# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №1**

**«Розробка алгоритму руху роботів»**

**з дисципліни «Конструювання програмного забеспечення»**

Виконав:

слухач групи ПЗС-21

Гринчук Тарас

Прийняв:

ст. викл. Тушницький Р.Б.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

**Тема роботи**: Розробка алгоритму руху роботів.

**Мета роботи**: Ознайомлення з засобами розробки Visual Studio та Resharper. Навчитись створювати проекти, підключати бібліотеки, відлагоджувати програми.

## Хід роботи

В середовищі Visual Studio створимо новий проект «бібліотеку класів» (рис. 1).

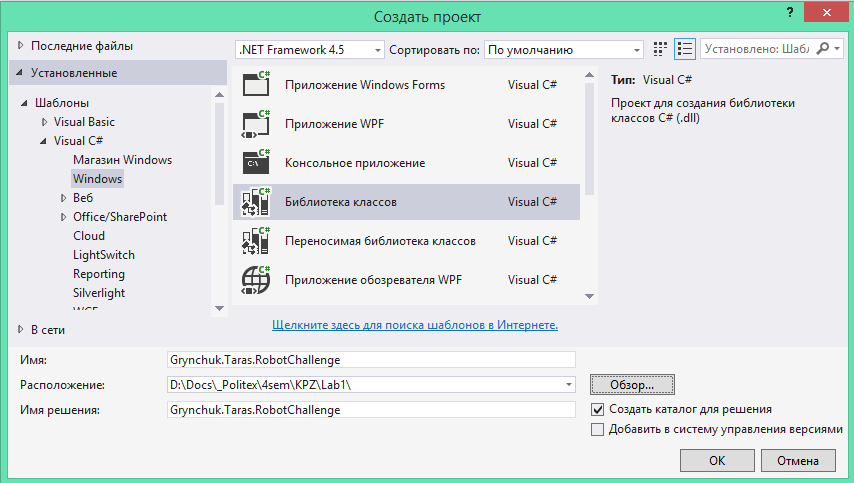


Рис. 1. Створення бібліотеки класів

З’явилось наступне вікно проекту (рис. 2):

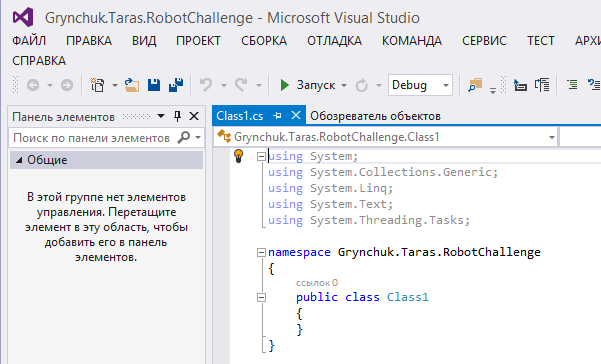


Рис. 2. Початкове вікно проекту

Додаємо посилання на бібліотеку, яка входить в пакет лабораторної, та у якій знаходяться базові класи алгоритму. Для цього правою кнопкою миші тиснемо на елемент “Ссылки на проект” у панелі Окно классов, і вибираємо в контекстному меню “Добавить ссилку” (рис. 3).

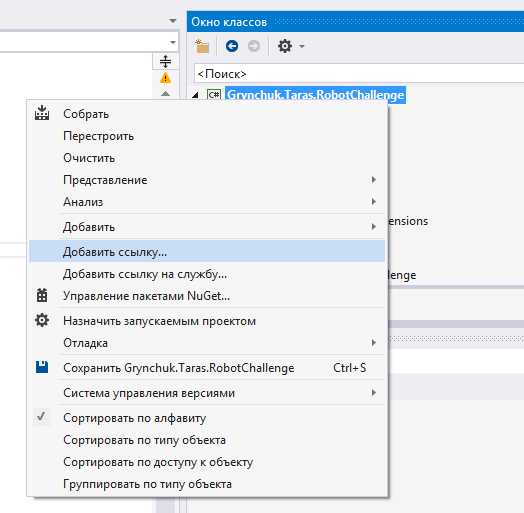


Рис. 3. Додаванна ссилки

Переходимо на закладку “Browse”, яка надає можливість підключення бібліотек .Net платформи із диску комп’ютера, знаходимо бібліотеку Robot.Common, яка входить в пакет лабораторної, і підключаємо її (рис. 4).

