МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Звіт з лабораторної роботи № 1

з предмету «Якість тестування ПЗ»

Виконав:

Студент групи КН-36а

Пашко А. І.

Перевірив:

Копп А. М.

Харків 2018

**Цель работы:**

По заданию преподавателя подготовить три программы на языках программирования Java, C# и одном по выбору студента*.*

*Требования к программам:*

1. *Наличие графического интерфейса*
2. *Подготовка сопроводительной документации в нотации UML (диаграммы вариантов использования, классов и деятельности).*

На основе заданных преподавателем метрикам провести расчет показателей качества разработанных программ*.*

*Требования к отчету:*

1. *Титульный лист.*
2. *Документации в нотации UML.*

*Листинги разработанных программ.*

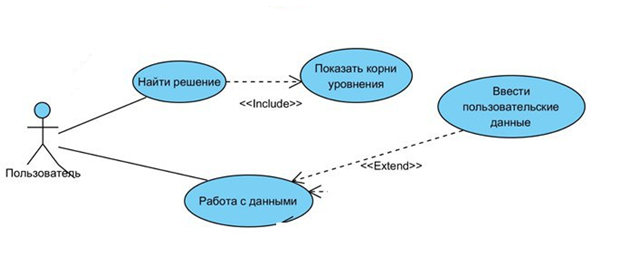
**Діаграма варіантів** використання є вихідним концептуальним уявленням або концептуальною моделлю системи в процесі її проектування і розробки.

Розробка діаграми варіантів використання програмного забезпечення дозволить:

1. визначити загальні межі і контекст модельованої предметної області на початкових етапах проектування системи;
2. сформулювати загальні вимоги до функціональної поведінки програмного забезпечення;
3. розробити вихідну концептуальну модель програмного забезепечення для його подальшої деталізації у формі логічних і фізичних моделей;
4. підготувати вихідну документацію для взаємодії розробників програмного забезпечення з його замовниками і користувачами.

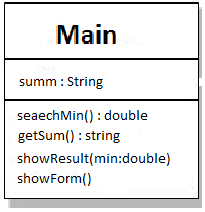
Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання та містять так звані зв’язки між собо. Ці зв’язки відображають взаємодію (такі, як включення або розширення, або include та extend) або переход від одного варіанту використання, до іншого.

Вимоги до програмного забезпечення відображаються на діаграмі варіантів використання, представленої на рисунку 1.

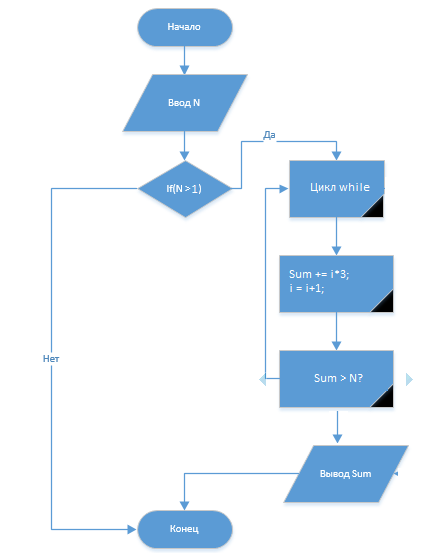


**Діаграма класів**

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.



**Діаграма діяльності** (англ. activity diagram) — в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.



**Код программы на C#:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Minimal

{

public partial class Main : Form

{

private String sum;

public double searchMin()

{

sum = getSum();

int i = 1, summ = 0;

while (true)

{

summ += (i \* 3);

i++;

if (summ > Convert.ToDouble(sum))

{

return i - 1;

}

}

}

public string getSum()

{

return textBox1.Text;

}

public Main()

{

showForm();

}

public void showForm()

{

InitializeComponent();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

showResult(searchMin());

}

private void showResult(double min)

{

textBox2.Text = Convert.ToString(min);

