МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ ІНСТИТУТ НОВІТНІХ ТЕХНОЛОГІЙ

ТА УПРАВЛІННЯ ІМЕНІ ВЯЧЕСЛАВА ЧОРНОВОЛА

###### Факультет інформаційних технологій

ДИПЛОМНА РОБОТА

на тему:

«Створення калькулятора (множення)»

Львів – 2008р

Тема: «Створення калькулятора»

Мета: скласти алгоритм програми калькулятор.

Завдання: розробити клас, який реалізує функцію множення. Зобразити у вигляді UML діаграм структуру розробленого програмного продукту.

Хід роботи:

1. Визначення інтерфейсу калькулятора.

Вікно калькулятора, має наступні кнопки:

* Кнопка множення.
* Набір кнопок з цифрами від 0 до 9.
* Кнопка очистити.
* Кнопка виконання операції.

Потім користувач відкриває програму калькулятор, в якій в полі вводу-виводу вже стоїть 0 по замовчуванню.

1. Визначення дій після того, як відбувся запуск програми калькулятор.

Виділяємо певну область пам'яті, щоб мати можливість зберігати вхідні дані і результат.

* Якщо натиснути кнопку С то очиститься вікно редагування(вводу виводу).
* Якщо натиснути кнопки із цифрами від 0 до 9 натиснувши комбінацію цифр, вона занесеться в вікно редагування (вводу), після чого ми натискаємо кнопку виконання операції.

Після обчислення даних, результат знову повертається в вікно (вводу-виводу), в системну мітку яка прихована за вікном редагування.

В нашому випадку це кнопка \* (множення).

* Вивести результат на екран, в результаті натискання кнопки \*.

Це ми розглянули основні операції при обчисленні десяткового логарифму.

* Тепер ми розглянемо систему вводу чисел. Це здійснюється таким чином, кожній кнопці присвоюється значення цифри, яке копіюється в поле вводу-виводу. Це рівноцінно вводу значень з клавіатури.

Діаграма діяльності програми «калькулятор»

Ввід даних( за допопомогою кнопок(цифр))

Виконати операцію

з числом

Вивести результат в вікно вводу-виводу

Очистити вікно вводу-виводу

 / Введено правильно

ні

так

так

**стоп**

**старт**

Діаграми діяльності забезпечують динамічне подання системи. Вони особливо важливі при моделюванні функціональності системи і виділяють потік керування між об’єктами. “Динамізм” таких моделей полягає в тому, що в них відображається зміна стану в процесі роботи системи.

Засоби мови UML, для створення динамічних моделей багато чисельні та різнобічні. Ці засоби орієнтовані не тільки на власні програмні системи, але і на відображення вимог замовника до поведінки таких систем. Для моделювання поведінки системи використовують:1.) автомати, 2.) взаємодія.

* Автомати (State machine)описують поведінку в терміналах послідовність станів через які проходить об’єкт на протязі свого життя. Таким чином, автомат задає поведінку системи як цілої, єдиної системи; моделює життєвий цикл єдиного об’єкту. В силу автоматичний підхід зручно застосовувати для формалізації динаміки окремого складного для розуміння блоку системи.
* Взаємодія (Interaction) описує поведінку в терміналах обміну повідомленнями між об’єктами. Також взаємодії визначають поведінку системи у вигляді комунікації між її частинами (об’єктами), подаючи систему як сукупність об’єктів які працюють разом. І тому взаємодія є основним апаратом для фіксації повної динаміки системи.

Виходячи із вище вказаного можна сказати що діаграмна діяльність - це спеціальна різновидність діаграми схем станів, яка вказує потік від дії до дії в середовищі системи.

Діаграма схем станів

Діаграма схем станів – одна з п’яти діаграм UML, що моделюють динаміку систем. Діаграма схем станів відображає скінчений автомат, виділяючи потік керування, який переходить від стану до стану. Скінчений автомат – поведінка, що розглядаються як відповідь на події та включає реакцію на ці події.

