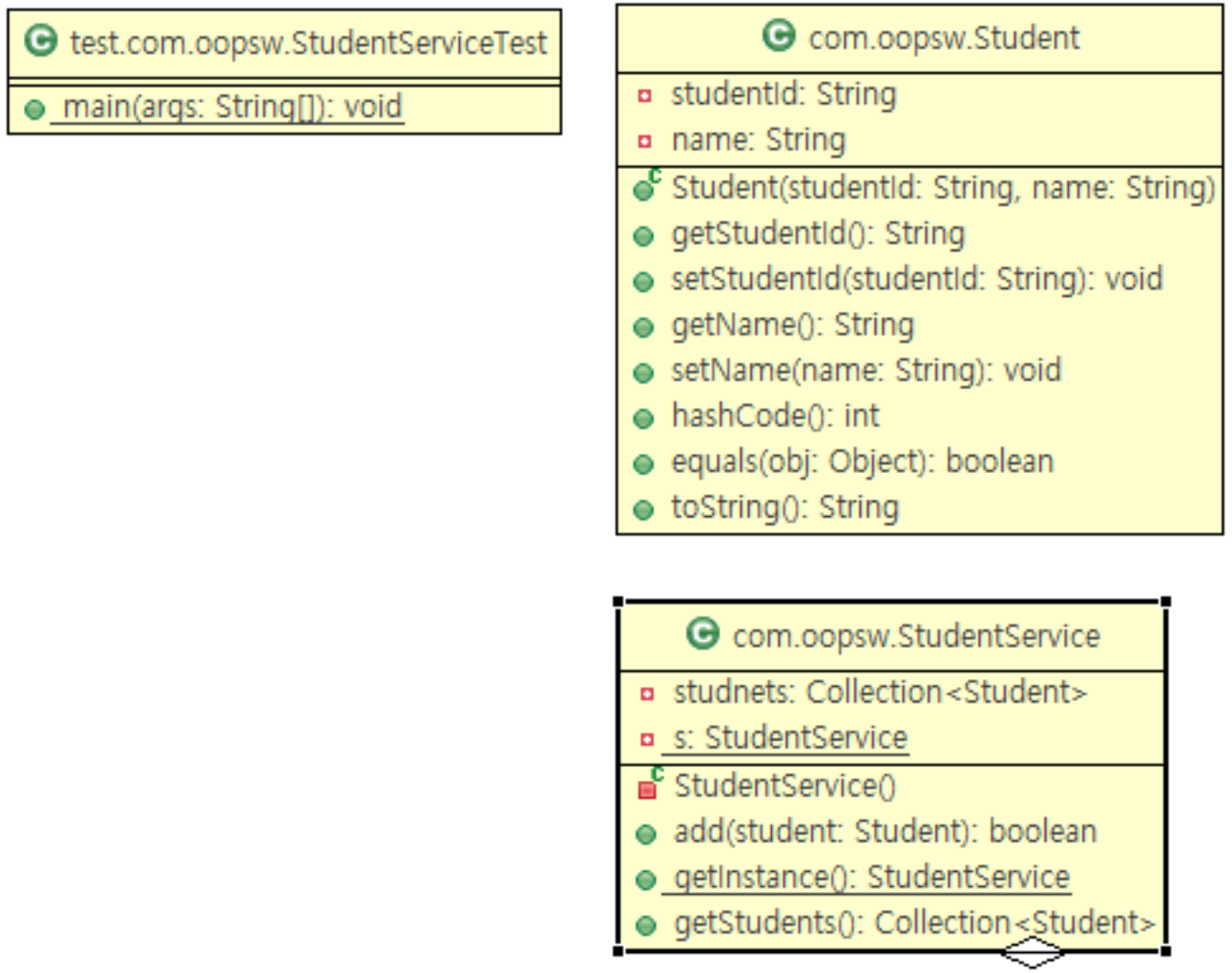
OOP 디자인패턴 #3

A014손은빈

생성패턴 중 싱글턴패턴 사용하기

ClassDiagram



프로그램 소스

Student.java -->
- 학번, 이름 필드를 가짐
- set메서드를 재사용해 초기화함
- setName에서 이름 길이를 검사해 예외를 던짐

```java
package com.oopsw;

public class Student {

    private String studentId;
    private String name;

    public Student(String studentId, String name) {
        super();
        setStudentId(studentId);
        setName(name);
    }

    public String getStudentId() {
        return studentId;
    }
    public void setStudentId(String studentId) {
        this.studentId = studentId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        if(name.length() < 2 || name.length() > 4)
            throw new RuntimeException("이름은 2-4자까지 허용됩니다.");
        this.name = name;
    }

```

프로그램 소스

Student.java -->
- 객체의 주소와 값을 비교할 때, 학번만 대상으로 해서 중복 학번의 학생 데이터는 저장되지 않도록 함 (HashSet 메모리를 사용함)

```
29      @Override
30      public int hashCode() {
31          final int prime = 31;
32          int result = 1;
33  //      result = prime * result + ((name == null) ? 0 : name.hashCode());
34          result = prime * result + ((studentId == null) ? 0 : studentId.hashCode());
35          return result;
36      }
37      @Override
38      public boolean equals(Object obj) {
39          if (this == obj)
40              return true;
41          if (obj == null)
42              return false;
43          if (getClass() != obj.getClass())
44              return false;
45          Student other = (Student) obj;
46  //      if (name == null) {
47  //          if (other.name != null)
48  //              return false;
49  //      } else if (!name.equals(other.name))
50  //          return false;
51          if (studentId == null) {
52              if (other.studentId != null)
53                  return false;
54          } else if (!studentId.equals(other.studentId))
55              return false;
56          return true;
57      }
58      @Override
59      public String toString() {
60          return "Student [studentId=" + studentId + ", name=" + name + "]";
61      }
```

프로그램 소스

StudentService.java -->
- 자기자신을 멤버로 갖되 static 공유 메모리 영역에 등록함.
static 영역의 공유 데이터는 풀 영역과 달리 수정이 가능함.
- 생성자를 private로 접근제한하고 static영역에 getInstance메서드를 등록해서 메모리에 객체가 한번만 등록되고 유지되게함.

