Prac 12) Junit testing

GradeCalculatorTest.java

```
package com.example.StudentGrade;

import org.junit.jupiter.api.Test;
import com.example.StudentGrade.service.GradeCalculator;
import static org.junit.jupiter.api.Assertions.*;

public class GradeCalculatorTest {

   private final GradeCalculator calc = new GradeCalculator();

   @Test
   void testGradeCalculation() {
      assertEquals("A+", calc.calculateGrade(95), "95 should be A+");
      assertEquals("A", calc.calculateGrade(80), "80 should be A");
      assertEquals("B", calc.calculateGrade(65), "65 should be B");
      assertEquals("C", calc.calculateGrade(45), "45 should be C");
      assertEquals("F", calc.calculateGrade(35), "35 should be F");
      assertEquals("Invalid", calc.calculateGrade(-1), "-1 should be Invalid");
      assertEquals("Invalid", calc.calculateGrade(150), "150 should be Invalid");
   }
}
```

StudentServiceTest.java

```
package com.example.StudentGrade;

import com.example.StudentGrade.model.Student;
import com.example.StudentGrade.repository.StudentRepository;
import com.example.StudentGrade.service.GradeCalculator;
import com.example.StudentGrade.service.StudentService;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

public class StudentServiceTest {

   @Mock
   private StudentRepository repository;

   @Mock
   private GradeCalculator gradeCalculator;

   @InjectMocks
   private StudentService service;
```

```
    @BeforeEach
    void setUp() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    void testAddStudent() {
        Student s = new Student();
        s.setName("Ravi");
        s.setMarks(85);

        when(gradeCalculator.calculateGrade(85)).thenReturn("A");
        when(repository.save(any(Student.class))).thenReturn(s);

        Student result = service.add(s);

        assertEquals("A", result.getGrade());
        verify(repository, times(1)).save(s);
    }

    @Test
    void testGetStudentById() {
        Student s = new Student();
        s.setId(1);
        s.setName("Meena");
        s.setMarks(90);
        s.setGrade("A+");

        when(repository.findById(1)).thenReturn(Optional.of(s));

        Student found = service.getById(1);

        assertNotNull(found);
        assertEquals("Meena", found.getName());
        verify(repository, times(1)).findById(1);
    }

    @Test
    void testDeleteStudent() {
        service.delete(1);
        verify(repository, times(1)).deleteById(1);
    }
}
```