МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Звіт з лабораторної роботи № 1

з предмету «Якість тестування ПЗ»

Варіант №9

Виконав:

Студент групи КН-36а

Кулик В.В.

Перевірив:

Копп А. М.

Харків 2018

**Мета роботи:**

За завданням викладача підготувати три програми на мовах програмування Java, C# і одному за вибором студента.

*Вимоги до програм:*

1. *Наявність графічного інтерфейсу*
2. *Підготовка супровідної документації в нотації UML (діаграми варіантів використання, класів і діяльності).*

На основі заданих викладачем метрик провести розрахунок показників якості розроблених програм*.*

*Вимоги до звіту:*

1. *Титульний аркуш.*
2. *Документації в нотації UML.*
3. *Лістниги розроблених програм.*

**Діаграма варіантів**

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.

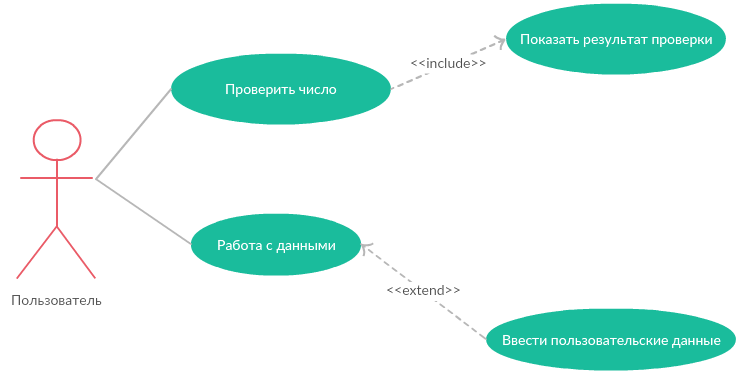


Рисунок 1 - Діаграма варіантів

**Діаграма класів**

Діаграма класів – статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення (рисунок 2).

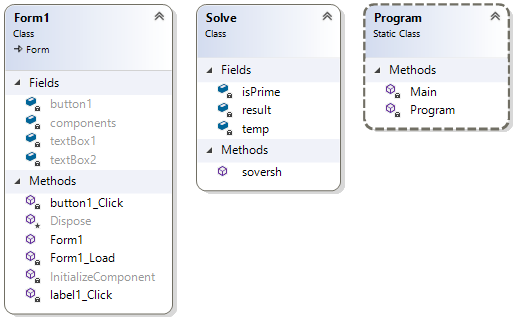


Рисунок 2 - Діаграма класів

**Діаграма діяльності**

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.

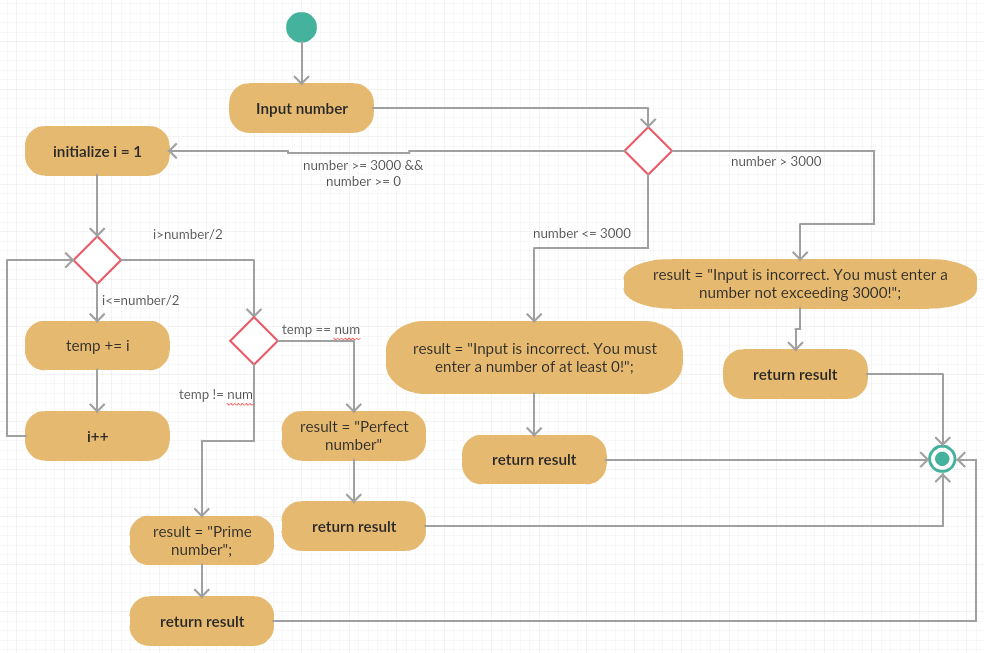


Рисунок 3 - Діаграма діяльності

**Код програми на C#:**

*Solve.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Perfect\_Number

{

class Solve

{

Boolean isPrime = true;

int temp;

