UT03_04 Patrón DAO (Tutorialspoint http://www.tutorialspoint.com)

Traducción libre de: https://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

1. Introducción

El patrón "Data Access Object" (DA) se usa para definir servicios de acceso a datos simples, de alto nivel, aislando la complejidad de la lógica del API de bajo nivel utilizado para acceder a la base de datos.

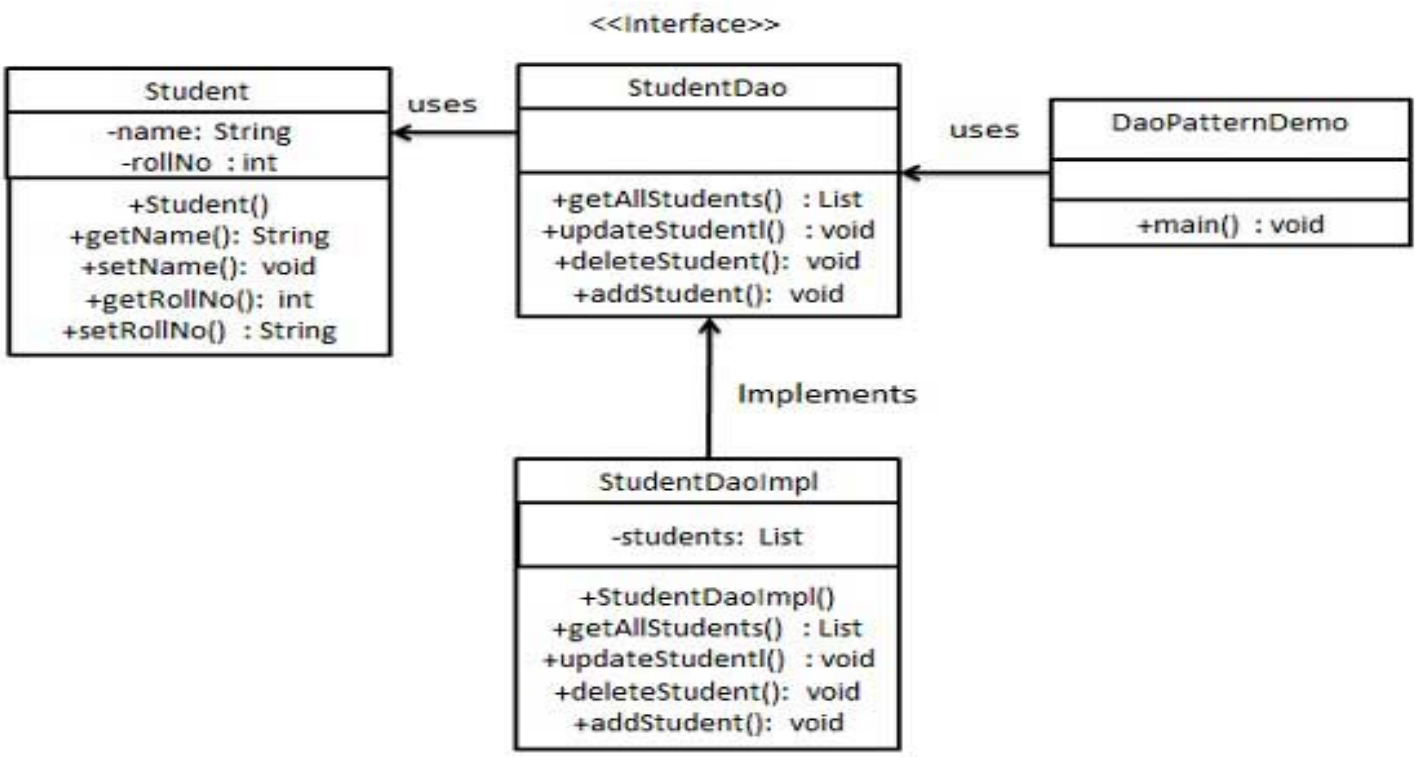
En este patrón identificamos los siguientes participantes:

- **Data Access Object interface** – Interfaz que define las operaciones que se van a realizar sobre el modelo de objetos.
- **Data Access Object clase implementación** – Clase que implementa la interfaz anterior. Es responsable de obtener los datos del origen de datos, que puede ser una base de datos, XML o cualquier otro tipo de almacenamiento.
- **Value Object o Model Object** – Esta clase es un simple POJO (Plain Old Java Object: objeto muy simple con propiedades y métodos getter/setter para acceder a las propiedades). Se utiliza para contener los datos que se intercambian con el origen de datos.

2. Implementación

Crearemos un objeto POJO de tipo Student que va a actuar como Value Object (almacena los campos intercambiados con la BD). StudentDao es la interfaz que define los métodos/operaciones a realizar sobre el modelo. StudentDaoImpl es la clase concreta que implementa los métodos anteriores (proporciona cuerpo, código para ellos).

La clase DaoPatternDemo es nuestra clase demo. Utilizaremos StudentDao para ver cómo se usa el patrón DAO (Data Access Object).



2.1 Paso 1

Crear el Value Object.

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;
   }
}
```

2.2 Paso 2

Crear la interfaz Data Access Object.

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);
}
```

2.3 Paso 3

Crear la clase de que implementa la interfaz anterior (clase concreta).

StudentDaoImpl.java

```
import java.util.ArrayList;
import java.util.List;

public class StudentDaoImpl implements StudentDao {

   //list is working as a database
   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");
   }

   //retrive list of students from the database
   @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");
   }
}
```

2.4 Paso 4

Usar *StudentDao* para demostrar el uso del patrón DAO (Data Access).

DaoPatternDemo.java

```
public class DaoPatternDemo {
   public static void main(String[] args) {
      StudentDao studentDao = new StudentDaoImpl();

      //print all students
      for (Student student : studentDao.getAllStudents()) {
          System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " +
student.getName() + " ]");
      }


      //update student
      Student student =studentDao.getAllStudents().get(0);
      student.setName("Michael");
      studentDao.updateStudent(student);

      //get the student
      studentDao.getStudent(0);
        System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " +
student.getName() + " ]");
   }
}
```

2.5 Paso 5

Verificar resultado.

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