Министерство науки и высшего образования РФ

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

**ОБЪЕКНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ДИЗАЙН**

Отчёт по лабораторной работе №3

«Разделение интерфейса и реализации при использовании объектно-ориентированного подхода»

Выполнил:

Студент: 2 курса

группы 20-КБ-ПР2

Филимонов Р.С.

Проверил:

ассистент каф. ИСП

Степанова Е.В.

г. Краснодар 2022

**Цель:** Освоить принцип разделения интерфейса и реализации при использовании объектно-ориентированного подхода к разработке программного обеспечения.

**Задания:**

1) Создать класс, реализующий один из алгоритмов (в соответствии с вариантом задания из п.5).

2) Создать проект для автоматизации модульного тестирования с необходимым набором тестов.

3) Запустить проект тестирования и проверить результаты работы.

4) Составить отчет о результатах проведенного тестирования.

Вариант задания: 8

**Код программы:**

**Интерфейс:**

interface IQueue<T>

{

T[] queue { get { return queue; } set { queue = value; } } // массив с очередью

public string ReturnQueue(); // метод для показа текущей очереди

public void AddItem(T item); // метод добавления нового элемента в очередь

public T RemoveItem(); // метод удаления элемента из очереди

}

**Класс:**

public class Queue<T> : IQueue<T>

{

T[] queue;

int count = -1; // счётчик для отслеживания заполненности списка(массива)

public Queue(T[] Queue)

{

this.queue = Queue;

this.count = Queue.Length - 1;

}

public Queue()

{

this.queue = new T[1];

}

public string ReturnQueue()

{

if (count >= 0)

{

string ret = "\nОчередь: ";

for (int i = 0; i < count + 1; i++)

ret += $"{queue[i]} ";

return ret;

}

else

return "Список пуст";

}

public void AddItem(T item)

{

try

{

if (count < (queue.Length - 1))

{

for (int i = count; i > 0; i--)

queue[i + 1] = queue[i];

queue[0] = item;

count++;

}

else

{

T[] expand = new T[queue.Length + 1];

for (int i = 0; i < queue.Length; i++)

expand[i + 1] = queue[i];

expand[0] = item;

queue = expand;

count++;

}

}

catch { Console.WriteLine("Не удалось провести операцию по добавлению элемента в очередь"); }

}

public T RemoveItem()

{

try

{

T RemovItem = queue[queue.Length - 1];

T[] remove = new T[queue.Length - 1];

for (int i = 0; i < count; i++)

remove[i] = queue[i];

queue = remove;

count--;

return RemovItem;

}

catch { Console.WriteLine("Не удалось провести операцию по удалению элемента из очереди"); return default(T); }

}

}

**Текст модульных тестов:**

public class Tests

{

[Test]

public void TestReturnQueue()

{

int[] a = { 4, 3, 2, 1, 0 };

Queue<int> queue = new Queue<int>(a);

string expected = "\nОчередь: 4 3 2 1 0 ";

string actual = queue.ReturnQueue();

Assert.AreEqual(expected, actual);

}

[Test]

public void TestAddItem()

{

int[] a = { 4, 3, 2, 1, 0 };

Queue<int> queue = new Queue<int>(a);

queue.AddItem(5);

string expected = "\nОчередь: 5 4 3 2 1 0 ";

string actual = queue.ReturnQueue();

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRemoveItem()

{

int[] a = { 5, 4, 3, 2, 1, 0 };

Queue<int> queue = new Queue<int>(a);

queue.RemoveItem();

string expected = "\nОчередь: 5 4 3 2 1 ";

string actual = queue.ReturnQueue();

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRemoveItemError()

{

int[] a = { };

Queue<int> queue = new Queue<int>(a);

int expected = queue.RemoveItem();

int actual = default(int);

Assert.AreEqual(expected, actual);

}

[Test]

public void TestAddAndRemove()

{

int[] a = { 4, 3, 2, 1, 0 };

Queue<int> queue = new Queue<int>(a);

queue.AddItem(5);

queue.RemoveItem();

string expected = "\nОчередь: 5 4 3 2 1 ";

string actual = queue.ReturnQueue();

Assert.AreEqual(expected, actual);

}

}

**Результаты тестов работы:**

