组合实体模式 前端控制器模式

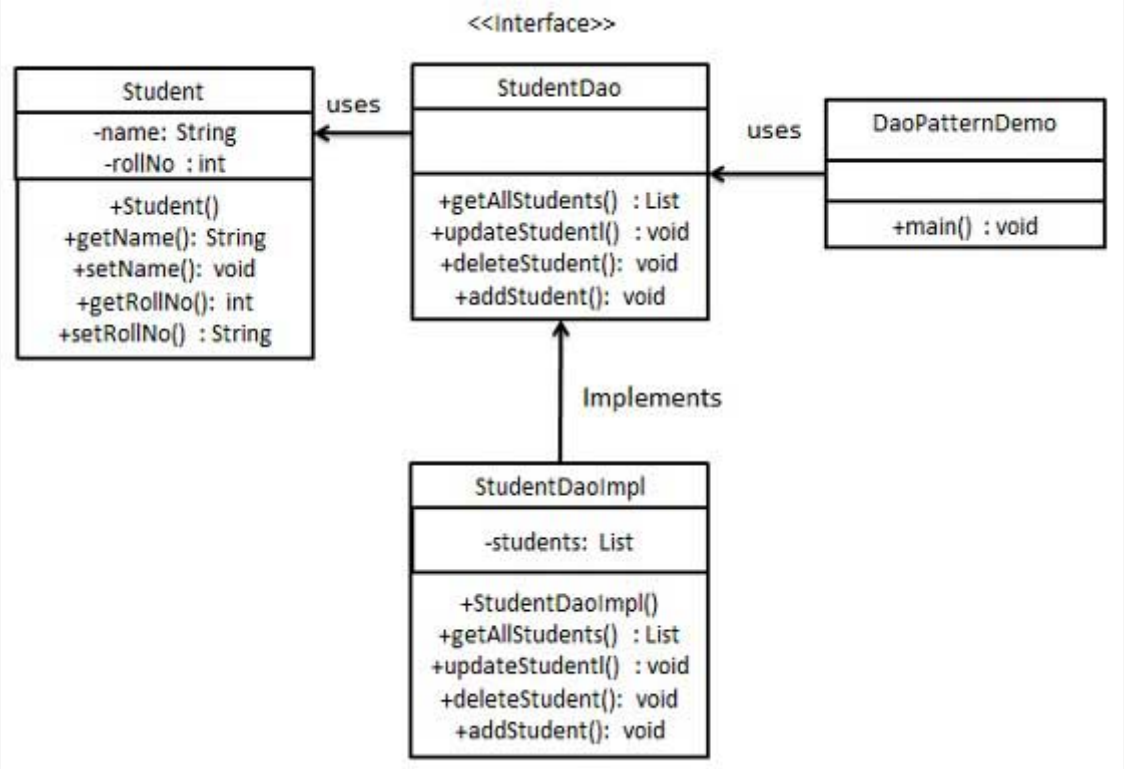
数据访问对象模式

数据访问对象模式（Data Access Object Pattern）或 DAO 模式用于把低级的数据访问 API 或操作从高级的业务服务中分离出来。以下是数据访问对象模式的参与者。

- **数据访问对象接口（Data Access Object Interface）** - 该接口定义了在一个模型对象上要执行的标准操作。
- **数据访问对象实体类（Data Access Object concrete class）** - 该类实现了上述的接口。该类负责从数据源获取数据，数据源可以是数据库，也可以是 xml，或者是其他的存储机制。
- **模型对象/数值对象（Model Object/Value Object）** - 该对象是简单的 POJO，包含了 get/set 方法来存储通过使用 DAO 类检索到的数据。

实现

我们将创建一个作为模型对象或数值对象的 *Student* 对象。*StudentDao* 是数据访问对象接口。*StudentDaoImpl* 是实现了数据访问对象接口的实体类。*DaoPatternDemo*，我们的演示类使用 *StudentDao* 来演示数据访问对象模式的用法。



步骤 1

创建数值对象。

Student.java

```
public class Student {
private String name;
private int rollNo;
Student(String name, int rollNo){
this.name = name;
```

```
this.rollNo = rollNo;
}
public String getName() {
return name;
}
public void setName(String name) {
this.name = name;
}
public int getRollNo() {
return rollNo;
}
public void setRollNo(int rollNo) {
this.rollNo = rollNo;
}
}
```

步骤 2

创建数据访问对象接口。

StudentDao.java

```
import java.util.List;
public interface StudentDao {
public List<Student> getAllStudents();
public Student getStudent(int rollNo);
public void updateStudent(Student student);
public void deleteStudent(Student student);
}
```

步骤 3

创建实现了上述接口的实体类。

StudentDaoImpl.java

```
import java.util.ArrayList;
import java.util.List;
public class StudentDaoImpl implements StudentDao {
//列表是当作一个数据库
List<Student> students;
public StudentDaoImpl(){
students = new ArrayList<Student>();
Student student1 = new Student("Robert",0);
Student student2 = new Student("John",1);
students.add(student1);
students.add(student2);
}
@Override
public void deleteStudent(Student student) {
students.remove(student.getRollNo());
System.out.println("Student: Roll No " + student.getRollNo()
+", deleted from database");
}
```

```
//从数据库中检索学生名单
@Override
public List<Student> getAllStudents() {
return students;
}
@Override
public Student getStudent(int rollNo) {
return students.get(rollNo);
}
@Override
public void updateStudent(Student student) {
students.get(student.getRollNo()).setName(student.getName());
System.out.println("Student: Roll No " + student.getRollNo()
+", updated in the database");
}
}
```

步骤 4

使用 *StudentDao* 来演示数据访问对象模式的用法。

DaoPatternDemo.java

```
public class DaoPatternDemo {
public static void main(String[] args) {
StudentDao studentDao = new StudentDaoImpl();
//输出所有的学生
for (Student student : studentDao.getAllStudents()) {
System.out.println("Student: [RollNo : "
+student.getRollNo()+", Name : "+student.getName()+" ]");
}
//更新学生
Student student =studentDao.getAllStudents().get(0);
student.setName("Michael");
studentDao.updateStudent(student);
//获取学生
studentDao.getStudent(0);
System.out.println("Student: [RollNo : "
+student.getRollNo()+", Name : "+student.getName()+" ]");
}
}
```

步骤 5

执行程序，输出结果：

```
Student: [RollNo : 0, Name : Robert ]
Student: [RollNo : 1, Name : John ]
Student: Roll No 0, updated in the database
Student: [RollNo : 0, Name : Michael ]
```

组合实体模式 前端控制器模式

点我分享笔记