**Лабораторная работа №6**

**Тема: Двумерные массивы (матрицы). Формирование матрицы и вывод ее элементов**

Выполнил студент: **Гилязов И.И**

Группа: **237**

**Вариант: 5**

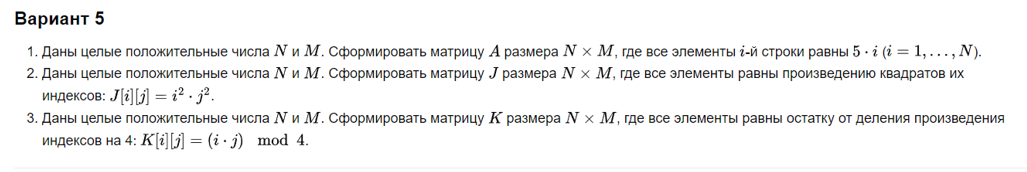
**Цель работы:**

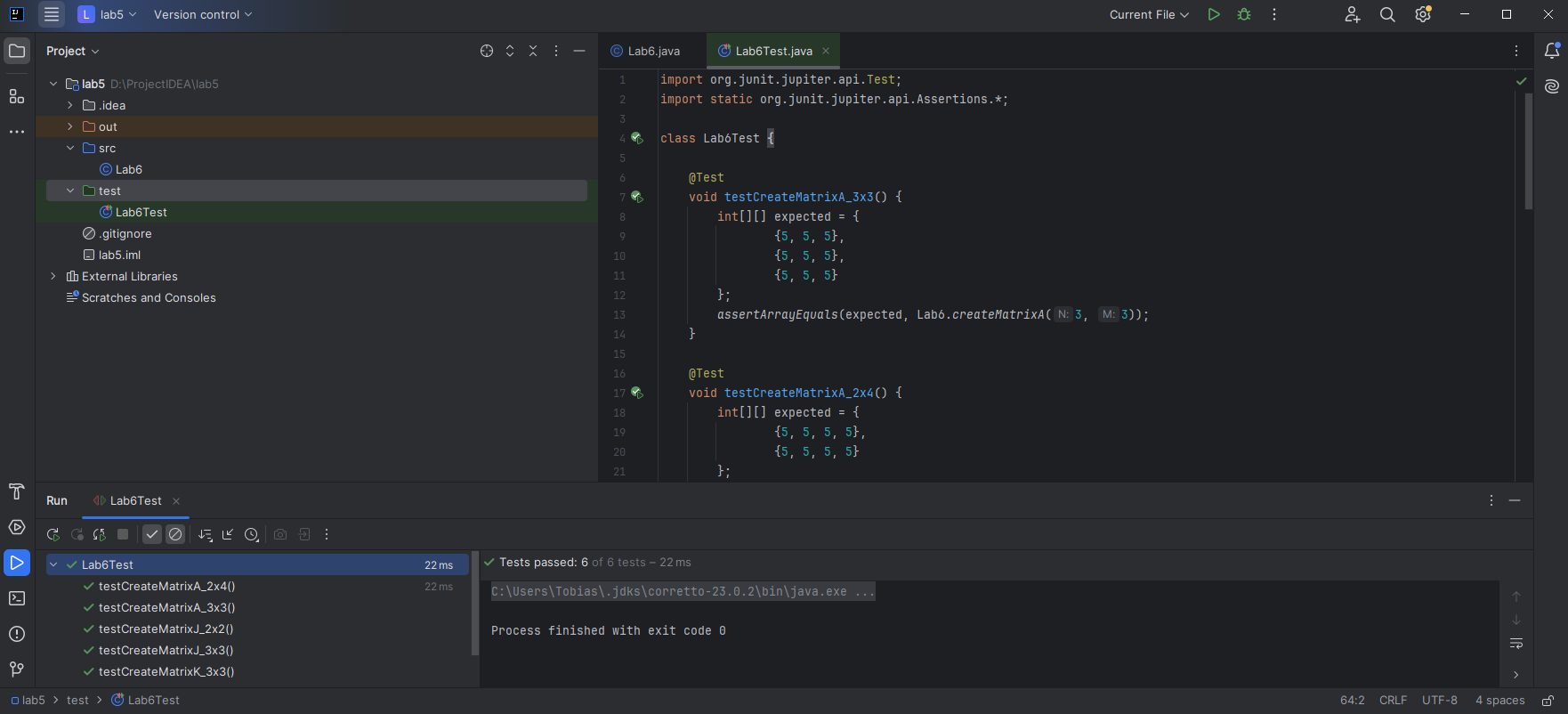
**\* Закрепить принципы тестирования массивов.**

