Лабораторна робота №2
Створення модульних тестів JUnit 5.

Для варіантів лабораторної роботи №1 забезпечити покриття тестами не менше ніж 80 відсотків, як позитивного так і негативного сценаріїв.

Для підключення JUnit 5 найкраще створити проект Maven або Gradle. Додати наступні залежності:

```
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
     </dependency>
     <dependency>
      <groupId>org.junit.vintage</groupId>
      <artifactId>junit-vintage-engine</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
     </dependency>

     <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-engine</artifactId>
      <version>${junit.version}</version>
      <scope>test</scope>
     </dependency>
```

Версію JUnit можна вказати в розділі properties

```
<properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
   <maven.compiler.source>1.8</maven.compiler.source>
   <maven.compiler.target>1.8</maven.compiler.target>
   <junit5.version>5.3.2</junit5.version>
 </properties>
```

Також необхідно підключити plugin

```
<plugin>
 <artifactId>maven-surefire-plugin</artifactId>
 <version>3.0.0-M3</version>
</plugin>
```

Приклад класу та тестів.

```
import java.util.Optional;

public class CalculatorService {
   private static CalculatorService instance = new CalculatorService();
   public static CalculatorService getInstance(){
      return instance;
   }

   public int add(int first, int second){
      return first + second;
   }

   public int sub(int first, int second) {
      return first - second;
   }

   public Optional<Integer> div(int first, int second){
      Optional<Integer> result = Optional.empty();
      if( second != 0){
         result = Optional.of(first / second);
      }
      return result;
   }
}
```

```
import org.itstep.calculator.model.CalculatorService;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

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

public class CalculatorServiceTest {

   public CalculatorService calculatorService = CalculatorService.getInstance();

   @Test
   @DisplayName("new test for add")
   public void add() {
      assertEquals(calculatorService.add(3, 5), 8);
      assertNotEquals(calculatorService.add(3, 2), 3);
   }


   @Test
   @DisplayName("new test for sub")
```

```
    public void sub() {
        assertEquals(calculatorService.sub(5, 3), 2);
        assertNotEquals(calculatorService.sub(3, 2), 3);
    }
}
```

Контрольні запитання.

1. Що таке модульне тестування?
2. Які переваги та недоліки модульного тестування?
3. Як порахувати відсоток покриття тестами?
4. Які обмеження модульного тестування?