МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тверской государственный технический университет»

(ТвГТУ)

Кафедра “Программного обеспечения”

**Лабораторная работа №2**

по дисциплине “Конструирование ПО”

Тема: «Разработка UML модели проекта “Численное интегрирование” »

Выполнила: студентка группы

ПИН-17.06

Антонян А.С

Проверил:

Мальков А.А

Тверь 2020

# **Введение**

**Цель работы**: получить навык создания на основе UML модели в системе Visual Studio 2013 и изучить особенности создания на основе UML кода на языке C# .

# **Ответы на контрольные вопросы**

1. Какие этапы жизненного цикла разработки программного проекта поддерживает среда Visual Studio 2013.

* проектирование
* реализация (кодирование)
* верификация (тестирование)

1. Какие UML-диаграммы можно построить с использованием Visual Studio 2013.

* UML Sequence Diagram - диаграмма последовательности;
* UML Use Case Diagram - диаграмма использования;
* UML Activity Diagram - диаграмма активности;
* UML Component Diagram - диаграмма компонент;
* Layer Diagram - диаграмма уровней;

1. Каковы возможности автоматической генерации кода в Visual Studio 2013.

* Выбор шаблона для генерации кода (класс, перечисление, интерфейс, структура)

1. Рефакторинг в Visual Studio 2013. Его возможности и ограничения.

Возможности:

* Rename - переименование имени переменной, метода, класса и т.п. с автоматическим обновлением всех ссылок на это имя в коде;
* Extract method - оформление выделенной части кода в новый, отдельный метод;
* Encapsulate field - создание свойства, скрывающего выбранную переменную-член класса;
* Extract interface - создание интерфейса на основе списка методов класса;
* Promote local variable to parameter - вынесение локальной переменной в параметр метода;
* Remove/Reorder parameters - удаление параметров метода и изменение порядка их следования с автоматическим обновлением всех ссылок в коде на данный метод.

Diagram Documentation

**Функциональная модель** - *(Analysis diagram)*



*Figure: 1*

**Модель требований** - *(Requirements diagram)*



*Figure: 2*

**0. Бизнес правила** - *(Requirements diagram)*



*Figure: 3*

**1. Общие требование** - *(Requirements diagram)*



*Figure: 4*

**Диаграмма прецедентов** - *(Use Case diagram)*



*Figure: 5*

**Деятельности** - *(Activity diagram)*



*Figure: 6*

**Последовательности** - *(Sequence diagram)*



*Figure: 7*

**Диаграмма классов** - *(Logical diagram)*



*Figure: 8*

## Пример доработанного сгенерированого кода

import time

import sys

import math

import numpy as np

from PyQt5.QtWidgets import (QWidget, QToolTip, QLineEdit,QTextEdit, QComboBox, QMainWindow, QPushButton, QApplication,QLabel)

from PyQt5.QtGui import QFont

from functools import partial

from scipy import integrate

import numba

import os

from joblib import Parallel, delayed

class Integral(object):

'Класс Integral предоставляет различные методы численного интегрирования'

# Конструктор

def \_\_init\_\_(self, f, a, b, N):

self.f = f

self.a = a

self.b = b

self.N = N

pass

# Метод Ньютона-Котеса

def newtonCotes(self):

degree = 3

w = np.array([[1,0,0,0,0,0,0,0,0,0],

[1,1,0,0,0,0,0,0,0,0],

[1,4,1,0,0,0,0,0,0,0],

[1,3,3,1,0,0,0,0,0,0],

[7,32,12,32,7,0,0,0,0,0],

[19,75,50,50,75,19,0,0,0,0],

[41,216,27,272,27,216,41,0,0,0],

[751,3577,1323,2989,2989,1323,3577,751,0,0],

[989,5888,-928,10496,-4540,10496,-928,5888,989,0],

[2857,15741,1080,19344,5778,5778,19344,1080,15741,2857]])

c = np.array([1, 1.0/2, 1.0/3, 3.0/8, 2.0/45, 5.0/288, 1.0/140, 7.0/17280, 4.0/14175, 9.0/89600])

h = (self.b - self.a) / (degree \* self.N)

Sum = 0

for k in range(self.N):

partSum = 0

for i in numba.prange(degree + 1):

partSum += w[degree][i] \* self.f(self.a + (i + k \* degree) \* h)

Sum += c[degree] \* partSum \* h

return Sum

# Метод трапеций

def trapeze(self):

i = 0

S = 0

h = (self.b - self.a) / self.N

for i in numba.prange(self.N):

x1 = self.a + h \* i

x2 = self.a + h \* (i + 1)

S += (self.f(x1) + self.f(x2)) \* h / 2

return S

# Метод Симпсона

def simpson(self):

i = 0

S = 0

h = (self.b - self.a) / self.N

for i in numba.prange(self.N):

x1 = self.a + h \* i

x2 = x1 + h

x3 = (x1 + x2)/2

S += (self.f(x1) + 4 \* self.f(x3) + self.f(x2)) \* h / 6

return S

# Метод Гаусса(n = 3)

def gauss(self):

def g(c, d):

