          List<T> results = (this.rowsExpected > 0 ? new
ArrayList<>(this.rowsExpected) : new ArrayList<>());
          int rowNum = 0;
          while (rs.next()) {
               results.add(this.rowMapper.mapRow(rs, rowNum+
+));
          }
          return results;
     }

}
```

```
@FunctionalInterface
public interface RowMapper<T> {
     @Nullable
     T mapRow(ResultSet rs, int rowNum) throws SQLException;
}
```

Spring İçerisinde Template Method Örüntüsünün Kullanım Yerleri

- JDBC ile veri erişimi **JdbcTemplate**
- Transaction yönetimi **TransactionTemplate**
- REST servis çağrıları **RestTemplate**
- JMS ile mesajlaşma **JmsTemplate**
- ...Template

Template Method Örüntüsünün Sonuçları

- Algoritmanın **ana akışı sabitlenmiş** ve **kontrol altına alınmış** olur
- Böylece alt sınıflar, üst sınıfın belirlediği davranışı **uygun olmayan biçimde** değiştiremezler!

İletişim

JAVA Eğitimleri

Kurumsal Java Eğitimleri

www.java-egitimleri.com

info@java-egitimleri.com

@javaegitimleri

youtube.com/c/
KurumsalJavaEğitimleri