==> 싱글턴 패턴

```java
package com.oopsw;

import java.util.ArrayList;

public class StudentService {

    private Collection<Student> studnets;
    //a3)
    private static StudentService s;

    //a1)생성자를 막고 (new) 1번만 메모리에 등록해서 쓰고싶다.
    private StudentService() {
        studnets=new HashSet<Student>();
    }
    /* 이름을 여러개 등록이 가능 - 단 이름은 중복 저장하지 않는다. */
    public boolean add(Student student) {
        return studnets.add(student);
    }
    //a2)
    public static StudentService getInstance() {
        if(s==null) s=new StudentService(); // 메모리에 없으면 올리고, 있으면 리턴함

        return s;
    }
    /* 기존 이름에서 새로 입력할 이름과 같은지 여부를 확인 */
    // 데이터 클래스에서 equals, hashCode를 오버라이딩해서 구현함
    /* 등록된 모든 이름을 확인 */
    public Collection<Student> getStudents(){
        return studnets;
    }

}
```

프로그램 소스

StudentServiceTest.java-->
- 다수의 사용자가 데이터를 저장하기 위해 메모리에 접근하는 상황을 가정함.
메모리는 같은 객체에 한번만 등록되어야 함.
- 중복 학번을 불허하고, 중복 이름은 허용하는지 확인함, 중복 학번 저장 시도 시에 예외가 잘 발생하는지 확인함

```java
package test.com.oopsw;

import com.oopsw.Student;
import com.oopsw.StudentService;
import com.oopsw.StudentService;

public class StudentServiceTest {
    public static void main(String[] args) {
//      StudentService s1=new StudentServiceImpl();
//      StudentService s1=new StudentService("학생학번", "학생이름");
//      StudentService s1=new StudentService("200101001", "홍길동");
//
//      StudentService studentList=new StudentService();
        try {
            StudentService client1=StudentService.getInstance();
            StudentService client2=StudentService.getInstance();

            client1.add(new Student("s0001", "홍길동"));
            client2.add(new Student("s0002", "홍길동"));
            client2.add(new Student("s0002", "홍길동"));

            System.out.println(client1.getStudents());
            System.out.println(client1);
            System.out.println(client2);
        }catch(RuntimeException e) {
            System.out.println(e.getMessage());
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

테스트 실행 결과

- 중복 학번은 불허하고 중복 이름은 허용함.
- 인스턴스를 두번 생상 시도했지만,
같은 객체임

```java
package test.com.oopsw;

import com.oopsw.Student;
import com.oopsw.StudentService;
import com.oopsw.StudentService;

public class StudentServiceTest {
    public static void main(String[] args) {
//      StudentService s1=new StudentServiceImpl();
//      StudentService s1=new StudentService("학생학번", "학생이름");
//      StudentService s1=new StudentService("200101001", "홍길동");
//      StudentService studentList=new StudentService();
        try {
            StudentService client1=StudentService.getInstance();
            StudentService client2=StudentService.getInstance();

            client1.add(new Student("s0001", "김학생"));
            client2.add(new Student("s0002", "김학생")); // 중복 이름 허용 테스트
            client2.add(new Student("s0002", "박학생"));

            System.out.println(client1.getStudents());
            System.out.println(client1);
            System.out.println(client2);
        }catch(RuntimeException e) {
            System.out.println(e.getMessage());
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Console

<terminated> StudentServiceTest [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 3. 28. 오후 7:26:54)

```
[Student [studentId=s0001, name=김학생], Student [studentId=s0002, name=김학생]]
com.oopsw.StudentService@15db9742
com.oopsw.StudentService@15db9742
```

테스트 실행 결과

- 이름 길이가 2-4자가 아닐 때, 예외발생함

```java
package test.com.oopsw;

import com.oopsw.Student;
import com.oopsw.StudentService;
import com.oopsw.StudentService;

public class StudentServiceTest {
    public static void main(String[] args) {
//      StudentService s1=new StudentServiceImpl();
//      StudentService s1=new StudentService("학생학번", "학생이름");
//      StudentService s1=new StudentService("200101001", "홍길동");
//
//      StudentService studentList=new StudentService();
        try {
            StudentService client1=StudentService.getInstance();
            StudentService client2=StudentService.getInstance();

            client1.add(new Student("s0001", "김학생김학생")); // 2-4 이름길이예외 테스트
            client2.add(new Student("s0002", "박학생"));
            client2.add(new Student("s0002", "박학생"));

            System.out.println(client1.getStudents());
            System.out.println(client1);
            System.out.println(client2);
        }catch(RuntimeException e) {
            System.out.println(e.getMessage());
        }catch(Exception e) {
            System.out.println(e.getMessage());
        }

    }
}
```

```
Console
<terminated> StudentServiceTest [Java Application] C:₩Program Files₩Java₩jdk1.8.0_202₩bin₩javaw.exe (2022. 3. 28. 오후 7:24:35)
이름은 2-4자까지 허용됩니다.
```