Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №5**

По дисциплине «СПП»

за 6-й семестр

Выполнил:

студент 3 курса

группы ПО-4 (2)

Яковчик И.А.

Проверил:

Монтик Н.С.

Брест, 2022

**Цель работы:** освоить приемы тестирования кода на примере использования библиотеки xUnit.

**Вариант:** 24

**Задание1:**

Создаете новый класс и скопируйте код класса Sum;

• Создаете тестовый класс SumTest;

• Напишите тест к методу Sum.accum и проверьте его исполнение. Тест должен проверять работоспособность функции accum.

• Очевидно, что если передать слишком большие значения в Sum.accum, то случится переполнение. Модифицируйте функцию Sum.accum, чтобы она возвращала значение типа long и напишите новый тест, проверяющий корректность работы функции с переполнением. Первый тест должен работать корректно.

**Задание2:**

Напишите метод String keep(String str, String pattern) который оставляет в первой строке все символы, которые присутствуют во второй. Спецификация метода: keep (null , null ) = NullPointerException keep (null , \*) = null keep ("", \*) = "" keep (\* , null ) = "" keep (\* , "") = "" keep (" hello ", "hl") = " hll " keep (" hello ", "le") = " ell "

**Задание3:**

1) Импорт проекта Импортируйте один из проектов по варианту (2):

• Queue – содержит реализацию очереди на основе связного списка: Queue.java.

Разберитесь как реализована ваша структура данных. Каждый проект содержит:

• Клиент для работы со структурой данных и правильности ввода данных реализации (см.

метод main()).

• TODO-декларации, указывающие на нереализованные методы и функциональность.

• FIXME-декларации, указывающую на необходимые исправления.

• Ошибки компиляции (Синтаксические)

• Баги в коде (!).

• Метод check() для проверки целостности работы класса.

2) Поиск ошибок

• Исправить синтаксические ошибки в коде.

• Разобраться в том, как работает код, подумать о том, как он должен работать и найти допущенные баги.

3) Внутренняя корректность

• Разобраться что такое утверждения (assertions) в коде и как они включаются в Java.

• Заставить ваш класс работать вместе с включенным методом check.

• Выполнить клиент (метод main() класса) передавая данные в структуру используя включенные проверки (assertions).

4) Реализация функциональности

• Реализовать пропущенные функции в классе.

• См. документацию перед методом относительно того, что он должен делать и какие исключения выбрасывать.

• Добавить и реализовать функцию очистки состояния структуры данных.

5) Написание тестов

• Все функции вашего класса должны быть покрыты тестами.

• Использовать фикстуры для инициализации начального состояния объекта.

• Итого, должно быть несколько тестовых классов, в каждом из которых целевая структура данных создается в фикстуре в некотором инициализированном состоянии (пустая, заполненная и тд), а после очищается.

• Написать тестовый набор, запускающий все тесты.

**Код программы:**

1. **Program.cs**
2. using System;
3. using System.Linq;
4. using System.Text;
5. namespace ForIllya
6. {
7. public class Sum
8. {
9. public static long Accum(params int[] values) => values.Select(Convert.ToInt64).Sum();
10. }
11. internal class Program
12. {
13. static void Main(String[] arts)
14. { }
15. }
16. public class StringUtils
17. {
18. public static string Keep(string str, string keep) => (str, keep) switch
19. {
20. (null, null) => throw new NullReferenceException(),
21. (null, not null) => null,
22. ("", not null) => "",
23. (not null, null) => "",
24. (not null, "") => "",
25. \_ => string.Join("", str.Where(keep.Contains))
26. };
27. }
28. public class Stack<T>
29. {
30. private class Node
31. {
32. public T \_item;
33. public Node \_next;
34. }
35. private Node \_first;
36. private int \_N;
37. public Stack()
38. {
39. \_first = null;
40. \_N = 0;
41. }
42. public bool IsEmpty() => \_N < 1;
43. public int Size() => \_N;
44. public void Push(T item)
45. {
46. Node oldfirst = \_first;
47. \_first = new Node();
48. \_first.\_item = item;
49. \_first.\_next = oldfirst;
50. \_N++;
51. }
52. public T Pop()
53. {
54. if (Size() == 0)
55. throw new ArgumentNullException();
56. T item = \_first.\_item;
57. \_first = \_first.\_next;
58. \_N--;
59. return item;
60. }
61. public T Peek()
62. {
63. if (Size() == 0)
64. throw new ArgumentNullException();
65. else
66. return \_first.\_item;
67. }
68. public override string ToString()
69. {
70. StringBuilder s = new StringBuilder();
71. for (Node current = \_first; current != current.\_next; current = current.\_next)
72. {
73. T item = current.\_item;
74. s.Append(item);
75. if (current.\_next == null)
76. return s.ToString();
77. s.Append(" - ");
78. }
79. return s.ToString();
80. }
81. }
82. }