Xi = np.array([-0.7745967, 0, 0.7745967])

Ci = np.array([0.5555556, 0.8888889, 0.5555556])

n = 3

ra = (d - c) / 2

su = (c + d) / 2

S = 0

for i in range(n):

Q = su + ra \* Xi[i]

S += Ci[i] \* self.f(Q)

return ra \* S

s = 0

list = Parallel(n\_jobs=1)(delayed(g)(self.a + i \* (self.b - self.a)/ self.N, self.a + (i + 1) \* (self.b - self.a)/ self.N) for i in range(self.N))

s = sum(list)

return s

# Метод Чебышева(n = 5)

def chebyshev(self):

x = np.array([0.083751, 0.312730, 0.500000, 0.687270, 0.916249])

aa = self.a

h = (self.b - self.a) / self.N

S4 = 0

S = 0

for i in numba.prange(self.N):

xx = aa + h

for j in range(5):

S += self.f(xx + h \* x[j])

S4 += S \* h / 5

S = 0

aa = xx

return S4

class Form(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.initUI()

def initUI(self):

QToolTip.setFont(QFont('SansSerif', 12))

self.setGeometry(300, 300, 650, 450)

self.setWindowTitle('Численное интегрирование')

self.s = 0

self.e = 0

self.n = 0

lbl1 = QLabel('a = ', self)

lbl1.move(350, 40)

lbl2 = QLabel('b = ', self)

lbl2.move(500, 40)

lbl3 = QLabel('f(x) = ', self)

lbl3.move(15, 45)

lbl4 = QLabel('n = ', self)

lbl4.move(350, 75)

lbl5 = QLabel('scipy = ', self)

lbl5.move(35, 275)

lbl6 = QLabel('Результат =', self)

lbl6.move(8, 333)

self.Scipy = QLabel(self)

self.Scipy.move(73, 261)

self.Scipy.resize(255, 40)

self.Scipy.setStyleSheet("background-color: white")

lbl10 = QLabel('Время = ', self)

lbl10.move(355, 333)

self.Time = QLabel(self)

self.Time.move(400, 320)

self.Time.resize(240, 40)

self.Time.setStyleSheet("background-color: white")

self.A = QLineEdit(self)

self.A.move(370, 40)

self.A.resize(80, 20)

self.A.setText('0')

self.B = QLineEdit(self)

self.B.move(520, 40)

self.B.resize(80, 20)

self.B.setText('1')

self.N = QLineEdit(self)

self.N.move(370, 75)

self.N.resize(80, 20)

self.N.setText('10')

self.Func = QTextEdit(self)

self.Func.move(50, 40)

self.Func.resize(250, 50)

self.Func.setText('4 / (1 + pow(x, 2))')

self.Result = QLabel(self)

self.Result.move(75, 320)

self.Result.resize(255,40)

self.Result.setStyleSheet("background-color: white")

btnN = QPushButton('Метод Ньютона-Котеса', self)

btnN.resize(140, 40)

btnN.move(30, 130)

btnN.clicked.connect(partial(self.calculate, 'newtonCotes'))

btnT = QPushButton('Метод трапеций', self)

btnT.resize(140, 40)

btnT.move(200, 130)

btnT.clicked.connect(partial(self.calculate, 'trapeze'))

btnS = QPushButton('Метод Симпсона', self)

btnS.resize(140, 40)

btnS.move(370, 130)

btnS.clicked.connect(partial(self.calculate, 'simpson'))

btnG = QPushButton('Метод Гаусса', self)

btnG.resize(140, 40)

btnG.move(30, 200)

btnG.clicked.connect(partial(self.calculate, 'gauss'))

btnCh = QPushButton('Метод Чебышева', self)

btnCh.resize(140, 40)

btnCh.move(200, 200)

btnCh.clicked.connect(partial(self.calculate, 'chebyshev'))

btnSav = QPushButton('Сохранить в файл', self)

btnSav.resize(140, 40)

btnSav.move(65,380)

btnSav.clicked.connect(self.save)

self.show()

def init(self):

self.s = (float)(self.A.text())

self.e = (float)(self.B.text())

self.n = (int)(self.N.text())

def calculate(self, method):

self.init()

integral = Integral(self.func, self.s, self.e, self.n)

dc = {'newtonCotes': integral.newtonCotes,

'trapeze': integral.trapeze ,

'simpson': integral.simpson,

'gauss': integral.gauss,

'chebyshev': integral.chebyshev}

start = time.time()

res = dc[method]()

end = time.time()

self.Result.setText(str(res))

self.Scipy.setText(str(integrate.quad(self.func, self.s, self.e)))

self.Time.setText(str(end - start))

def save(self):

file = open('res.txt', 'w')

text = 'Результат: '+ self.Result.text() + '\nТочность'

file.write(text)

file.close()

def func(self, x):

str = self.Func.toPlainText()

d = {'x' : x }

result = eval(str, d)

return result

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = Form()

sys.exit(app.exec\_())