День 5

DLL, Тестирование

Реализация dll-библиотеки

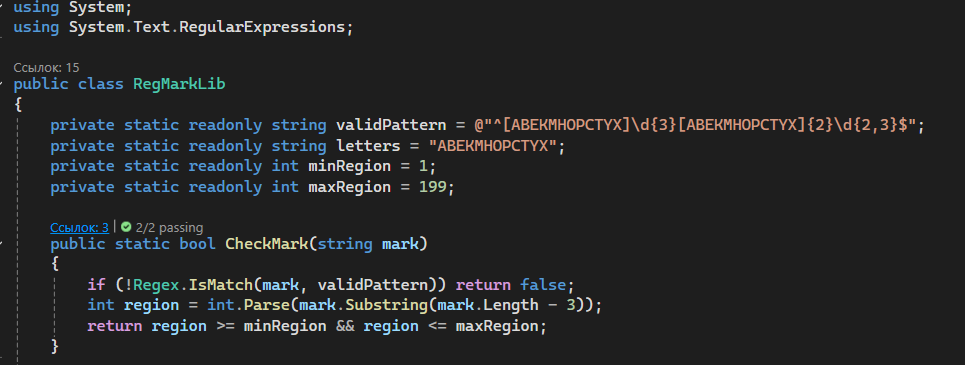


Рис – объявление переменных и метод проверки корректности

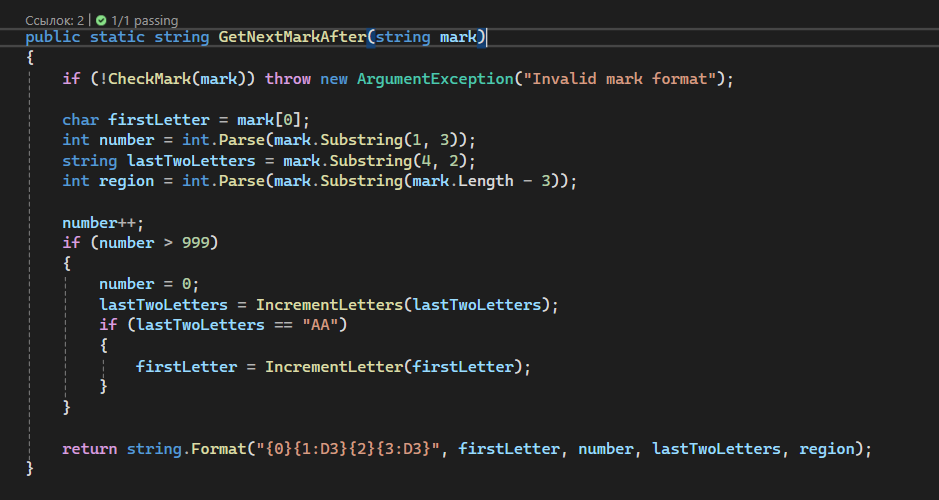


Рис – метод создания следующего номера

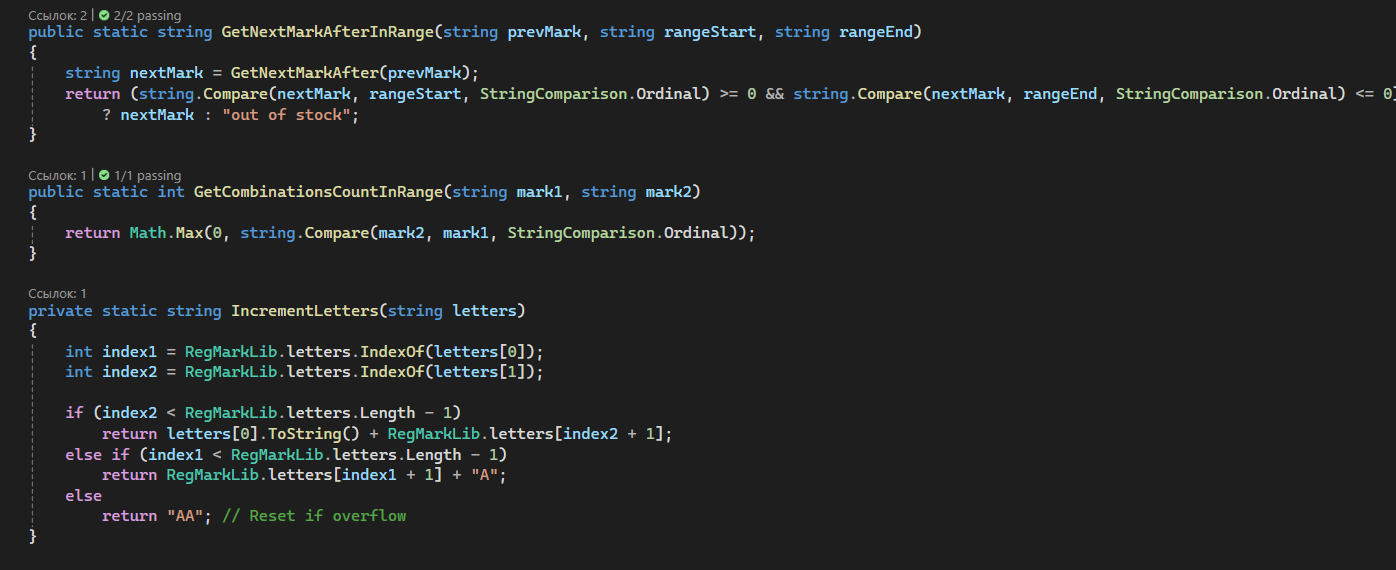