Ввід даних

Main клас

тангенс

Конвертування величин і обчислення

вихід

вивід

Даним знаком позначається вибір: дії та операції

Функціональна схема програми «Калькулятор»

Для отримання результату при виконанні будь якої тригонометричної чи арифметичної операції не можна обійтись без поля вводу даних в певні поля і натиснути кнопку завершення операції тобто дорівнює, після чого результат виведеться в поле виводу даних. Це видно із функціональної схеми показаної нижче.

Введення числових даних в поле вводу -виводу

Вибір функції множення

Виконання вибраної операції

Вивід результату в поле

виводу

Умовно периферійні складові нашої програми можна поділити на:

1. ввід даних;
2. вивід даних;
3. обчислення даних;
4. супроводжуючий графічний інтерфейс.

Діаграми в UML

Діаграма – графічне подання множини елементів, що найчастіше зображається зв’язним графом з вершин (предметів) і дуг (відношень). Діаграми малюються візуалізації системи з різних точок зору, потім перетворюються в систему. Як правило діаграма дає неповне відображення елементів, які створюють систему. Хоча один і той же елемент може появитись у всіх діаграмах, на практиці він з’являється тільки в деяких. Теоретично діаграма може містити довільну комбінацію предметів, які відповідають п’яти поданням (зображенням) архітектури ПС. Тому UML містить девять видів діаграм:

1. діаграми класів;
2. діаграми об’єктів;
3. діаграми UseCase (діаграми прецедентів);
4. діаграми послідовності;
5. діаграми співробітництва;
6. діаграми схем станів;
7. діаграми діяльності;
8. компонентні діаграми
9. діаграми розміщення (розгортання);

Статична діаграма класів

Діаграма класів показує набір класів, інтерфейсів, кооперацій та їх відношень. При моделюванні об’єктно - орієнтованих систем діаграми класів використовуються найбільш часто. Діаграми класів забезпечують статичне проектне подання системи.

Діаграми класів, що включають активні класи, забезпечують статичне подання процесів.



Протокол відображає графічне зображення (віконне) реалізації калькулятора (множення чисел).

Включає в себе наступні елементи та компоненти:

1. основна форма (Form1);
2. блок чисельної клавіатури кнопок (Button);
3. блок операційних кнопок (операцій ) визначення десяткового логарифму (Button);
4. поле для вводу даних (Edit1);
5. поле для відображення результату (на мітці) (Label1);

Протокол реалізації даної програми в (віконному) режимі JavaBuilder 2005

Лістінг програмного продукту «КАЛЬКУЛЯТОР» що перемножує числа

package rozr\_bincalc\_multiplication;

import java.awt.\*;

import javax.swing.\*;

import com.borland.jbcl.layout.XYLayout;

import com.borland.jbcl.layout.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class Rozr\_BinCalc\_Multiplication extends JFrame {

String x1 = "", x2, x3, str;

double d1, d2, res, b1 = 1.0, b2 = 2.0;

public Rozr\_BinCalc\_Multiplication() {

super();

setLocation(300, 100);

setSize(310, 360);

setVisible(true);

try {

jbInit();

} catch (Exception ex) {

ex.printStackTrace();

}

}

public static double roundDouble(double val, int places) {

long factor = (long)Math.pow(10,places);

val = val \* factor;

long tmp = Math.round(val);

return (double)tmp / factor;

}

public static void main(String[] args) {

new Rozr\_BinCalc\_Multiplication();

}

private void jbInit() throws Exception {

BOp = new Binary\_Operations();

this.getContentPane().setBackground(new Color(180, 166, 159));

this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);

this.setForeground(SystemColor.control);

this.setIconImage(null);

this.setResizable(false);

this.setTitle("Binary calculator - multiplication");

this.getContentPane().setLayout(xYLayout1);

xYLayout1.setWidth(442);

xYLayout1.setHeight(400);

jTextField1.setEditable(false);

jTextField1.setHorizontalAlignment(SwingConstants.RIGHT);

jButton1.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton1\_actionAdapter(this));