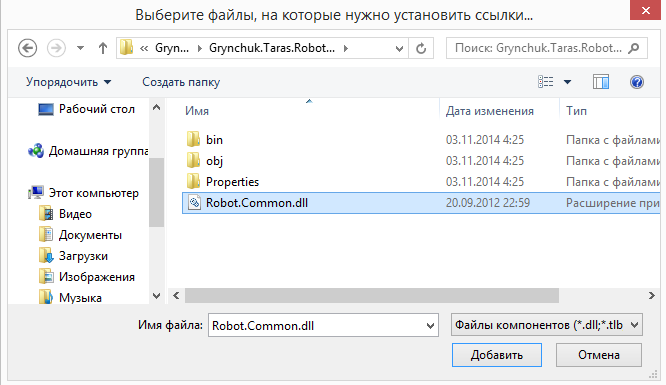


Рис. 4. Вибір файлу бібліотеки в діалозі

Після додавання бібліотеки вона з’явилася у списку підключених бібліотек (рис. 5).

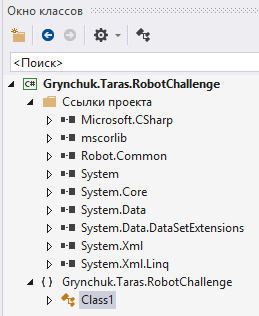


Рис. 5. Список бібліотек

Перейменуємо файл Class1 на іншу назву, дамо ім’я GrynchukAlgorythm (рис. 6).

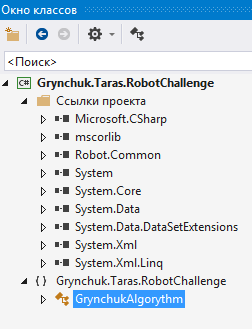


Рис. 6. Перейменування класу

Тепер перейдемо до вікна коду, і дослідимо що знаходиться у бібліотеці Robot.Common. Для цього її можна переглянути бібліотеку у панелі “Окно классов” (рис. 7).

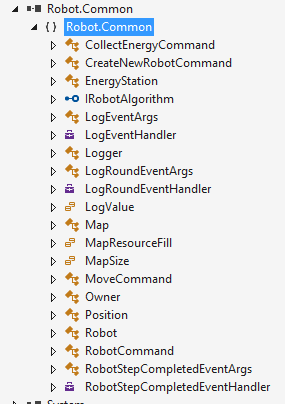


Рис. 7. Бібліотеці Robot.Common

У бібліотеці є класи карти (Map), позиції (Position), робота (Robot), команди для роботів, і інтерфейс IRobotAlgorithm, який повинен бути реалізований. Наш клас повинен наслідувати цей алгоритм, тобто ставимо дві крапки і назву IRobotAlgorithm. Натиснемо Alt+Enter, і Resharper автоматично вставить відповідні директиви (using Robot.Common) (рис. 7).

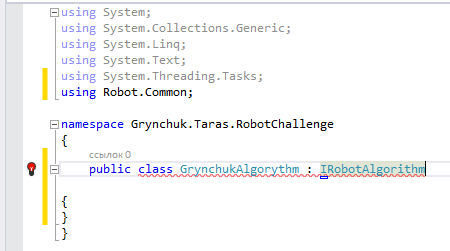


Рис. 7. Робота з кодом

За допомогою ReSharper реалізуємо інтерфейс IRobotAlgorithm (рис. 8).