**\* Разработать тест-кейсы для проверки корректности работы методов.**

**\* Провести тестирование на основе позитивных и негативных сценариев.**

**Задание**

****

**Пример класса для тестирования**

public class Lab6 {  
  
 // Метод для создания матрицы A, где каждый элемент равен 5  
 public static int[][] createMatrixA(int N, int M) {  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < M; j++) {  
 matrix[i][j] = 5;  
 }  
 }  
 return matrix;  
 }  
  
 // Метод для создания матрицы J, где каждый элемент равен i² - j²  
 public static int[][] createMatrixJ(int N, int M) {  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < M; j++) {  
 matrix[i][j] = (i \* i) - (j \* j);  
 }  
 }  
 return matrix;  
 }  
  
 // Метод для создания матрицы K, где каждый элемент равен (i \* j) mod 4  
 public static int[][] createMatrixK(int N, int M) {  
 int[][] matrix = new int[N][M];  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < M; j++) {  
 matrix[i][j] = (i \* j) % 4;  
 }  
 }  
 return matrix;  
 }  
}

**Пример тестового класса**

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
class Lab6Test {  
  
 @Test  
 void testCreateMatrixA\_3x3() {  
 int[][] expected = {  
 {5, 5, 5},  
 {5, 5, 5},  
 {5, 5, 5}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixA*(3, 3));  
 }  
  
 @Test  
 void testCreateMatrixA\_2x4() {  
 int[][] expected = {  
 {5, 5, 5, 5},  
 {5, 5, 5, 5}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixA*(2, 4));  
 }  
  
 @Test  
 void testCreateMatrixJ\_3x3() {  
 int[][] expected = {  
 {0, -1, -4},  
 {1, 0, -3},  
 {4, 3, 0}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixJ*(3, 3));  
 }  
  
 @Test  
 void testCreateMatrixJ\_2x2() {  
 int[][] expected = {  
 {0, -1},  
 {1, 0}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixJ*(2, 2));  
 }  
  
 @Test  
 void testCreateMatrixK\_3x3() {  
 int[][] expected = {  
 {0, 0, 0},  
 {0, 1, 2},  
 {0, 2, 0}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixK*(3, 3));  
 }  
  
 @Test  
 void testCreateMatrixK\_4x2() {  
 int[][] expected = {  
 {0, 0},  
 {0, 1},  
 {0, 2},  
 {0, 3}  
 };  
 *assertArrayEquals*(expected, Lab6.*createMatrixK*(4, 2));  
 }  
}

**тест-кейс**

|  |  |
| --- | --- |
| **Поле** | **Значение** |
| **ID тест-кейса** | TC001 |
| **Название тест-кейса** | Проверка функции createMatrixA |
| **Описание** | Проверить, что функция createMatrixA(int N, int M) корректно создает матрицу размера 3×3, где каждый элемент равен 5. |
| **Предусловия** | Нет |
| **Шаги выполнения** | 1. Объявить ожидаемую матрицу: int[][] expected = {{5, 5, 5}, {5, 5, 5}, {5, 5, 5}};  2. Вызвать метод createMatrixA(3, 3), который возвращает созданную матрицу: int[][] actual = Lab6.createMatrixA(3, 3);  3. Сравнить ожидаемую и фактическую матрицы с помощью Assertions.assertArrayEquals(expected, actual);. |
| **Ожидаемый результат** | Метод createMatrixA(3, 3) возвращает матрицу:  {{5, 5, 5},  {5, 5, 5},  {5, 5, 5}}. |
| **Фактический результат** | Метод возвращает матрицу, совпадающую с ожидаемой. |
| **Статус** | Pass |
| **Примечания** | Нет |