jButton2.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton2\_actionAdapter(this));

jButton3.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton3\_actionAdapter(this));

jButton8.setText("8");

jButton8.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton8\_actionAdapter(this));

jButton9.setText("9");

jButton9.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton9\_actionAdapter(this));

jButton10.setText(",");

jButton10.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton10\_actionAdapter(this));

jButton11.setText("0");

jButton11.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton11\_actionAdapter(this));

jButton12.setText("-");

jButton12.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton12\_actionAdapter(this));

jButton4.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton4\_actionAdapter(this));

jButton5.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton5\_actionAdapter(this));

jButton6.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton6\_actionAdapter(this));

jButton7.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton7\_actionAdapter(this));

jButton13.setText("\*");

jButton13.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton13\_actionAdapter(this));

jButton14.setText("=");

jButton14.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton14\_actionAdapter(this));

jButton15.setText("C");

jButton15.addActionListener(new Rozr\_BinCalc\_Multiplication\_jButton15\_actionAdapter(this));

this.getContentPane().add(jButton2, new XYConstraints(100, 80, 42, 42));

this.getContentPane().add(jButton1, new XYConstraints(40, 80, 42, 42));

this.getContentPane().add(jButton3, new XYConstraints(160, 80, 42, 42));

this.getContentPane().add(jButton7, new XYConstraints(40, 200, 42, 42));

this.getContentPane().add(jButton6, new XYConstraints(160, 140, 42, 42));

this.getContentPane().add(jButton5, new XYConstraints(100, 140, 42, 42));

this.getContentPane().add(jButton4, new XYConstraints(40, 140, 42, 42));

this.getContentPane().add(jButton8, new XYConstraints(100, 200, 42, 42));

this.getContentPane().add(jButton10, new XYConstraints(40, 260, 42, 42));

this.getContentPane().add(jButton9, new XYConstraints(160, 200, 42, 42));

this.getContentPane().add(jButton11, new XYConstraints(100, 260, 42, 42));

this.getContentPane().add(jButton12, new XYConstraints(160, 260, 42, 42));

this.getContentPane().add(jButton14, new XYConstraints(220, 260, 42, 42));

this.getContentPane().add(jTextField1, new XYConstraints(40, 23, 220, -1));

this.getContentPane().add(jButton15, new XYConstraints(220, 140, 42, 42));

this.getContentPane().add(jButton13, new XYConstraints(220, 200, 42, 42));

jTextField1.setText("0");

jButton1.setText("1");

jButton2.setText("2");

jButton3.setText("3");

jButton4.setText("4");

jButton5.setText("5");

jButton6.setText("6");

jButton7.setText("7");

}

Binary\_Operations BOp;

XYLayout xYLayout1 = new XYLayout();

JButton jButton1 = new JButton();

JTextField jTextField1 = new JTextField();

JButton jButton2 = new JButton();

JButton jButton3 = new JButton();

JButton jButton4 = new JButton();

JButton jButton5 = new JButton();

JButton jButton6 = new JButton();

JButton jButton7 = new JButton();

JButton jButton8 = new JButton();

JButton jButton9 = new JButton();

JButton jButton10 = new JButton();

JButton jButton11 = new JButton();

JButton jButton12 = new JButton();

JButton jButton13 = new JButton();

JButton jButton14 = new JButton();

JButton jButton15 = new JButton();

public void jButton1\_actionPerformed(ActionEvent e) {

x1 = x1 + "1";

jTextField1.setText(x1);

}

public void jButton2\_actionPerformed(ActionEvent e) {

x1 = x1 + "2";

jTextField1.setText(x1);

}

public void jButton3\_actionPerformed(ActionEvent e) {

x1 = x1 + "3";

jTextField1.setText(x1);

}

public void jButton4\_actionPerformed(ActionEvent e) {

x1 = x1 + "4";

jTextField1.setText(x1);

}

public void jButton5\_actionPerformed(ActionEvent e) {

x1 = x1 + "5";

jTextField1.setText(x1);