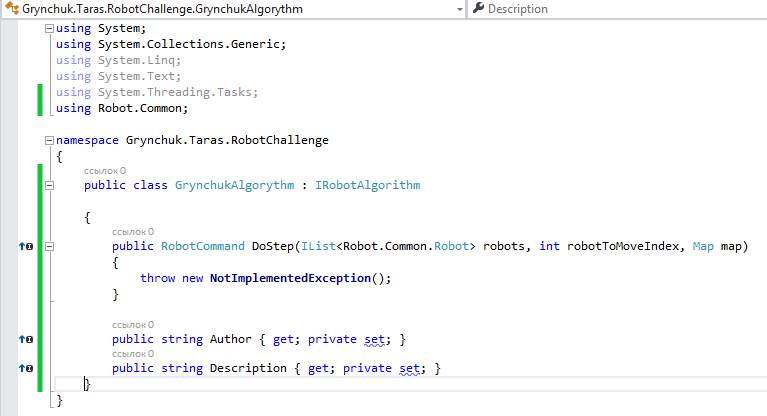


Рис. 8. Початкова реалізація інтерфейсу IRobotAlgorithm

Заповнимо властивості Author та Description власною інформацією (рис. 9).

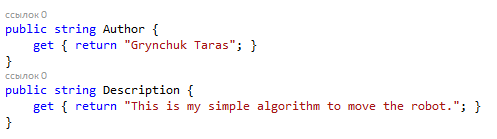


Рис. 9. Заповнення властивостей Author та Description

Отже, залишилось реалізувати метод DoStep, який отримає усю інформацію та повертає команду. Команди які може робити робот, які мають спільний інтерфейс IRobotCommand із єдиним методом Apply.

Як приклад, реалізуємо дуже простий варіант алгоритму. Нехай робот знаходить найближчу станцію, не зайняту іншими роботами, і рухається до неї зі сталою швидкістю. Якщо він знаходить на станції, то нехай там і залишається. Для цього нам необхідно запрограмувати функцію, яка повертає найближчу незайняту станцію.

Додамо її в окремий допоміжний клас. Напишемо нашу допоміжну функцію (рис. 10).

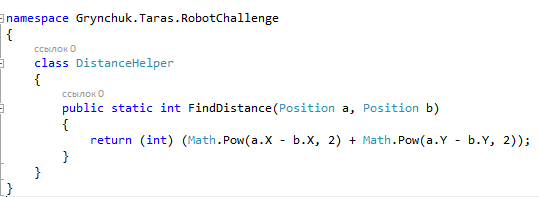


Рис. 10. Допоміжний клас

Тепер напишемо наш алгоритм. Для цього реалізуємо функції : IsStationFree – функцію, яка визначає чи станція незайнята; FindNearestFreeStation – функцію, яка визначає найближчу вільну станцію, і відповідно наш алгоритм:

public class GrynchukAlgorythm : IRobotAlgorithm

{

public Position FindNearestFreeStation(Robot.Common.Robot movingRobot, Map map, IList<Robot.Common.Robot> robots)

{

EnergyStation nearest = null;

int minDistance = int.MaxValue;

foreach (var station in map.Stations)

{

if (IsStationFree(station, movingRobot, robots))

{

int d = DistanceHelper.FindDistance(station.Position, movingRobot.Position);

if (d < minDistance)

{

minDistance = d;

nearest = station;

}

}

}

return nearest == null ? null : nearest.Position;

}

public bool IsStationFree(EnergyStation station, Robot.Common.Robot movingRobot, IList<Robot.Common.Robot> robots)

{

return IsCellFree(station.Position, movingRobot, robots);

}

public bool IsCellFree(Position cell, Robot.Common.Robot movingRobot, IList<Robot.Common.Robot> robots)

{

foreach (var robot in robots)

{

if (robot != movingRobot)

{

if (robot.Position == cell)

return false;

}

}

return true;

}

public RobotCommand DoStep(IList<Robot.Common.Robot> robots, int robotToMoveIndex, Map map)

{

/\*

if ((movingRobot.Energy > CreateNewRobotCommand.MinEnergyToCreateNewRobot) && (robots.Count < map.Stations.Count))

{

return new CreateNewRobotCommand();

}

\*/

Robot.Common.Robot movingRobot = robots[robotToMoveIndex];