Рис. – методы «Создание нового номера в диапазоне» и «Получение количества номеров в диапазоне»

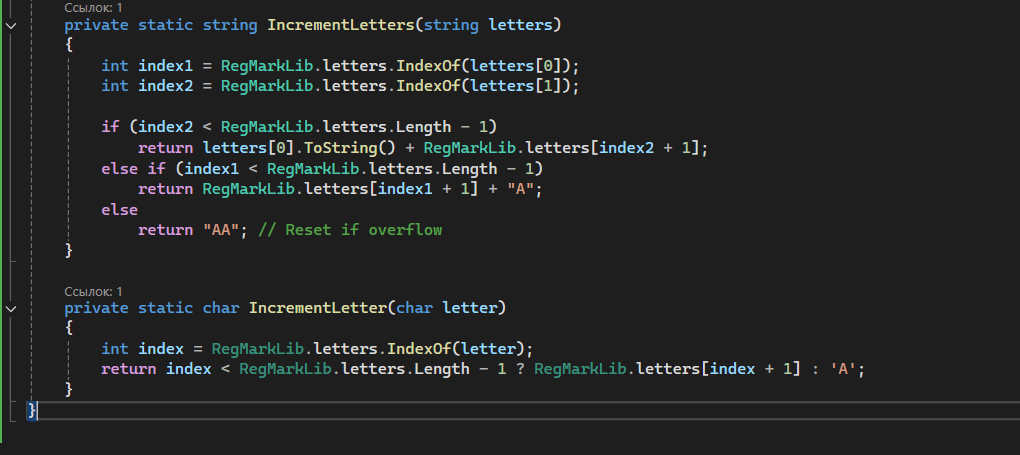
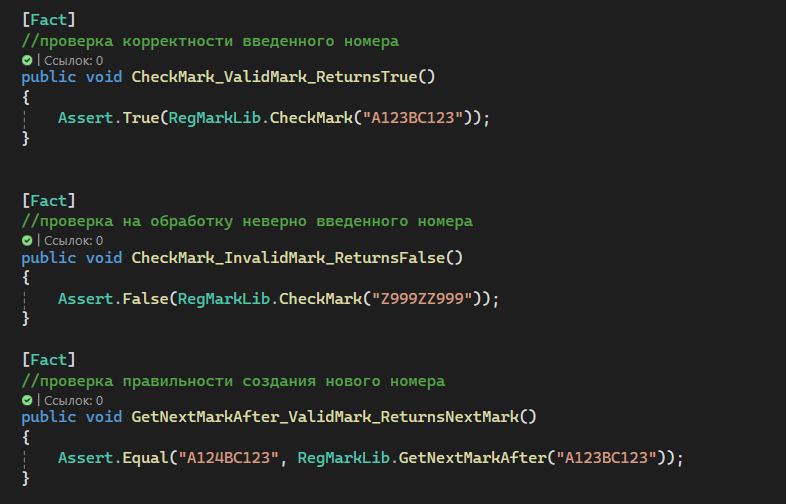


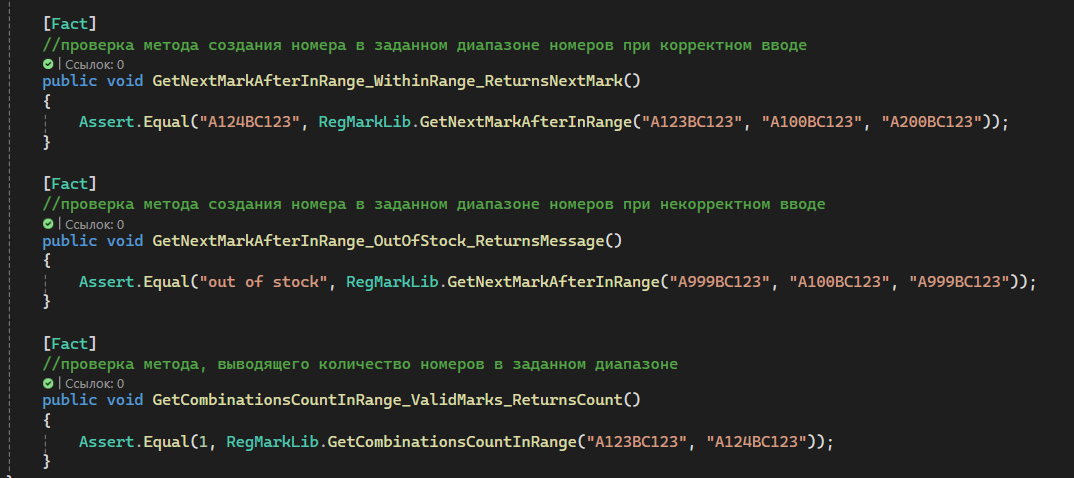
Рис – вспомогательные методы для увеличения буквенной последовательности