|  |  |
| --- | --- |
| **Поле** | **Значение** |
| **ID тест-кейса** | TC002 |
| **Название тест-кейса** | Проверка функции createMatrixJ |
| **Описание** | Проверить, что функция createMatrixJ(int N, int M) корректно создает матрицу размера 3×3, где каждый элемент равен i² - j². |
| **Предусловия** | Нет |
| **Шаги выполнения** | 1. Объявить ожидаемую матрицу: int[][] expected = {{0, -1, -4}, {1, 0, -3}, {4, 3, 0}};  2. Вызвать метод createMatrixJ(3, 3), который возвращает созданную матрицу: int[][] actual = Lab6.createMatrixJ(3, 3);  3. Сравнить ожидаемую и фактическую матрицы с помощью Assertions.assertArrayEquals(expected, actual);. |
| **Ожидаемый результат** | Метод createMatrixJ(3, 3) возвращает матрицу:  {{0, -1, -4},  {1, 0, -3},  {4, 3, 0}}. |
| **Фактический результат** | Метод возвращает матрицу, совпадающую с ожидаемой. |
| **Статус** | Pass |
| **Примечания** | Нет |

|  |  |
| --- | --- |
| **Поле** | **Значение** |
| **ID тест-кейса** | TC003 |
| **Название тест-кейса** | Проверка функции createMatrixK |
| **Описание** | Проверить, что функция createMatrixK(int N, int M) корректно создает матрицу размера 3×3, где каждый элемент равен (i \* j) mod 4. |
| **Предусловия** | Нет |
| **Шаги выполнения** | 1. Объявить ожидаемую матрицу: int[][] expected = {{0, 0, 0}, {0, 1, 2}, {0, 2, 0}};  2. Вызвать метод createMatrixK(3, 3), который возвращает созданную матрицу: int[][] actual = Lab6.createMatrixK(3, 3);  3. Сравнить ожидаемую и фактическую матрицы с помощью Assertions.assertArrayEquals(expected, actual);. |
| **Ожидаемый результат** | Метод createMatrixK(3, 3) возвращает матрицу:  {{0, 0, 0},  {0, 1, 2},  {0, 2, 0}}. |
| **Фактический результат** | Метод возвращает матрицу, совпадающую с ожидаемой. |
| **Статус** | Pass |
| **Примечания** | Нет |

|  |  |
| --- | --- |
| **Поле** | **Значение** |
| **ID тест-кейса** | TC004 |
| **Название тест-кейса** | Проверка функции createMatrixA для 2×4 |
| **Описание** | Проверить, что функция createMatrixA(int N, int M) корректно создает матрицу размера 2×4, где каждый элемент равен 5. |
| **Предусловия** | Нет |
| **Шаги выполнения** | 1. Объявить ожидаемую матрицу: int[][] expected = {{5, 5, 5, 5}, {5, 5, 5, 5}};  2. Вызвать метод createMatrixA(2, 4), который возвращает созданную матрицу: int[][] actual = Lab6.createMatrixA(2, 4);  3. Сравнить ожидаемую и фактическую матрицы с помощью Assertions.assertArrayEquals(expected, actual);. |
| **Ожидаемый результат** | Метод createMatrixA(2, 4) возвращает матрицу:  {{5, 5, 5, 5},  {5, 5, 5, 5}}. |
| **Фактический результат** | Метод возвращает матрицу, совпадающую с ожидаемой. |
| **Статус** | Pass |
| **Примечания** | Нет |