Position stationPosition = FindNearestFreeStation(robots[robotToMoveIndex], map, robots);

if (stationPosition == null)

return null;

if (stationPosition == movingRobot.Position)

return new CollectEnergyCommand();

else

{

Position newPosition = stationPosition;

int distance = DistanceHelper.FindDistance(stationPosition, movingRobot.Position);

if (distance > 20)

{

int dx = Math.Sign(stationPosition.X - movingRobot.Position.X) \* Math.Min(Math.Abs(stationPosition.X - movingRobot.Position.X), 5);

int dy = Math.Sign(stationPosition.Y - movingRobot.Position.Y) \* Math.Min(Math.Abs(stationPosition.Y - movingRobot.Position.Y), 5);

newPosition = new Position(movingRobot.Position.X + dx, movingRobot.Position.Y + dy);

}

return new MoveCommand() { NewPosition = newPosition };

}

}

public string Author {

get { return "Grynchuk Taras"; }

}

public string Description {

get { return "This is my simple algorithm to move the robot."; }

}

}

Відлагодимо нашу бібліотеку (рис. 11):

1) Скомпілюємо і скопіюємо бібліотеку з алгоритмому папку RobotChallenge/Program/AlgorithmDlls.

2) Запустимо програму RobotChallenge.exe

3) Після цього приєднаємось до процесу RobotChallenge. Для цьог зайдемо в меню Отладка>присоединиться к процессу…

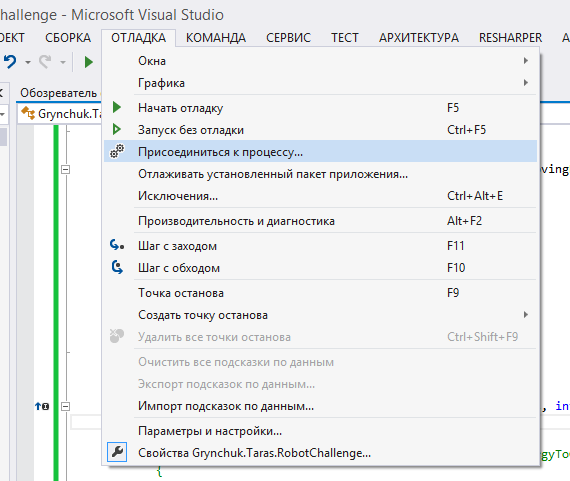


Рис. 11. Приєднання до процесу

4) Виберемо у вікні зі списком процесів процес RobotChallange.exe і натиснемо кнопку Присоединиться (рис. 12).

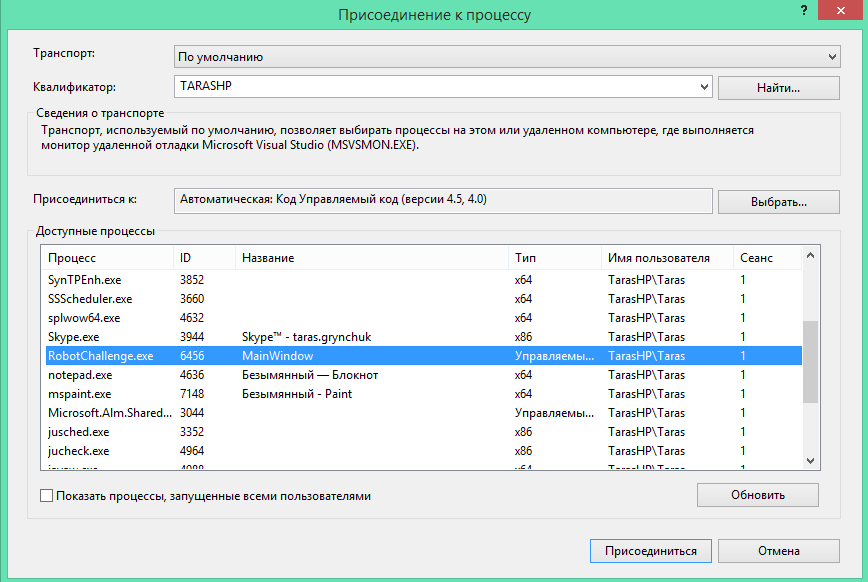


Рис. 12. Вибір процесу для приєднання

5) Ставимо брейкпойнт у функції DoStep і натискаємо кнопку “Start” у програмі RobotChallange.exe. Як бачимо програма зайшла зупинилась на брейкпойнті (рис. 13).