**2)UnitTest1.cs:**

using System;

using Xunit;

using ForIllya;

namespace UnitTest1

{

public class UnitTest1

{

//[Fact]

//public void TestSmallInt()

//{

// //arrange

// int a = 1;

// int b = 2;

// int c = 3;

// int d = 4;

// int expected = 10;

// //act

// int actual = Sum.Accum(a, b, c, d);

// //assert

// Assert.Equal(expected, actual);

//}

//[Fact]

//public void TestBigInt1()

//{

// int a = 2\_000\_000\_000;

// int b = 1\_000\_000\_000;

// uint expected = 3\_000\_000\_000;

// int actual = Sum.Accum(a, b);

// Assert.Equal(expected, (uint)actual);

//}

[Fact]

public void TestBigInt2()

{

int a = 2\_000\_000\_000;

int b = 1\_000\_000\_000;

long expected = 3\_000\_000\_000;

long actual = Sum.Accum(a, b);

Assert.Equal(expected, actual);

}

}

public class UnitTest2

{

[Fact]

public void TestKeepNullAndNull()

{

string a = null;

string b = null;

Assert.Throws<NullReferenceException>(() => StringUtils.Keep(a, b));

}

[Fact]

public void TestKeep\_NullAndAsterisk()

{

string a = null;

string b = "test";

string expected = a;

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

[Fact]

public void TestKeep\_EmptyStringAndAsterisk()

{

string a = string.Empty;

string b = "test";

string expected = a;

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

[Fact]

public void TestKeep\_AsteriskAndNull()

{

string a = "test";

string b = null;

string expected = string.Empty;

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

[Fact]

public void TestKeep\_AsteriskAndEmptyString()

{

string a = "test";

string b = string.Empty;

string expected = string.Empty;

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

[Fact]

public void TestKeep\_helloAndhl()

{

string a = "hello";

string b = "hl";

string expected = "hll";

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

[Fact]

public void TestKeep\_helloAndle()

{

string a = "hello";

string b = "le";

string expected = "ell";

string result = StringUtils.Keep(a, b);

Assert.Equal(expected, result);

}

}

public class UnitTest3

{

private Stack<string> \_stack = new Stack<string>();

[Fact]

public void TestStack\_PopEmptyStack()

{

Assert.Throws<ArgumentNullException>(() => \_stack.Pop());

}

[Fact]

public void TestStack\_PeekEmptyStack()

{

Assert.Throws<ArgumentNullException>(() => \_stack.Peek());

}

[Fact]

public void TestStack\_IsEmpty()

{

bool expected = true;

bool actual = \_stack.IsEmpty();

Assert.Equal(expected, actual);

}

[Fact]

public void TestStack\_Size()

{

int expected = 0;

int actual = \_stack.Size();

Assert.Equal(expected, actual);

}

[Fact]

public void TestStack\_Push555()

{

string expected = "555";

\_stack.Push("555");

string actual = \_stack.Pop();

Assert.Equal(expected, actual);

}

[Fact]

public void TestStack\_Peek()

{

string expected = "555";

\_stack.Push("555");

string actual = \_stack.Peek();

Assert.Equal(expected, actual);

}

[Fact]

public void TestStack\_ToString()

{

string expected = "333 - 444 - 555";

\_stack.Push("555");

\_stack.Push("444");

\_stack.Push("333");

string actual = \_stack.ToString();

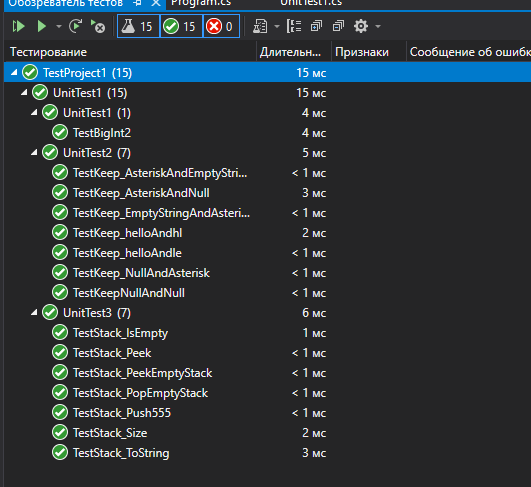
Assert.Equal(expected, actual);

}

}

}

**Результаты работы:**

****

**Выводы:** в ходе выполнения лабораторной работы были освоены приемы тестирования кода на примере использования библиотеки JUnit.