Практичне заняття №2

Тема: Модульне тестування в Java.

Мета: Вивчити особливості роботи з утилітою **JUnit**.

Завдання: Здобути навички створення та відлагодження проектів модульних тестів у **JUnit**.

(max: 5 балів)

Теоретичні відомості:

Тестування є невід'ємною частиною процесу розробки програмного забезпечення. Метою тестування є перевірка правильності роботи програми, і без цього етапу експлуатація розроблених програмних засобів була б або неможливою, або надто ризикованою.

Розрізняють різні типи тестування. Зокрема, м***одульне тестування*** спрямовано на перевірку правильності роботи окремих класів та функцій. Метою ж ***інтеграційнего тестування*** є перевірка роботи всього програмного комплексу в цілому.

Типова схема модульного тестування полягає в перевірці роботи модулю на основі заздалегідь підібраних ***тестів*** – наборів даних, для яких відомі правильні результати. Розбіжність між очікуваними та фактично отриманими результатами сигналізує про наявність помилки. Співпадіння результатів не доводить правильності програми, але підвищує міру впевненості у відсутності помилок.

Чим повнішим і більш представницьким є набір тестів, тим більше підстав довіряти програмі. Тестові приклади мають покривати, по-перше, можливий діапазон вхіднихданих, а за можливості – варіанти роботи програми відповідно до її логіки.

В ряді методик розробки програмного забезпечення тестування взагалі відіграє ключову роль. Наприклад, ***TDD*** (***Test-Driven Development***, розробка на основі тестів) передбачає спочатку написання тестів, а вже потім реального коду.

Очевидним підходом до модульного тестування є просте виведення на екран фактичних результатів роботи програми поряд з правильними. Але перегляд виведення з метою співставлення результатів не є особливо зручним. Крім того, в коді з'являється багато операцій виведення, які потім стануть непотрібними. Гарною ідеєю було б, по-перше, відділити основний робочий код від запуску тестів, а по-друге – якимось чином автоматизувати сам процес перевірки. Існує ряд утиліт, призначених для автоматизації процесу модульного тестування. Широкого розповсюдження набула утиліта **JUnit**, якуможна використовувати автономно або в складі інтегрованих середовищ.

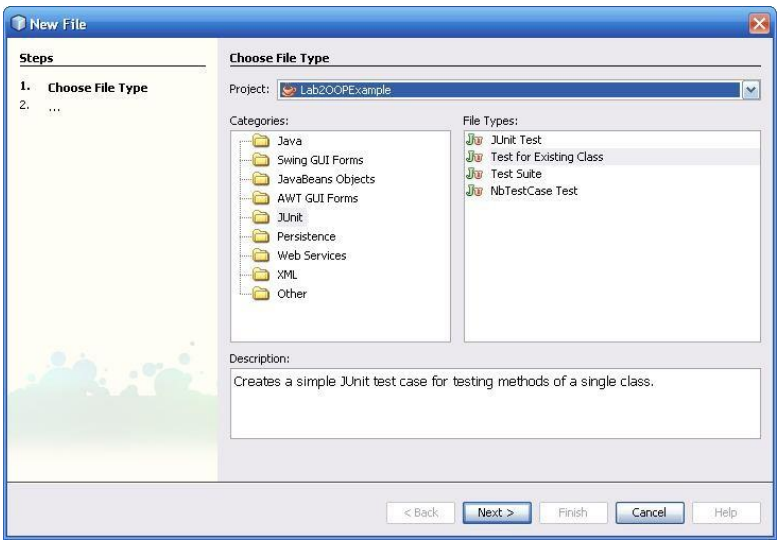
Розглянемо, як створюються та виконуються тести на прикладі найпростішої програми. Для того, щоб виконувати наступні кроки треба створити головний клас програми і додати до нього хоча б один метод.

Нехай у нас є проект *Example1*, що складається з пакету **example1** в якому міститься єдиний клас Main (у файлі Main.java).