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

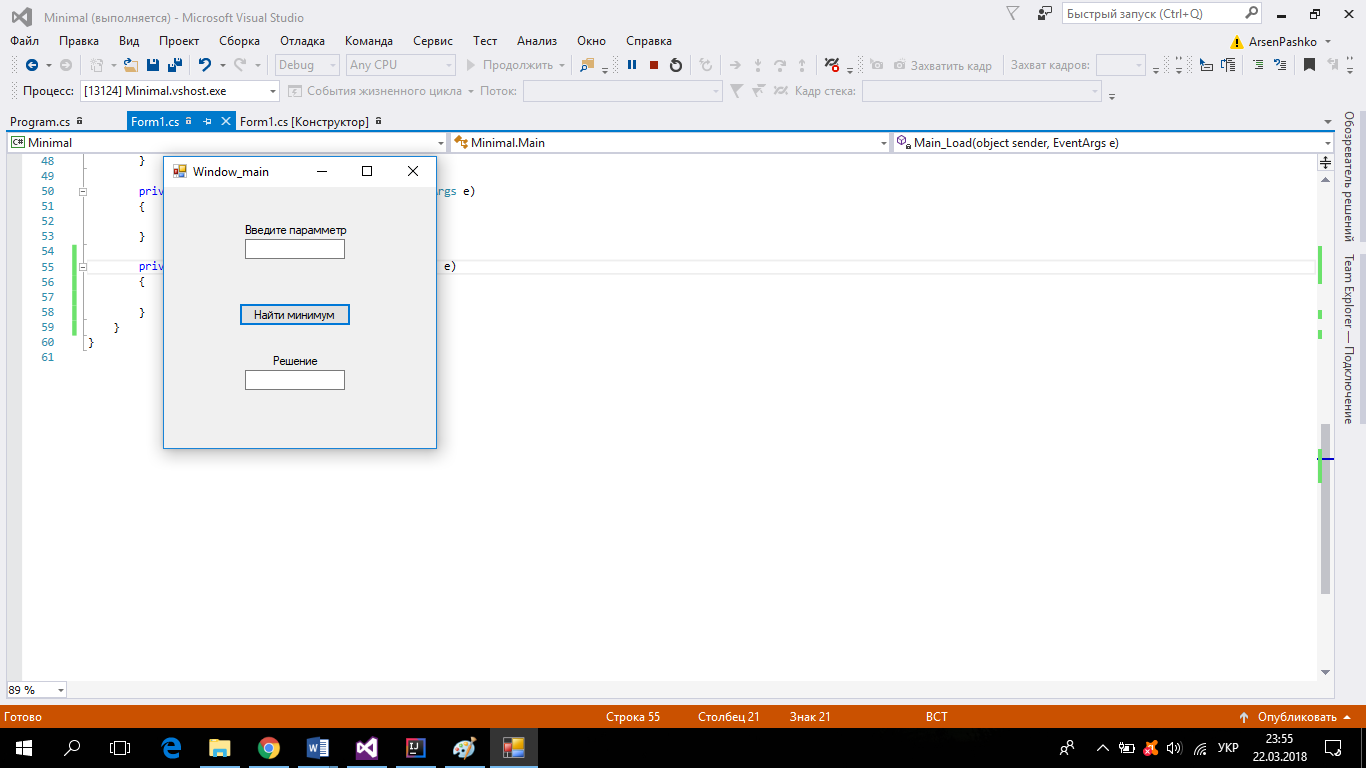
private void Main\_Load(object sender, EventArgs e)