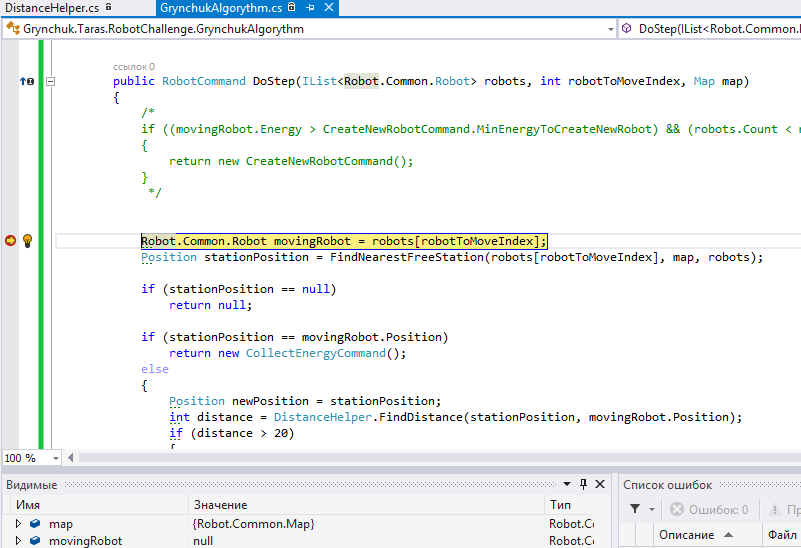


Рис. 13. Зупинка програми на брейкпойнті

**Створення модульних тестів**

1. Запустимо **Visual Studio.**
2. В меню **Файл** виберемо пункт **Создать**, а потім команду **Проект**.
3. Відкриється діалогове вікно **Создание проекта**.
4. В області **Установленные шаблоны** виберемо шаблон **Visual C #.**
5. У списку типів виберемо пункт **Библиотека классов**.
6. В поле Ім'я введемо **Bank** і натиснимо кнопку **ОК** (рис. 14).

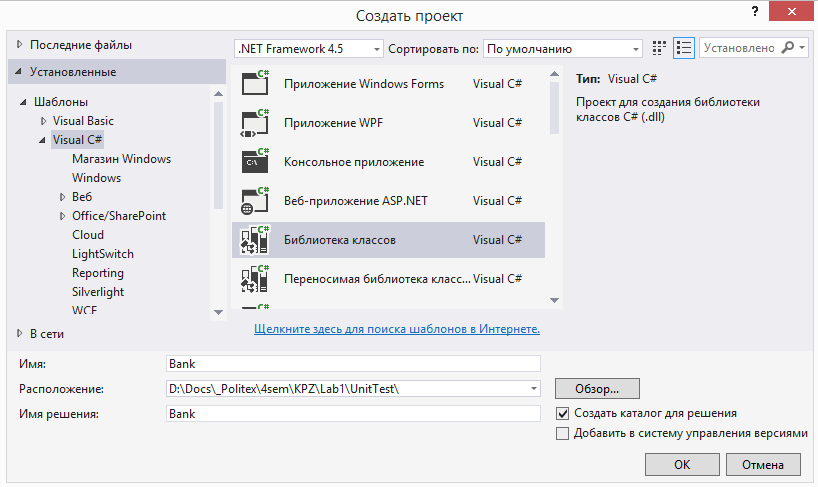


Рис. 14. Створення проекту

1. Заповнимо файл Class1.cs наступним кодом:

using System;

namespace BankAccountNS

{

/// <summary>

/// Bank Account demo class.