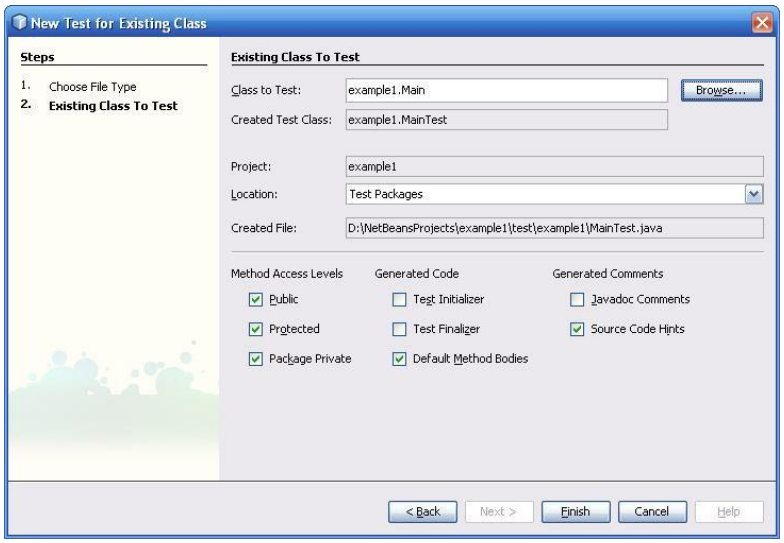
```
package example1;

/**
 * @author Eugeny
 */
public class Main {

    /**
     * Метод, що обчислює значення y за формулоюз завдання
     * @param a перший параметр
     * @param b другий параметр
     * @param x третій параметр
     * @return обчислене значення y
     */
```

```
        public double calcY(double a, double b, double x) {return
       (Math.pow(a, 2 * x) + Math.pow(b, -x) *Math.cos(a + b) *
                                                    x) / (x + 1);
    }
    /**
     * Метод, що обчислює значення r за формулоюз завдання
     * @param a перший параметр
     * @param b другий параметр
     * @param x третій параметр
     * @return обчислене значення r
     */
  public double calcR(double a, double b, double x) {return
          Math.sqrt(x * x + b - b * b *
               Math.pow(Math.sin(x + a), 3) / x);
     }
     /**
      * метод присвоює a,b,x значення з завдання
       * та обчислює значення змінних r та y –calcR() i
calcY()
       * Після того выводяться початкові та об-числені
значення
      */
     public void printResults() {double a =
         0.3;
         double b = 0.9;double x =
         0.61;
         double r = calcY(a, b, x);double y =
         calcR(a, b, x);
             System.out.printf("x=%5.3f  y=%5.3fz=%5.3f%n", a,
b, x);
         System.out.printf("y=%8.4f%n",  y);
         System.out.printf("r=%8.4f%n",  r);
     }
     /**
      * @param args аргументи командного рядка
      */
     public static void main(String[] args) {Main program
         = new Main(); program.printResults();
     }
  }
```

Для створення класу модульного тестування у середовищі NetBeans оберіть пункт New File… у меню File. У вікні, що відкриється, обрати категорію JUnit і в цій категорії тип файлу Test for Existing Class. Для продовження треба натиснути кнопку "Next >". Відкриється наступне вікно.



Укажіть в ньому, для якого класу створюються тестовий клас (**Class to Test:**). Для обрання потрібного класу натисніть кнопку **Browse…** та оберіть потрібний клас. В нашому випадку це клас ***example1.Main.*** Після налаштування створюваного тестового класу треба натиснути "Finish". Якщо Ви робите це перший раз, то буде запропоновано вибрати версію бібліотеки модульного тестування JUnit 4.x або JUnit 5.x. Оберемо JUnit 4.x.



Буде створено клас (у нашому випадку це буде клас MainTest), який буде розміщено у розділі Test Packages.

При створенні тестового класу, середовище NetBeans формує шаб- лони тестових методів, їх треба відредагувати таким чином:

- Метод для тестування метода main() (він буде називатися testMain()) можна (або навіть, треба) видалити.
- Оскільки метод printResults() лише викликає інші методи і виводить результати їх виконання, то його можна не тестувати (тобто метод testPrintResults() теж можна видалити).
- На початку тестового класу додайте описання константи EPS – точності для порівнянь.
- Методи testCalcY() та testCalcR() треба змінити так, щоб за їх допомогою можна було тестувати відповідні методи головного класу програми. Спочатку треба видалити останні два рядки кожного мето- да – вони були додані туди тільки для того, щоб пересвідчитися, що тес- ти насправді писалися людиною, а не були використані шаблони.
- У кожному з методів впишіть початкові значення, що треба при-

своїти змінним, які передаються цим методам (в нашому випадку – змінні a, b, x). Ці значення треба підібрати таким чином, щоб можна було про- тестувати різні випадки в процесі обчислення функцій.