|  |  |
| --- | --- |
| **Поле** | **Значение** |
| **ID тест-кейса** | TC005 |
| **Название тест-кейса** | Проверка функции createMatrixK для 4×2 |
| **Описание** | Проверить, что функция createMatrixK(int N, int M) корректно создает матрицу размера 4×2, где каждый элемент равен (i \* j) mod 4. |
| **Предусловия** | Нет |
| **Шаги выполнения** | 1. Объявить ожидаемую матрицу: int[][] expected = {{0, 0}, {0, 1}, {0, 2}, {0, 3}};  2. Вызвать метод createMatrixK(4, 2), который возвращает созданную матрицу: int[][] actual = Lab6.createMatrixK(4, 2);  3. Сравнить ожидаемую и фактическую матрицы с помощью Assertions.assertArrayEquals(expected, actual);. |
| **Ожидаемый результат** | Метод createMatrixK(4, 2) возвращает матрицу:  {{0, 0}, {0, 1},  {0, 2}, {0, 3}}. |
| **Фактический результат** | Метод возвращает матрицу, совпадающую с ожидаемой. |
| **Статус** | Pass |
| **Примечания** | Нет |

### Контрольные вопросы

1. **Как объявить двумерный массив в Java? Приведите пример объявления и инициализации матрицы размера 3×4, заполненной нулями.**

**Ответ**: В Java двумерный массив объявляется как массив массивов. Для инициализации можно использовать конструктор с указанием размеров и заполнить элементы нулями с помощью цикла. Пример:

int[][] matrix = new int[3][4]; // Объявляем матрицу 3x4

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++) {

matrix[i][j] = 0; // Заполняем все элементы нулями

}

}

1. **Как вывести на экран элементы матрицы размера N×M в виде таблицы? Напишите фрагмент кода, который выполняет эту задачу.**

**Ответ**: Для вывода матрицы в виде таблицы можно использовать вложенные циклы и метод System.out.print() с разделителями (например, табуляцией \t). Пример:

public static void printMatrix(int[][] matrix, int N, int M) {

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

System.out.print(matrix[i][j] + "\t"); // Выводим элемент с табуляцией

}

System.out.println(); // Переход на новую строку после каждой строки матрицы

}

}

1. **Что произойдет, если попытаться обратиться к элементу матрицы с индексом, выходящим за пределы массива? Как избежать этой ошибки?**

**Ответ**: Если обратиться к элементу матрицы с индексом за пределами массива (например, matrix[3][4] для матрицы 3×4, где допустимые индексы i от 0 до 2, j от 0 до 3), возникнет исключение ArrayIndexOutOfBoundsException. Чтобы избежать этой ошибки, нужно проверять индексы перед доступом:

if (i >= 0 && i < N && j >= 0 && j < M) {

// Безопасный доступ к matrix[i][j]

} else {

System.out.println("Индекс вне диапазона!");

}

1. **Как создать матрицу размера N×M, где каждый элемент равен произведению его индексов? Напишите соответствующий код на Java.**

**Ответ**: Можно использовать вложенные циклы для заполнения матрицы, где каждый элемент равен произведению его индексов i и j. Пример:

public static int[][] createProductMatrix(int N, int M) {

int[][] matrix = new int[N][M];

for (int i = 0; i < N; i++) {

for (int j = 0; j < M; j++) {

matrix[i][j] = i \* j; // Каждый элемент равен произведению индексов

}

}

return matrix;

}

1. **Как проверить, является ли матрица квадратной (т.е. количество строк равно количеству столбцов)? Напишите фрагмент кода, который выполняет эту проверку.**

**Ответ**: Для проверки, является ли матрица квадратной, нужно сравнить количество строк (длина массива) с количеством столбцов (длина первой строки, если она существует). Пример:

public static boolean isSquareMatrix(int[][] matrix) {

int rows = matrix.length; // Количество строк

int cols = (rows > 0) ? matrix[0].length : 0; // Количество столбцов

return rows == cols; // Возвращаем true, если квадратная

}