Implementing Design Pattern in Java

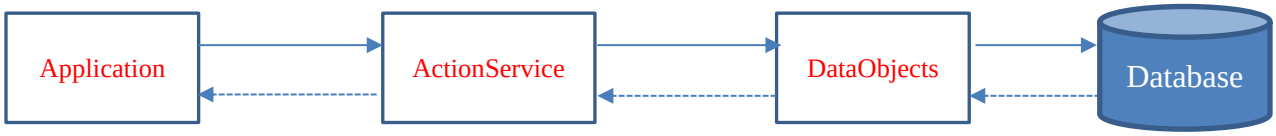
❖ Design Pattern là gì?

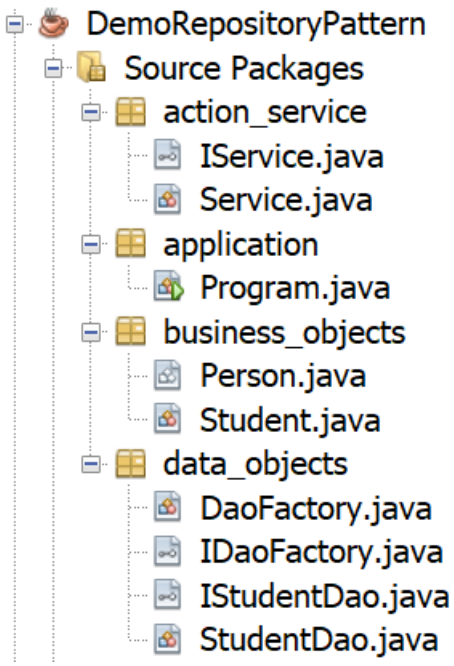
- **Design Pattern** là một kỹ thuật trong lập trình hướng đối tượng, cung cấp cho chúng ta cách tư duy trong các tình huống xảy ra của lập trình hướng đối tượng cũng như trong quá trình phân tích thiết kế và phát triển phần mềm. Vì vậy **Design Pattern** không phải là một Class, cũng không phải là một Library, và cũng không hề là một ngôn ngữ cụ thể nào cả.
- **Design Pattern** cung cấp cho chúng ta các **mẫu thiết kế**, các giải pháp cho các vấn đề chung thường gặp trong lập trình, đảm bảo sẽ cung cấp cho chúng ta các giải pháp tối ưu trong việc giải quyết các vấn đề trong lập trình.

❖ Vì sao nên sử dụng Design Pattern?

- **Design Pattern** cung cấp cho chúng ta các giải pháp ở dạng tổng quát nhất, giúp chúng ta tăng tốc độ phát triển phần mềm thông qua các mô hình đã được kiểm nghiệm thực tế.
- Sử dụng **Design Pattern** giúp chúng ta tránh được các lỗi tiềm ẩn (nhất là trong những hệ thống lớn), đồng thời có khả năng tái sử dụng cao để có thể dễ nâng cấp và bảo trì trong tương lai.
- Trong bài lab này chúng ta sử dụng các design pattern : DAO pattern, Factory pattern và Repository pattern.
- Tham khảo thêm các design pattern : https://www.tutorialspoint.com/design_pattern/index.htm

Bước 1 . Tạo ứng dụng Java tên LabDesignPattern và có cấu trúc như sau





Bước 2 Viết code cho các tập tin .java trong package **business_objects** như sau :

2.1 **Person.java**

```
package business_layer;
/**...4 lines */
public abstract class Person {
    protected String name;
    protected int id;
    public Person() {
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
}
```

2.2 **Student.java**

```java
package business_layer;
/**...4 lines */
public class Student extends Person{
   private String email;
   public Student(int id, String name, String email){
      this.name = name;
      this.id = id;
      this.email = email;
   }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    @Override
    public String toString() {
        return String.format("Id:%d,Name:%s,email:%s",id,name,email);
    }
}
```

Bước 3 Viết code cho các tập tin .java trong package **data_objects** như sau :

3.1 **IStudentDao.java**

```java
package dal.dao;

...4 lines
import business_layer.Student;
import java.util.List;
// defines methods to access orders.
// this is a database-independent interface.
// Implementations are database specific
// ** DAO Pattern
public interface IStudentDao {
   public List<Student> getAllStudents();
   public Student getStudent(int id);
   public void updateStudent(Student student);
   public void deleteStudent(Student student);
}
```