• Змінній **expResult** треба присвоїти значення, що очікується у ре- зультаті виконання метода, що буде тестуватися (його треба обчислити вручну, або за допомогою калькулятора).

• Після виклику метода, що тестується, змінній **result** буде присвоєне значення, яке обчислить метод.

• Значення змінних **expResult** та **result** треба порівняти за допомо- гою методу assertEquals(arg1,arg2,eps), де arg1, arg2 – значення, що по- рівнюються, eps – похибка порівняння. Цей метод перевіряє, чи вико- нується нерівність $|arg1 - arg2| < eps$, тобто якщо eps – мале число, то можна вважати, що arg1 та arg2 приблизно рівні.

• Для кожного з методів класу, що тестується, можна (і треба) ство- рити декілька тестових методів (якщо ж в методі, який тестується, є оператори розгалуження, то потрібно створити стільки тестів, скільки є різних шляхів виконання метода, а крім того, ОБОВ'ЯЗКОВО ТРЕБА перевірити умови на межі переходу в умовному операторі).

• Виконати тести можна обравши в головному меню NetBeans у пункті Run підпункт Test "<Ім'я проекту>" (в нашому прикладі – Test"example1".

• Після виконання тесту у вікні тестування будуть показані результати: **passed** (зеленим кольором) означає, що відповідний тест "пройшов", **FAILED** (червоним кольором) означає, що тест "не пройшов" і треба перевірити, чому так сталося, та знайти помилку. Якщо хоча б один тест не пройшов, то вважається, що проект в цілому не пройшов тестування і не може здаватися, як завершений продукт. Треба виправляти помилки до тих пір, поки ВСІ тести будуть "проходити".

• Відмітимо, що тестовий клас найчастіше виходить більшим, ніжтой клас, який він тестує, проте тестовий клас складається з простих методів і тому його розроблювати нескладно.

Приклад тестового класу:

```
package example1;

import org.junit.AfterClass;import
org.junit.BeforeClass;import org.junit.Test;
import  static  org.junit.Assert.*;

       /**
        *
 * @author Eugeny
 */
public class MainTest {

    public static final double EPS = 1e-6;
    // припустима точність порівнянь

    public MainTest() {
    }

    @BeforeClass
     public  static  void  setUpClass()  throwsException {
    }

    @AfterClass
     public static void tearDownClass() throwsException {
  }
    @Test // Тест загального випадку, очікуєморезультат
1.0
```

```
        public void testCalcY() { System.out.println("calcY (1.0, -
        1.0, 0.0)");
            double a = 1.0;double b = -
            1.0;double x = 0.0;
            Main instance = new Main();double
            expResult = 1.0;
           double result = instance.calcY(a, b, x);
            assertEquals(expResult, result, EPS);
        }

        @Test
    // Тест окремого випадку, очікуємо результат
 "+НЕСКІНЧЕНОСТЬ"
        public void testCalcY0() { System.out.println("calcY (1.0, -1.0,0.0)");


                                                                    }

 double a = 1.0;double b = -1.0;double x
 = -1.0;
 Main instance = new Main();
double expResult = Double.POSITIVE_INFINITY; double result =
instance.calcY(a, b, x);assertEquals(expResult, result, EPS);

        @Test // Тест загального випадку, очікуємо результат
 2.0
        public void testCalcR() { System.out.println("calcR
        (0.0, 0.0, 2.0)");
            double a = 0.0; double b =
            0.0;double x = 2.0;
            Main instance = new Main();double
            expResult = 2.0;
            double result = instance.calcR(a, b, x);
           assertEquals(expResult, result, EPS);
        }

        @Test
    // Тест окремого випадку, очікуємо результат"НЕ-ЧИСЛО",
 0/0
        public void testCalcR0() { System.out.println("calcR
        (0.0, 0.0, 0.0)");
            double a = 0.0; double b =
            0.0;double x = 0.0;
            Main instance = new Main(); double
            expResult = Double.NaN;
            double result = instance.calcR(a, b, x);
            assertEquals(expResult, result, EPS);
        }
    }
```