}

public void jButton6\_actionPerformed(ActionEvent e) {

x1 = x1 + "6";

jTextField1.setText(x1);

}

public void jButton7\_actionPerformed(ActionEvent e) {

x1 = x1 + "7";

jTextField1.setText(x1);

}

public void jButton8\_actionPerformed(ActionEvent e) {

x1 = x1 + "8";

jTextField1.setText(x1);

}

public void jButton9\_actionPerformed(ActionEvent e) {

x1 = x1 + "9";

jTextField1.setText(x1);

}

public void jButton10\_actionPerformed(ActionEvent e) {

x1 = x1 + ".";

jTextField1.setText(x1);

}

public void jButton11\_actionPerformed(ActionEvent e) {

x1 = x1 + "0";

jTextField1.setText(x1);

}

public void jButton12\_actionPerformed(ActionEvent e) {

if (Double.parseDouble(x1)>0)

{

x1 = "-" + x1;

jTextField1.setText(x1);

}

else

{

x1 = ""+(Double.parseDouble(x1)\*(-1))+"";

jTextField1.setText(x1);

}

}

public void jButton13\_actionPerformed(ActionEvent e) {

x2 = x1;

x1 = "";

}

public void jButton14\_actionPerformed(ActionEvent e) {

x3 = x1;

d1 = roundDouble(Double.parseDouble(x2), 10);

d2 = roundDouble(Double.parseDouble(x3), 10);

String str1 = BOp.Double\_to\_Binary(b1);

String str2 = BOp.Double\_to\_Binary(b2);

String ress = BOp.Bin\_Multiplication(str1,str2);

res = BOp.Binary\_to\_Double(ress);

res = roundDouble(d1\*d2, 10);

str = ""+res+"";

jTextField1.setText(str);

x1 = jTextField1.getText();

}

public void jButton15\_actionPerformed(ActionEvent e) {

jTextField1.setText("0");

x1 = "";

x2 = "";

x3 = "";

str = "";

d1 = 0;

d2 = 0;

}

class Rozr\_BinCalc\_Multiplication\_jButton1\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton1\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton1\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton2\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton2\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton2\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton3\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton3\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton3\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton4\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton4\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton4\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton5\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton5\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton5\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton6\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton6\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton6\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton7\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton7\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton7\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton8\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton8\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton8\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton9\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton9\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton9\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton10\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton10\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton10\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton11\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton11\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton11\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton12\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton12\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton12\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton13\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton13\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton13\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton14\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton14\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton14\_actionPerformed(e);

}

}

class Rozr\_BinCalc\_Multiplication\_jButton15\_actionAdapter implements ActionListener {

private Rozr\_BinCalc\_Multiplication adaptee;

Rozr\_BinCalc\_Multiplication\_jButton15\_actionAdapter(Rozr\_BinCalc\_Multiplication adaptee) {

this.adaptee = adaptee;

}

public void actionPerformed(ActionEvent e) {

adaptee.jButton15\_actionPerformed(e);

}

}

class Binary\_Operations

{

private double d2;

public String Double\_to\_Binary(double d\_val){

String res = "", bin\_fr = "";

double fract, res\_fr;

long whole\_part;

if (d\_val >= 0)

{

d2 = Math.floor(d\_val);

whole\_part = Math.round(d2);

}

else

{

d2 = Math.floor(d\_val)+1;

whole\_part = Math.round(d2);

}

fract = Math.abs(roundDouble(d\_val - whole\_part, 10));

res\_fr = fract;

if (res\_fr == 0d) return ""+(Long.toBinaryString(whole\_part))+".0";

while (roundDouble(res\_fr,10) != 1d)

{

res\_fr = res\_fr \* 2;

if (res\_fr > 1)

{

bin\_fr = bin\_fr + "1";

res\_fr--;

}

else

if (roundDouble(res\_fr,10) != 1d)

bin\_fr = bin\_fr + "0";