String result;

public String soversh(int num)

{

if (num <= 3000 && num >= 0)

{

for (int i = 1; i <= num / 2; i++)

{

if (num % i == 0)

{

temp += i;

}

}

if (temp == num)

{

//System.out.println(num + " - простое число");

result = "Perfect number";

return result;

}

else

{

//System.out.println(num + " - совершенное число");

result = "Prime number";

return result;

}

}

else if (num > 3000)

{

result = "Input is incorrect. You must enter a number not exceeding 3000!";

return result;

}

else

{

result = "Input is incorrect. You must enter a number of at least 0!";

return result;

}

}

}

}

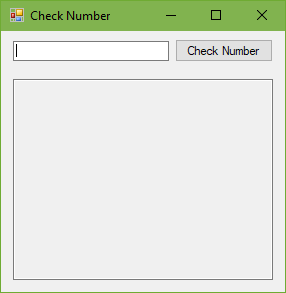


Рисунок 4 - Робоче вікно програми на C#

**Код програми на Java:**

*Controller.java*

package sample;

import javafx.fxml.Initializable;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import java.net.URL;

import java.util.ResourceBundle;

public class Controller implements Initializable{

public Button pressBtn;

public TextField txtNum;

public TextArea txtResult;

@Override

public void initialize(URL location, ResourceBundle resources) {

}

public void main(javafx.event.ActionEvent actionEvent) {

int N;

if (txtNum.getText().matches("[0-9]+$")){

N = Integer.parseInt(txtNum.getText());

Solve s = new Solve();

txtResult.setText(s.soversh(N));

} else {

txtResult.setText("Input is incorrect. You must enter a number!");

}

}

}

*Solve.java*

*package sample;*

*public class Solve {*

*boolean isPrime = true;*

*int temp = 0;*

*String result;*

*public String soversh(int num) {*

*if (num <= 3000 && num >= 0) {*

*for(int i=1;i<=num/2;i++){*

*if(num%i == 0){*

*temp += i;*

*}*

*}*

*if (temp == num) {*

*//System.out.println(num + " - простое число");*

*result = "Perfect number";*

*return result;*

*} else {*

*//System.out.println(num + " - совершенное число");*

*result = "Prime Number";*

*return result;*

*}*

*} else if (num > 3000) {*

*result = "Input is incorrect. You must enter a number not exceeding 3000!";*

*return result;*

*} else {*

*result = "Input is incorrect. You must enter a number of at least 0!";*

*return result;*

*}*

*}*

*}*

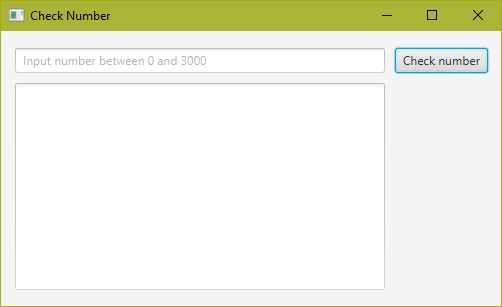


Рисунок 5 - Робоче вікно програми на Java

**Таблиця показників**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | n1 | n2 | N1 | N2 | n | N |  | V | V\* | n2\* |
| C# | 14 | 16 | 47 | 39 | 30 | 86 | 117,30296 | 421,99259 | 8 | 2 |
| Java | 12 | 16 | 43 | 38 | 28 | 81 | 107,01955 | 389,39574 | 8 | 2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | L |  | I | E | T | B | Mх | Мм |
| Java | 0,02054 | 0,16435 | 27,32601 | 18953 | 1895 | 0,12979 | 0,33333 | 4 |
| C# | 0,01895 | 0,15166 | 24,73216 | 22259 | 2225 | 0,14066 | 0,33333 | 4 |

**Висновок:** під час виконання лабораторної роботи було розроблене програмне забезпечення, яке перевіряє число N на досконалість. Програма була виконана на двох мовах(C# та Java), та має графічний застосунок.