Індивідуальні завдання:

(Для виконання індивідуальних завдань № варінта є для 532 групи порядковим номером прізвища студента в списку групи, для 531 – вказаний у додатку (окремий документ). Усі проекти та, за наявності, тести до них завантажити у власні репозиторії на Git Hub. Посилання на репозиторії проектів вказати у звіті та обов'язково долучати скріни успішного виконання)

1. Створити програми для розв'язування задачі згідно свого варіанту та модульний тест до неї. Виконати завдання з використанням функцій. Організувати окремий метод для обчислення кожного показника. Модульний тест повинен перевіряти коректність роботи усіх створених у проекті методів. Передбачити в проекті обробку виключних ситуацій для усіх можливих у вашому варіанті випадків, коли розрахунки виконати неможливо.

Для одновимірного масиву цілих чисел, заданого випадковим чином з діапазону [-100;100]…

(2 бала)

1. Визначити 1) номер мінімального за модулем елемента масиву; 2) суму елементів масиву, розташованих між першим й останнім додатними елементами.

2. Визначити 1) суму елементів масиву з непарними номерами; 2) суму елементів масиву, розташованих між першим й другим додатними елементами.

3. Визначити 1) максимальний елемент масиву; 2) добуток елементів масиву, розташованих між першим й останнім від'ємними елементами.

4. Визначити 1) кількість елементів масиву, менших за число сім; 2) добуток елементів масиву, розташованих між першим й другим нульовими елементами.

5. Визначити 1) кількість від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального елемента.

6. Визначити 1) суму від'ємних елементів масиву; 2) добуток елементів масиву, розташованих між максимальним і мінімальним елементами.

7. Визначити 1) добуток додатних елементів масиву; 2) суму елементів масиву, розташованих до останнього додатного елемента.

8. Визначити 1) номер мінімального за модулем елемента масиву; 2) добуток елементів масиву, розташованих між першим й другим нульовими елементами.

9. Визначити 1) суму елементів масиву з непарними номерами; 2) добуток елементів масиву, розташованих між першим й останнім від'ємними елементами.

10. Визначити 1) максимальний елемент масиву; 2) суму елементів масиву, розташованих між першим й другим додатними елементами.

11. Визначити 1) кількість елементів масиву, менших за число сім; 2) суму елементів масиву, розташованих між першим й останнім додатними елементами.

12. Визначити 1) номер максимального елемента масиву; 2) суму модулів елементів масиву, розташованих між першим й останнім нульовими елементами.

13. Визначити 1) мінімальний елемент масиву; 2) суму елементів масиву, розташованих між першим і другим від'ємними елементами.

14. Визначити 1) кількість від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального за модулем елемента.

15. Визначити 1) максимальний за модулем елемент масиву; 2) суму елементів масиву, розташованих після останнього нульового елемента.

16. Визначити 1) суму модулів від'ємних елементів масиву; 2) добуток елементів масиву, розташованих до останнього від'ємного елемента.

17. Визначити 1) номер максимального за модулем елемента масиву; 2) добуток елементів масиву, розташованих після першого додатного елемента.

18. Визначити 1) добуток ненульових елементів масиву; 2) суму модулів елементів масиву, розташованих після першого від'ємного елемента.

19. Визначити 1) кількість нульових елементів масиву; 2) добуток елементів масиву, розташованих після максимального за модулем елемента.

20. Визначити 1) добуток від'ємних елементів масиву; 2) суму елементів масиву, розташованих після мінімального елемента.

21. Визначити 1) кількість додатних елементів масиву; 2) добуток елементів масиву, розташованих до мінімального за модулем елемента.

22. Визначити 1) добуток елементів масиву з парними номерами; 2) суму елементів масиву, розташованих до максимального за модулем елемента.

23. Визначити 1) номер мінімального елемента масиву; 2) добуток елементів масиву, розташованих до першого нульового елемента.

