Лабораторная работа No4

**Исследование способов структурного тестирования программного обеспечения**

**Цель работы**

Исследовать основные принципы интеграционного тестирования программного обеспечения. Приобрести практические навыки организации интеграционных тестов для объектно-ориентированных программ.

**Вариант - 8**

Варианты заданий соответствуют заданиям по лабораторной работе No1. По варианту задаются требования к программам. Для каждой из них необходимо:

1) **Названия взаимодействующих классов**: SquareMatrix, SquareMatrixArray.  
2) **Название теста**: squareArrayTest1.  
3) **Описание теста**: тест проверяет возможность создания объекта типа SquareMatrix и добавления его в массив при вызове метода AddMatrixs().

4) **Начальные условия**: массив пустой  
5) **Ожидаемый результат**: в массив будет добавлена матрица.

**Ход работы:**

**Код программы:**

const fs = require('fs');

class SquareMatrixArray {

constructor(){

this.squareMatrixArray = []

}

addMatrixs() {

let matrix = SquareMatrix([[0,2],[0,1]])

this.squareMatrixArray.push(matrix)

}

rotate(matrix, k) {

if(k <0 ) return 'k is wrong'

let isSquare = SquareMatrix.isSquareMatrix(matrix)

if (isSquare == 'neverno') return 'matrix is no square'

var result = [...matrix];

let size = matrix[0].length

var delta = matrix

for (let d = 0; d < k; d++) {

for (let i = 0; i < size; i++) {

for (let j = 0; j < size; j++) {

result[i][j] = delta[size - j - 1][i]

}

}

delta = result

}

return result

}

}

class SquareMatrix {

constructor(){

this.matrix = [[]]

}

init(matrix) {

self.matrix = isSquareMatrix(matrix ? matrix : [])

}

static isSquareMatrix(matrix) {

for(let item in matrix){

if(matrix[item].length != matrix[Number(item + 1)].length){

return 'neverno'

}

}

}

}

class Tester {

constructor(){

this.squareMatrixArray;

this.init()

}

init() {

console.log("OK")

this.run()

}

Init() {

this.squareMatrixArray = new SquareMatrixArray()

}

dump() {

if(!squareMatrixArray.length) {

Logger.log("Пустой массив")

return

}

Logger.log("///////////////Массив матриц/////////////")

squareMatrixArray.squareMatrixArray.forEach(item =>{

Logger.log(`Element: ${element.matrix}`)

})

Logger.log("/////////////////////////////////////////")

}

squareArrayTest1() {

this.Init()

Logger.log("//////////////////squareArrayTest1/////////////////")

Logger.log("Проверяем, создаётся ли объект типа SquareMatrix")

this.dump()

this.squareMatrixArray.addMatrixs()

Logger.log("Команда добавлена")

this.dump()

}

run() {

this.squareArrayTest1()

console.log("Run")

}

}

class Logger {

static log(\_message) {

if (fs.existsSync('myloger.txt')) {

let date = new Date('dd-MM-yyyy');

fs.writeFileSync('myloger.txt', `${date}:${\_message} `);

}

else{

console.log('file not found')

}

}

}

var tester = new Tester()

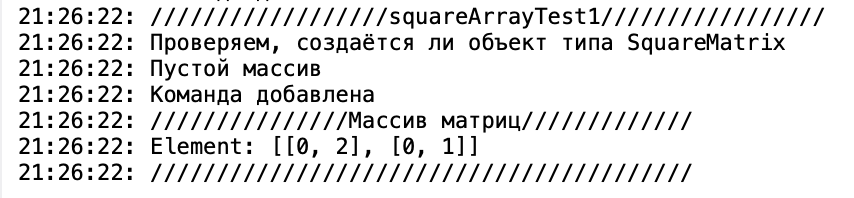


Рисунок 1 – Содержимое файла Log.txt

**Контрольные вопросы:**

1.Для чего применяется интеграционное тестирование программного обеспечения?

Целью интеграционного тестирования является только проверка правильности взаимодействия объектов, а не проверка правильности функционирования системы в целом.

2. Какие существуют типы взаимодействий объектов?

1) Общедоступная операция имеет один или большее число формальных параметров объектного типа. Сообщение устанавливает ассоциацию между получателем и параметром, которая позволяет получателю взаимодействовать с этим параметрическим объектом.

2)Общедоступная операция возвращает значения объектного типа. На класс может быть возложена задача создания возвращаемого объекта, либо он может возвращать модифицированный параметр.

3) Метод одного класса создает экземпляр другого класса как часть своей реализации.

4) Метод одного класса ссылается на глобальный экземпляр некоторого другого класса.

5.3. Исходя из каких соображений выполняется выбор тестовых случаев при интеграционном тестировании?

Если есть выбор, то мы отдаем предпочтение таким тестовым случаям, которые позволяют найти ошибки, в обнаружении которых мы заинтересованы больше всего.

Существуют различные способы определения, какое подмножество из множества всех возможных тестовых случаев следует выбирать. При любом подходе мы заинтересованы в том, чтобы систематически повышать уровень покрытия.

5.4.Какие разделы включает спецификация тестового случая для интеграционного тестирования?

1) **Названия взаимодействующих классов**: TСommandQueue, TCommand.

2) **Название теста**: TCommandQueueTest1.

3) **Описание теста**: тест проверяет возможность создания объекта типа

TCommand и добавления его в очередь при вызове метода AddCommand(). 4) **Начальные условия**: очередь команд пуста.

5) **Ожидаемый результат**: в очередь будет добавлена одна команда.

**Вывод:**

Исследованы основные принципы интеграционного тестирования программного обеспечения. Приобретены практические навыки организации интеграционных тестов для объектно-ориентированных программ.