}

bin\_fr = bin\_fr + "1";

res = ""+(Long.toBinaryString(whole\_part))+"."+bin\_fr;

return res;

}

public double Binary\_to\_Double(String res\_s){

double d\_res = 0, d\_val, fract;

long whole\_part;

String temp = "";

int fr\_count = 0;

int [][] table;

d\_val = Double.parseDouble(res\_s);

if (d\_val == 0d)

return d\_val;

if (d\_val > 0)

{

d2 = Math.floor(d\_val);

whole\_part = Math.round(d2);

table = new int[2][res\_s.length()-1];

}

else

{

d2 = Math.floor(d\_val)+1;

whole\_part = Math.round(d2);

table = new int[2][res\_s.length()-2];

}

fract = Math.abs(roundDouble(d\_val - whole\_part, 10));

temp = ""+fract+"";

fr\_count = temp.length()-2;

int len = res\_s.length() - 1;

if (d\_val < 0) len = res\_s.length() - 2;

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

table[1][i] = i - fr\_count;

d\_val = Math.abs(d\_val);

res\_s = ""+d\_val+"";

int pos = res\_s.length()-1;

for (int i = 0; i < len; i++, pos--)

{

if (res\_s.charAt(pos) == '.')

pos--;

table[0][i] = Integer.parseInt(""+res\_s.charAt(pos)+"");

}

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

d\_res = d\_res + table[0][i]\*Math.pow(2.0,table[1][i]);

return d\_res;

}

public String Bin\_Multiplication(String str1, String str2)

{

String res\_str = "", res\_s = "";

String res1=str1, res2=str2;

int pos1=0, pos2=0;

int b=0;

for (int i=0; i<str1.length(); i++)

{

if (str1.charAt(i) == '.')

{

pos1 = i;

break;

}

}

for (int i=0; i<str2.length(); i++)

{

if (str2.charAt(i) == '.')

{

pos2 = i;

break;

}

}

if (pos1>pos2)

{

for (int i = 0; i < pos1 - pos2; i++)

res1 = '0' + res1;

}

else

if (pos2>pos1)

{

for (int i = 0; i < pos2 - pos1; i++)

res2 = '0' + res2;

}

if (res1.length()>res2.length())

{

for (int i = 0; i < res1.length() - res2.length(); i++)

res2 = res2 + '0';

}

else

if (res2.length()>res1.length())

{

for (int i = 0; i < res2.length() - res1.length(); i++)

res1 = res1 + '0';

}

for (int i=0; i<res1.length(); i++)

{

if (res1.charAt(i) == '.')

{

pos1 = i;

break;

}

}

str1 = "";

pos1 = Math.abs(res1.length()-pos1);

str2 = "";

for (int i=0; i<res1.length(); i++)

{

if (res1.charAt(i) != '.') str1 = str1+res1.charAt(i);

if (res2.charAt(i) != '.') str2 = str2+res2.charAt(i);

}

res\_s = " ";

pos2 = 0;

for (int i=0; i<res1.length(); i++)

{

if (res2.charAt(i) == '1')

{

for (int j=0; j<res1.length(); j++)

{

res\_s.replace(res\_s.charAt(j+pos2),res1.charAt(j));

}

pos2++;

}

}

str1 = "0";

str1 = str1 +'.';

for (int i=pos1; i<res\_s.length(); i++)

{

str1 = str1 + res\_s.charAt(i);

}

return str1;

}

public double roundDouble(double val, int places) {

long factor = (long)Math.pow(10,places);

val = val \* factor;

long tmp = Math.round(val);

return (double)tmp / factor;

}

}

}

Висновок

В моїй розрахунковій роботі було представлено приклад калькулятора, що перемножує числа. Було представлено побудову діаграм UML, розглянуто деякі типи діаграм та їх значення, структуру, на зворот ньому процесі до проектування було продемонстровано переваги попереднього проектування розроблюваного даного програмного продукту.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Программирование на Java 2 (2-е издание) ФОЛИО 2003
2. 11-12 лекції.