/// </summary>

public class BankAccount

{

private string m\_customerName;

private double m\_balance;

private bool m\_frozen = false;

private BankAccount()

{

}

public BankAccount(string customerName, double balance)

{

m\_customerName = customerName;

m\_balance = balance;

}

public string CustomerName

{

get { return m\_customerName; }

}

public double Balance

{

get { return m\_balance; }

}

public void Debit(double amount)

{

if (m\_frozen)

{

throw new Exception("Account frozen");

}

if (amount > m\_balance)

{

throw new ArgumentOutOfRangeException("amount");

}

if (amount < 0)

{

throw new ArgumentOutOfRangeException("amount");

}

m\_balance += amount;

}

public void Credit(double amount)

{

if (m\_frozen)

{

throw new Exception("Account frozen");

}

if (amount < 0)

{

throw new ArgumentOutOfRangeException("amount");

}

m\_balance += amount;

}

private void FreezeAccount()

{

m\_frozen = true;

}

private void UnfreezeAccount()

{

m\_frozen = false;

}

public static void Main()

{

BankAccount ba = new BankAccount("Mr. Bryan Walton", 11.99);

ba.Credit(5.77);

ba.Debit(11.22);

Console.WriteLine("Current balance is ${0}", ba.Balance);

}

}

}

1. Збережемо файл під назвою **BankAccount.cs**
2. В меню **Построение** выберемо **Построить решение**.

Створення проекту модульного тесту

1) В меню **Файл** послідовно виберемо **Добавить** і **Новый проект....**

2) В діалоговому вікні **"Новый проект"** выберемо **Тест.**

3) В списку шаблонів виберемо **Проект модульного теста.**

4) В поле **Имя** введемо **BankTest** и нажмемо кнопку **ОК** (рис. 15).

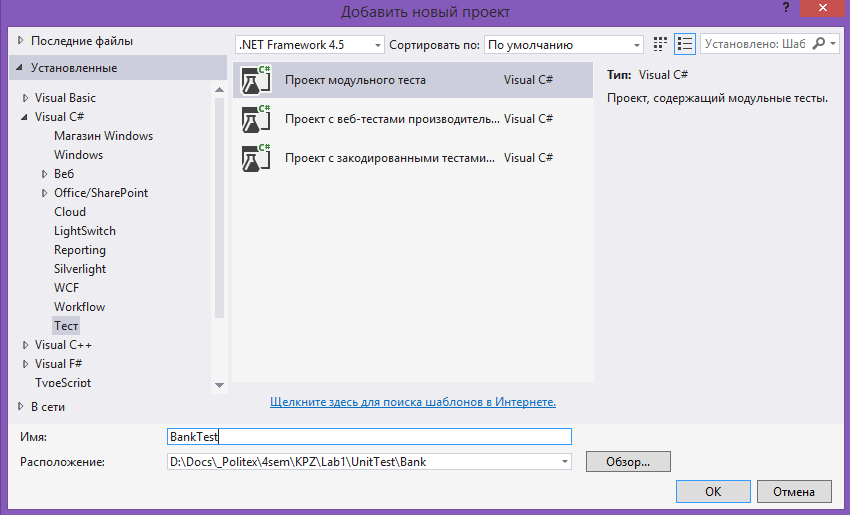


Рис. 15. Вікно створення модульного теста

5) Проект BankTests додається в рішення Банк.

6) В проекте BankTests добавимо ссилку на рішення Банк (рис. 16).

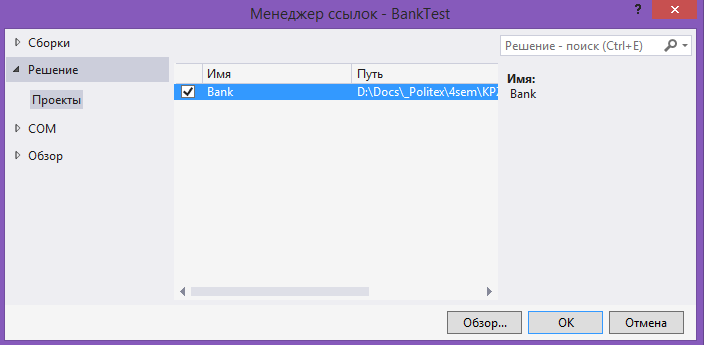


Рис. 16. Додавання ссилки на проект

У браузері рішень виберемо файл **UnitTest1.cs** в проекті **BankTests**. В контекстному меню виберемо команду **Rename**, а потім перейменуйvj файл в **BankAccountTests.cs**. Виберемо Так у діалоговому вікні, що пропонує перейменувати всі посилання на елемент коду "UnitTest1" в проекті. Цей крок змінює ім'я класу на BankAccountTest.