{

}

}

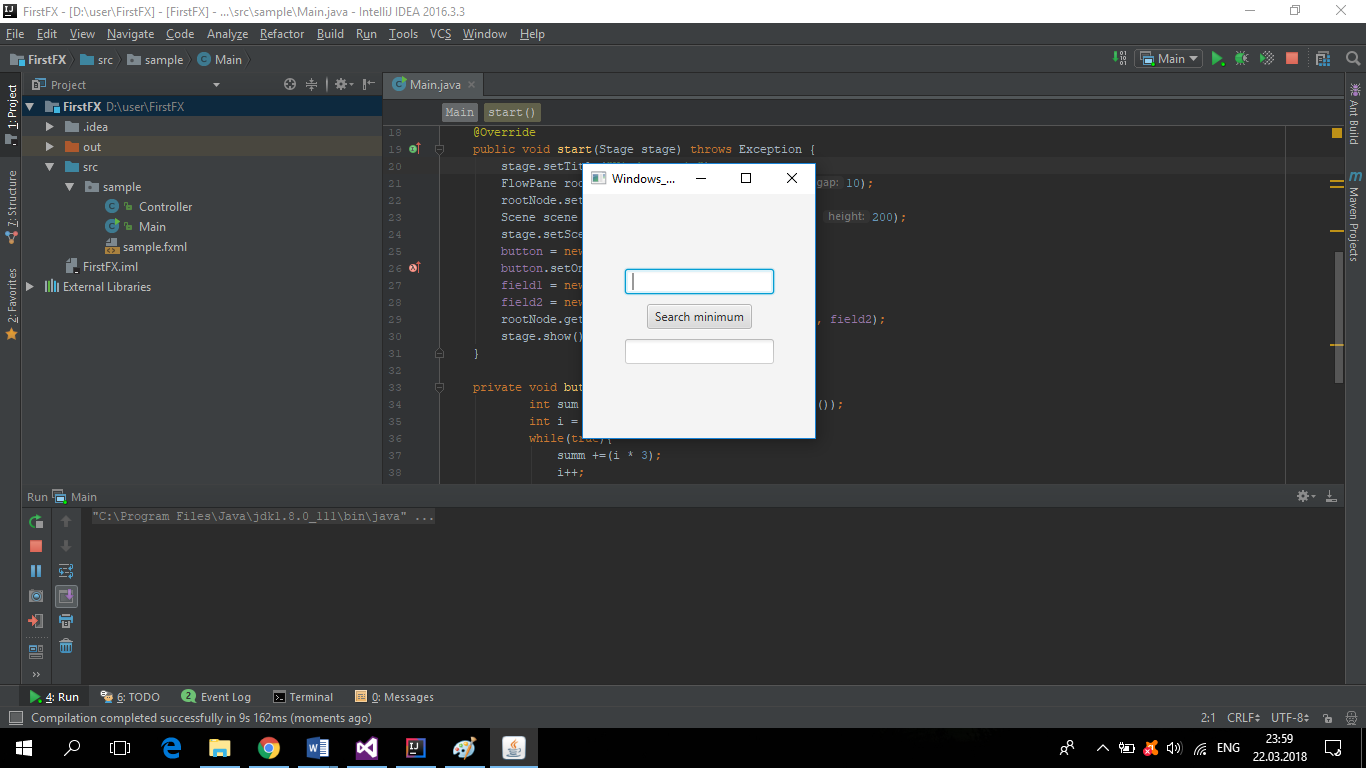
}



Робоче вікно програми на C#

**Код програми на JAVA:**

package sample;  
  
import javafx.application.Application;  
import javafx.event.ActionEvent;  
import javafx.event.Event;  
import javafx.event.EventHandler;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.layout.FlowPane;  
import javafx.scene.control.Button;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
  
public class Main extends Application {  
 private Button button;  
 private TextField field1, field2;  
 private Stage stage;  
 private String sum;  
  
 @Override  
 public void start(Stage stage) throws Exception {  
 this.stage = stage;  
 showForm();  
 }  
  
 private void showForm() {  
 stage.setTitle("Windows\_main");  
 FlowPane rootNode = new FlowPane(10, 10);  
 rootNode.setAlignment(Pos.*CENTER*);  
 Scene scene = new Scene(rootNode, 200, 200);  
 stage.setScene(scene);  
 button = new Button("Search minimum");  
 button.setOnAction(event -> showResult(searchMin()));  
 field1 = new TextField();  
 field2 = new TextField();  
 rootNode.getChildren().addAll(field1, button, field2);  
 stage.show();  
 }  
  
 private String getSum() {  
 return field1.getText();  
 }  
  
 private double searchMin() {  
 int sum = Integer.*parseInt*(getSum());  
 int i = 1, summ = 0;  
 while (true) {  
 summ += (i \* 3);  
 i++;  
 if (summ > sum) {  
 break;  
 }  
 }  
 return i;  
 }  
  
 private void showResult(double min) {  
 field2.setText((min- 1) + "");  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
  
}



Робоче вікно програми на JAVA

**Метрики**

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозн. метрика** | **Краткое описание** | **Знач. C#** | **Знач. Java** |
| *n1* | Число уникальных операторов | 9 | 7 |
| *N1* | общее число всех операторов | 20 | 13 |
| *n2* | число уникальных операндов | 4 | 5 |
| *N2* | общее число всех операндов | 5 | 6 |
| *n* | *n = n1 + n2* – словарь программы | 13 | 12 |
| *N* | *N = N1 + N2* – длина программы | 25 | 19 |
| *N'* | теоретическая длина программы | 4.8 | 4 |
| *V* | *V = N \* log2(n)* – объем программы | 7,8 | 6,8 |
| *V'* | *V' = N' \* log2(n')* – теоретический объем программы | 6,9 | 9 |
| *L* | *L = V' / V* – уровень качества программы | 0,89 | 0,81 |
| *λ* | *λ = L / V'* – уровень языка программирования | 0.12 | 0,14 |
| *D* | *D = 1 / L'* – трудоемкость кодирования программы | 0,17 | 1,23 |
| *I* | *I = V / D* – информационное содержание программы | 4,5 | 4 |
| *E* | *V / L* – необходимые интеллектуальные усилия | 7,75 | 8,8 |
| *T* | *T = E / S* – время составления программы, (*S = 18*) | 0,4 | 0,5 |
| *T'* | оценка времени составления программы | 9,5111 | 12,21 |
| *B* | *= E2/3 / 3000* – число дефектов в программе | 0.003 | 0,003 |
| *B'* | *B' = V / 3000* – оценка числа дефектов в программе | 0,02 | 0.1 |

**Висновок:** під час виконання лабораторної роботи було розроблене програмне забезпечення, яке знаходить мінімум числа N. Программа була виконана на двох мовах(C# та Java), та має графічний застосунок.