МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи № 4

**ОБЧИСЛЕННЯ ОБ’ЄКТНО-ОРІЄНТОВАНИХ МЕТРИК**

з дисципліни

«Якість програмного забезпечення та тестування»

Виконав:

студент групи КНТ-136сп В.В. Гарковенко

Перевірила:

доцент, к.т.н. Т.І. Каплієнко

2018

**Мета роботи:** Вивчити основні об’єктно-орієнтовані метрики програмних систем та реалізувати їх із застосуванням мови С++ та Qt.

**Завдання до роботи:**

1. Ознайомитися з теоретичними відомостями щодо об’єктно-орієнтованих метрик програмних систем.
2. Реалізувати на мові C++ програму розрахунку об’єктно-орієнтованих метрик.
3. Виконати аналіз отриманих результатів.
4. Оформити звіт та відповісти на контрольні питання.

**Виконання:**

Вихідний код:

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QMessageBox>

#include <QTextStream>

#include <QDebug>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~MainWindow()

{

delete ui;

}

/\*

\* Global variables

\*

\* code - save text (code) from file

\*/

QString code;

float MainWindow::averNumMethLines(QString code)

{

QStringList conditions = code.split(QRegExp("(void|int|string|float|double|byte)[A-Za-z0-9]\*"), QString::SkipEmptyParts);

int s = 0;

for(int i = conditions.count()-1; i>=0; --i)

{

const QString &item = conditions[i];

s=s+item.count("\n");

}

return (float)s/(conditions.count()) - 1;

}

float MainWindow::averNumModuleLines(QString code)

{

QStringList includes = code.split(QRegExp("#include"), QString::SkipEmptyParts);

return (float) includes.count() - 1;

}

float MainWindow::averNumClassLines(QString code)

{

QStringList classes = code.split(QRegExp("class "), QString::SkipEmptyParts);

int s = 0;

for(int i = classes.count()-1; i>=0;--i)

{

const QString &item = classes[i];

s=s+item.count("\n")-1;

}

return (float) s/classes.count();

}

int MainWindow::NORM(QString code)

{

QStringList classes = code.split(QRegExp("class "), QString::SkipEmptyParts);

int s = 0;

for(int i = classes.count()-1; i>=0; --i)

{

const QString &item = classes[i];

QStringList list = item.split("){",QString::SkipEmptyParts);

s = s + item.count(")") - item.count("this->") - item.count(" )");

}

return s;

}

int MainWindow::RFC(QString code)

{

QStringList classes = code.split(QRegExp("class "), QString::SkipEmptyParts);

int s = 0;

for(int i = classes.count()-1; i>=0; --i)

{

const QString &item = classes[i];

QStringList list = item.split("){",QString::SkipEmptyParts);

s = s + item.count(")") - item.count(" )");

}

return s;

}

int MainWindow::WMC(QString code)

{

QStringList classes = code.split(QRegExp("class "), QString::SkipEmptyParts);

int s = 0;

for(int i = classes.count()-1; i>=0; --i)

{

const QString &item = classes[i];

QStringList list = item.split("){",QString::SkipEmptyParts);

s = 0;

for(int j = list.count()-1;j>=0; --j)

{

s = s + 1;

}

return s;

}

return s;

}

void MainWindow::NOC(QString code)

{

QStringList classes = code.split(QRegExp("class"), QString::SkipEmptyParts);

QStringList list,Lls, Rls;

for(int i=0; i<classes.count(); i++)

{

const QString &item = classes[i].trimmed();

QString str = "";

for(int j=0; j<item.count();j++)

{

if(item[i] != '\n')

str=str+QString(item[i]);

else

break;

}

list<<str;

}

for(int i=0; i<list.count() ;i++)

{

const QString &item = classes[i].trimmed();

QString str = "";

for(int j=0; j<item.count();j++)

{

if(item[j] != ' ')

str=str+QString(item[j]);

else

break;

}

Rls << str;

}

for(int i=0; i<list.count(); i++)

{

const QString &item = list[i].trimmed();

QString str = " ";

classes = item.split(QRegExp("public|protected|private"), QString::SkipEmptyParts);

for(int j=0; j < classes.count(); j++)

{

str = str + classes[j];

}

Lls << str;

}

for(int i = 0; i<Rls.count(); i++)

{

Lls[i].replace(0, Rls[i].count(),"");

Lls[i] = Lls[i].trimmed();

}

for(int i = 0; i<Rls.count(); i++)

{

Rls[i].remove(":");

int s = 0;

for(int j = 0; j < Rls.count(); j++)

{

s = s + Lls[j].count(Rls[i]);

}

ui->label\_NOC->setText(ui->label\_NOC->text()+" "+ Rls[i]+" " + QString::number(s)+"\n");

}

}

void MainWindow::on\_openFile\_Button\_clicked()

{

QString fileName = QFileDialog::getOpenFileName(this, tr("Open file"),"",tr("C++ files (\*.cpp \*.h)"));

if(fileName!="")

{

QFile file(fileName);

if(!file.open(QIODevice::ReadOnly))

{

QMessageBox::critical(this, tr("Error"), tr("Could not open file!"));

return;

}

QTextStream input(&file);

ui->plainTextEdit->setPlainText(input.readAll());

file.close();

code = ui->plainTextEdit->toPlainText();

}

}

void MainWindow::on\_calculate\_Button\_clicked()

{

ui->label\_averNumMethLines->setText("Average number of method lines: " + QString::number(averNumMethLines(code)));

ui->label\_averNumModuleLines->setText("Average number of module lines: " + QString::number(averNumModuleLines(code)));

ui->label\_averNumClassLines->setText("Average number of class lines: " + QString::number(averNumClassLines(code)));

ui->label\_NORM->setText("Number Of Remote Methods: " + QString::number(NORM(code)));

ui->label\_RFC->setText("Response For Class: " + QString::number(RFC(code)));

ui->label\_WMC->setText("Class method complexity: " + QString::number(WMC(code)));

MainWindow::NOC(code);

}

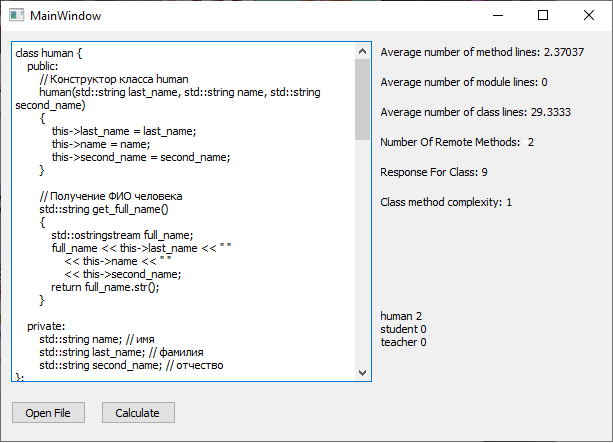


Рисунок 4.1 – Головне вікно програми

**Висновок:** При виконанні практичної роботи вивчив основні об’єктно-орієнтовані метрики програмних систем та реалізувати їх із застосуванням мови С++ та Qt.