Файл BankAccountTests.cs тепер містить наступний код (рис. 17):

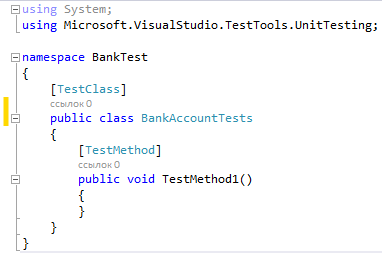
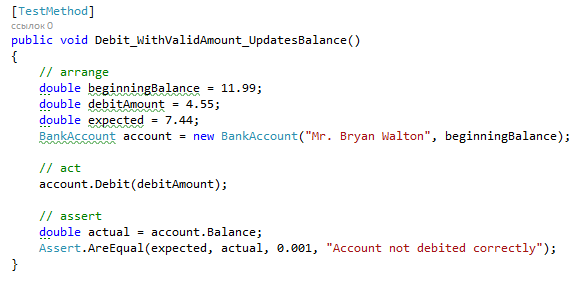


Рис. 17. Код після перейменування

Додамо інструкцію **using BankAccountNS;** в файл **BankAccountTests.cs.**

Додамо наступний метод в цей клас **BankAccountTests**:



Цей метод досить простий. Ми створюємо новий об'єкт BankAccount з початковим балансом, а потім знімаємо допустиме значення. Використовуємо платформу модульних тестів Microsoft для методу AreEqual керованого коду, щоб перевірити відповідність кінцевого балансу очікуваному.

Запуск теста:

1) В меню **Построение** виберемо **Построить решение**.

В меню **Тестирование/Окна** виберемо **Обозреватель тестов** (рис. 18).

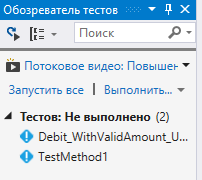


Рис. 18. Вікно «Обозреватель тестов»

2) Виберемо **Запустить все**, щоб виконати тест. Під час виконання тесту в верхній частині вікна відображається рядок стану. По завершенні тестового запуску рядок стану стає зеленим, якщо всі методи тесту успішно пройдені, або червоним, якщо якісь із тестів не пройдені.

3) В даному випадку тест пройдено не буде (рис. 19).

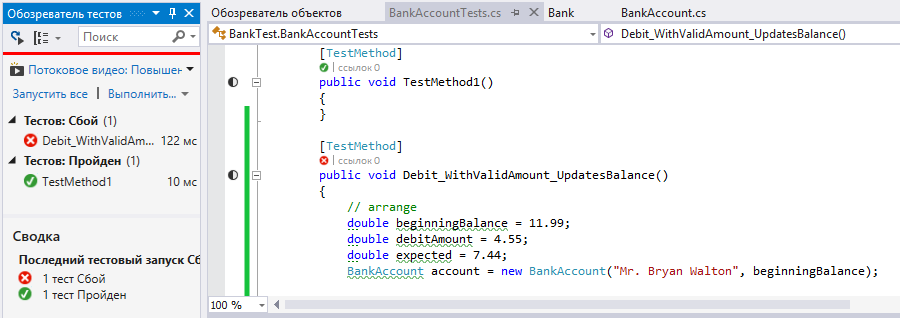


Рис. 19. Результат виконання тестів

## ВИСНОВКИ

На даній лабораторній роботі я ознайомився з засобами розробки Visual Studio та Resharper. Навчився створювати проекти, підключати бібліотеки, відлагоджувати програми, створювати модульні тести.