3.2 **StudentDao.java**

```java
package dal.dao;

import business_layer.Student;
import java.util.ArrayList;
import java.util.List;
 // Data access object for Category
// ** DAO Pattern
public class StudentDao implements IStudentDao {
   //list is working as a database
   List<Student> students;
   public StudentDao(){
      students = new ArrayList<Student>();
      Student Robert = new Student(1,"Robert","Robert@gmail.com");
      Student John = new Student(2,"John","John@gmail.com");
      students.add(Robert);
      students.add(John);
   }
   @Override
   public void deleteStudent(Student student) {
      students.remove(student);
      System.out.println("Student: ID:" + student.getId()+ ", deleted from database");
   }

   //retrive list of students from the database
   @Override
   public List<Student> getAllStudents() {
      return students;
   }
   @Override
   public Student getStudent(int id) {
      Student student = students.stream()
        .filter(s -> s.getId()==id)
        .findAny()
        .orElse(null);
      return student;
   }
   @Override
   public void updateStudent(Student student) {
      var std = getStudent(student.getId());
      std = student;
      System.out.println("Student: ID " + student.getId()+ ", updated into the database");
   }
}
```

3.3 **IDaoFactory.java**

```java
package dal.dao;
// abstract factory interface. Creates data access objects.
// ** GoF Design Pattern: Factory.
public interface IDaoFactory {
    IStudentDao studentDao();
    //IStaffDao staff();
    //.....
}
```

3.4 **DaoFactory.java**

```
package dal.dao;
 // Data access object factory
 // ** Factory Pattern
public class DaoFactory implements IDaoFactory{
    @Override
    public IStudentDao studentDao() {
         return new StudentDao();
    }
    //Write codes for Teacher, Staff, Course and so on
}
```

Bước 4 Viết code cho các tập tin .java trong package **action_service** như sau :

4.1 **IService.java**

```
package action_service;
import business_layer.Student;
import java.util.List;

// single interface to all 'repositories'
public interface IService {
     // Student Repository
     void deleteStudent(Student student);
     List<Student> getAllStudents();
     Student getStudent(int id) ;
     void updateStudent(Student student) ;
     //The other repositories
     //.....
}
```

4.2 **Service.java**

```java
import java.util.List;
/**...4 lines */
public class Service implements IService{
    static final IDaoFactory factory = new DaoFactory();
    static final IStudentDao studentDao = factory.studentDao() ;
    //Student
    @Override
    public void deleteStudent(Student student) {
        studentDao.deleteStudent(student);
    }
    @Override
    public List<Student> getAllStudents() {
       return studentDao.getAllStudents();
    }
    @Override
    public Student getStudent(int id) {
        return studentDao.getStudent(id);
    }
    @Override
    public void updateStudent(Student student) {
        studentDao.updateStudent(student);
    }
    //.......
}
```

Bước 5 Viết code cho tập tin **Program.java** trong package **application** như sau và chạy ứng dụng

```java
package app;

import action_service.Service;
import business_layer.Student;
import java.util.List;
/**
 * @author Sword Lake
 */
public class Program {
    static void printList(List<Student> list){
       for(var student : list){
            System.out.println(student);
       }
    }
    public static void main(String[] args) {
        Service service = new Service();
        System.out.println("********Student List*******");
        printList(service.getAllStudents());
```

```
        System.out.println("********Update Student*****");
        var student = service.getStudent(1);
        student.setName("David");
        service.updateStudent(student);
        printList(service.getAllStudents());
        System.out.println("********Delete Student*****");
        service.deleteStudent(student);
        printList(service.getAllStudents());
    }
}
```

```
-----------------------------------------

********Student List*******
Id:1,Name:Robert,email:Robert@gmail.com
Id:2,Name:John,email:John@gmail.com
********Update Student*****
Student: ID 1, updated in the database
Id:1,Name:David,email:Robert@gmail.com
Id:2,Name:John,email:John@gmail.com
********Delete Student*****
Student: ID:1, deleted from database
Id:2,Name:John,email:John@gmail.com
```