Таблица – реализация dll-библиотеки

|  |
| --- |
| using System;  using System.Text.RegularExpressions;  public class RegMarkLib  {  private static readonly string validPattern = @"^[ABEKMHOPCTYX]\d{3}[ABEKMHOPCTYX]{2}\d{2,3}$";  private static readonly string letters = "ABEKMHOPCTYX";  private static readonly int minRegion = 1;  private static readonly int maxRegion = 199;  public static bool CheckMark(string mark)  {  if (!Regex.IsMatch(mark, validPattern)) return false;  int region = int.Parse(mark.Substring(mark.Length - 3));  return region >= minRegion && region <= maxRegion;  }  public static string GetNextMarkAfter(string mark)  {  if (!CheckMark(mark)) throw new ArgumentException("Invalid mark format");  char firstLetter = mark[0];  int number = int.Parse(mark.Substring(1, 3));  string lastTwoLetters = mark.Substring(4, 2);  int region = int.Parse(mark.Substring(mark.Length - 3));  number++;  if (number > 999)  {  number = 0;  lastTwoLetters = IncrementLetters(lastTwoLetters);  if (lastTwoLetters == "AA")  {  firstLetter = IncrementLetter(firstLetter);  }  }  return string.Format("{0}{1:D3}{2}{3:D3}", firstLetter, number, lastTwoLetters, region);  }  public static string GetNextMarkAfterInRange(string prevMark, string rangeStart, string rangeEnd)  {  string nextMark = GetNextMarkAfter(prevMark);  return (string.Compare(nextMark, rangeStart, StringComparison.Ordinal) >= 0 && string.Compare(nextMark, rangeEnd, StringComparison.Ordinal) <= 0)  ? nextMark : "out of stock";  }  public static int GetCombinationsCountInRange(string mark1, string mark2)  {  return Math.Max(0, string.Compare(mark2, mark1, StringComparison.Ordinal));  }  private static string IncrementLetters(string letters)  {  int index1 = RegMarkLib.letters.IndexOf(letters[0]);  int index2 = RegMarkLib.letters.IndexOf(letters[1]);  if (index2 < RegMarkLib.letters.Length - 1)  return letters[0].ToString() + RegMarkLib.letters[index2 + 1];  else if (index1 < RegMarkLib.letters.Length - 1)  return RegMarkLib.letters[index1 + 1] + "A";  else  return "AA"; // Reset if overflow  }  private static char IncrementLetter(char letter)  {  int index = RegMarkLib.letters.IndexOf(letter);  return index < RegMarkLib.letters.Length - 1 ? RegMarkLib.letters[index + 1] : 'A';  }  } |

Юнит-тесты





|  |
| --- |
| namespace FactRegMark  {  public class RegMarkLibFacts  {  [Fact]  //проверка корректности введенного номера  public void CheckMark\_ValidMark\_ReturnsTrue()  {  Assert.True(RegMarkLib.CheckMark("A123BC123"));  }  [Fact]  //проверка на обработку неверно введенного номера  public void CheckMark\_InvalidMark\_ReturnsFalse()  {  Assert.False(RegMarkLib.CheckMark("Z999ZZ999"));  }  [Fact]  //проверка правильности создания нового номера  public void GetNextMarkAfter\_ValidMark\_ReturnsNextMark()  {  Assert.Equal("A124BC123", RegMarkLib.GetNextMarkAfter("A123BC123"));  }  [Fact]  //проверка метода создания номера в заданном диапазоне номеров при корректном вводе  public void GetNextMarkAfterInRange\_WithinRange\_ReturnsNextMark()  {  Assert.Equal("A124BC123", RegMarkLib.GetNextMarkAfterInRange("A123BC123", "A100BC123", "A200BC123"));  }  [Fact]  //проверка метода создания номера в заданном диапазоне номеров при некорректном вводе  public void GetNextMarkAfterInRange\_OutOfStock\_ReturnsMessage()  {  Assert.Equal("out of stock", RegMarkLib.GetNextMarkAfterInRange("A999BC123", "A100BC123", "A999BC123"));  }  [Fact]  //проверка метода, выводящего количество номеров в заданном диапазоне  public void GetCombinationsCountInRange\_ValidMarks\_ReturnsCount()  {  Assert.Equal(1, RegMarkLib.GetCombinationsCountInRange("A123BC123", "A124BC123"));  }  }  } |