24. Визначити 1) мінімальний за модулем елемент масиву; 2) добуток елементів масиву, розташованих між першим й останнім нульовими елементами

25. Визначити 1) кількість елементів масиву, більших за число п'ять; 2) суму елементів масиву, розташованих після максимального елемента.

2. Створити програми для розв'язування задачі згідно свого варіанту та модульний тест до неї. Під підпрограмою розуміється функція, дружня функція або процедура. Модульний тест повинен перевіряти коректність роботи підпрограми.

(3 бала)

1. Написати підпрограму, яка обчислює суму елементів одновимірного масиву із n елементів цілого типу з непарними індексами.
2. Написати підпрограму, яка обчислює суму максимального та мінімального елементів одновимірного масиву із n елементів цілого типу.
3. Написати підпрограму, яка шукає індекс найбільшого парного елемента одновимірного масиву із n елементів цілого типу.
4. Написати підпрограму, яка обчислює суму індексів максимального та мінімального елементів одновимірного масиву із n елементів цілого типу.
5. Написати підпрограму, яка шукає найбільший непарний елемент одновимірного масиву із n елементів цілого типу.
6. Написати підпрограму, яка обчислює середнє арифметичне індексів парних елементів одновимірного масиву (вектора) із n елементів цілого типу.
7. Написати підпрограму, яка обчислює суму парних елементів одновимірного масиву (вектора) із n елементів цілого типу.
8. Поміняти місцями елементи одновимірного масиву із $2n$ елементів цілого типу так, щоб вони розмістилися в такому порядку: a_{n+1}, ..., a_{2n}, a_1, …, a_n (перша половина елементів вектора міняється місцями з другою).
9. Впорядкувати за спаданням елементи одновимірного масиву із n елементів цілого типу.
10. Обчислити середнє арифметичне максимального та мінімального елементів одновимірного масиву із n елементів цілого типу.
11. Написати підпрограму, яка міняє місцями максимальний та мінімальний елементи одновимірного масиву із n елементів цілого типу.
12. Написати підпрограму, яка шукає індекс найменшого непарного елемента одновимірного масиву із n елементів цілого типу.
13. Написати підпрограму, яка обчислює кількість непарних елементів одновимірного масиву із n елементів цілого типу.
14. Написати підпрограму, яка обчислює середнє арифметичне індексів максимального

та мінімального елементів одновимірного масиву із n елементів цілого типу.
15. Написати підпрограму, яка обчислює середнє арифметичне непарних елементів одновимірного масиву із n елементів цілого типу.
16. Написати підпрограму, яка міняє місцями перший елемент із найменшим парним елементом одновимірного масиву із n елементів цілого типу.
17. Написати підпрограму, яка шукає максимальний та мінімальний елементи одновимірного масиву із n елементів цілого типу.
18. Написати підпрограму, яка міняє місцями елементи одновимірного масиву із n елементів цілого типу так, щоб вони розмістилися в зворотному порядку: $a_n, a_{n-1}, \ldots, a_2, a_1$.
19. Написати підпрограму, яка обчислює суму індексів непарних елементів одновимірного масиву із n елементів цілого типу.
20. Написати підпрограму, яка обчислює середнє арифметичне елементів одновимірного масиву із n елементів цілого типу з парними індексами.
21. Написати підпрограму, яка міняє місцями останній елемент із найбільшим непарним елементом одновимірного масиву із n елементів цілого типу.
22. Написати підпрограму, яка шукає індекси найбільшого та найменшого елементів одновимірного масиву із n елементів цілого типу.
23. Написати підпрограму, яка обчислює середнє арифметичне максимального та мінімального елементів одновимірного масиву із n елементів цілого типу.
24. Написати підпрограму, яка шукає найменший парний елемент одновимірного масиву із n елементів цілого типу.
25. Написати підпрограму, яка міняє місцями елементи одновимірного масиву із $2n$ елементів цілого типу так, щоб вони розмістилися в такому порядку:$a_2, a_1, a_4, a_3, \ldots, a_{2n}, a_{2n-1}$. (кожний елемент з парним індексом міняється місцями з попереднім елементом).