|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по домашней работе №** | 3 |

**Название:**

Программирование на С++ с использованием классов

**Дисциплина:** Объектно-ориентированное программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22Б |  |  | Баканов Роман |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  |  |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание:**

Разработать и реализовать диаграмму классов для описанных объектов предметной области, используя механизм композиции. Протестировать все методы каждого класса. Все поля классов должны быть скрытыми (private) или защищенными (protected). Методы не должны содержать операций ввода/вывода, за исключением процедуры, единственной задачей которой является вывод информации об объекте на экран.

Объект – экзамен. Параметры: название предмета и оценка. Методы: процедура инициализации, процедура вывода на экран значений параметров, процедура пересдачи экзамена на заданную оценку и функции, возвращающие по запросу значения параметров.

Объект – зачетная книжка. Параметры: номер книжки, фамилия студента, количество экзаменов и массив экзаменов. Методы должны позволять инициализировать объект, выводить информацию об объекте на экран, пересдавать экзамен по заданному предмету, определять среднюю оценку, выявлять возможность получить красный диплом.

**Текст программы:**

(Composition.h)

#include <string>

using namespace std;

class exam

{

protected:

string name;

int mark;

public:

void print();

void init(string args, int argi);

void retake(int argi);

string getname();

int getmark();

};

class book

{

protected:

int number;

string surname;

int amount;

exam exams[5];

public:

void massinit(int argi3, string args2, int argi1, string args1[], int argi2[]);

void massprint();

void cretake(string args, int argi);

double average();

string red();

};

(Composition.cpp)

#include <iostream>

#include "Composition.h"

using namespace std;

void exam::retake(int argi)

{

mark = argi;

}

void exam::init(string args, int argi)

{

name = args;

mark = argi;

}

void exam::print()

{

cout << "Subject and mark: " << name << " - " << mark << endl;

}

int exam::getmark()

{

return mark;

}

string exam::getname()

{

return name;

}

void book::massinit(int argi3, string args2, int argi1, string args1[], int argi2[])

{

number = argi3;

surname = args2;

amount = argi1;

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

exams[i].init(args1[i], argi2[i]);

}

void book::massprint()

{

cout << "Surname and number of student book: " << surname << " " << number << endl;

cout << "Amount of subject: " << amount << endl;

cout << "Subjects and marks:" << endl;

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

cout << exams[i].getname() << " - " << exams[i].getmark() << endl;

}

void book::cretake(string args, int argi)

{

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

if (exams[i].getname() == args)

exams[i].retake(argi);

}

double book::average()

{

double rez = 0;

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

rez += exams[i].getmark();

rez /= amount;

return rez;

}

string book::red()

{

double fives = 0, threes = 0;

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

{

if (exams[i].getmark() == 5)

fives++;

if (exams[i].getmark() == 3)

threes++;

}

if ((threes > 0) or (fives / amount < 0.75))

return "No";

else

return "Yes";

}

int main()

{

book me;

cout << "Enter your surname and number of record book" << endl;

string gets1, gets2[10];

int geti1, geti2[10], geti3;

cin >> gets1 >> geti1;

cout << "Enter amount of exams you have had" << endl;

cin >> geti3;

cout << "Enter subject name and your mark per it" << endl;

for (int i = 0; i < geti3; i++)

cin >> gets2[i] >> geti2[i];

me.massinit(geti1, gets1, geti3, gets2, geti2);

int code = 0;

while (code != 5)

{

cout << "Choose one of the following actions:" << endl;

cout << " 1 - show info" << endl;

cout << " 2 - retake exam" << endl;

cout << " 3 - show average mark" << endl;

cout << " 4 - is it possible to get a red diploma?" << endl;

cout << " 5 - exit" << endl;

cin >> code;

switch (code)

{

case 1:

me.massprint();

break;

case 2:

cout << "Enter subject name and new mark" << endl;

cin >> gets1 >> geti1;

me.cretake(gets1, geti1);

cout << "Mark was changed" << endl;

break;

case 3:

cout << "Average mark: " << me.average() << endl;

break;

case 4:

cout << me.red() << endl;

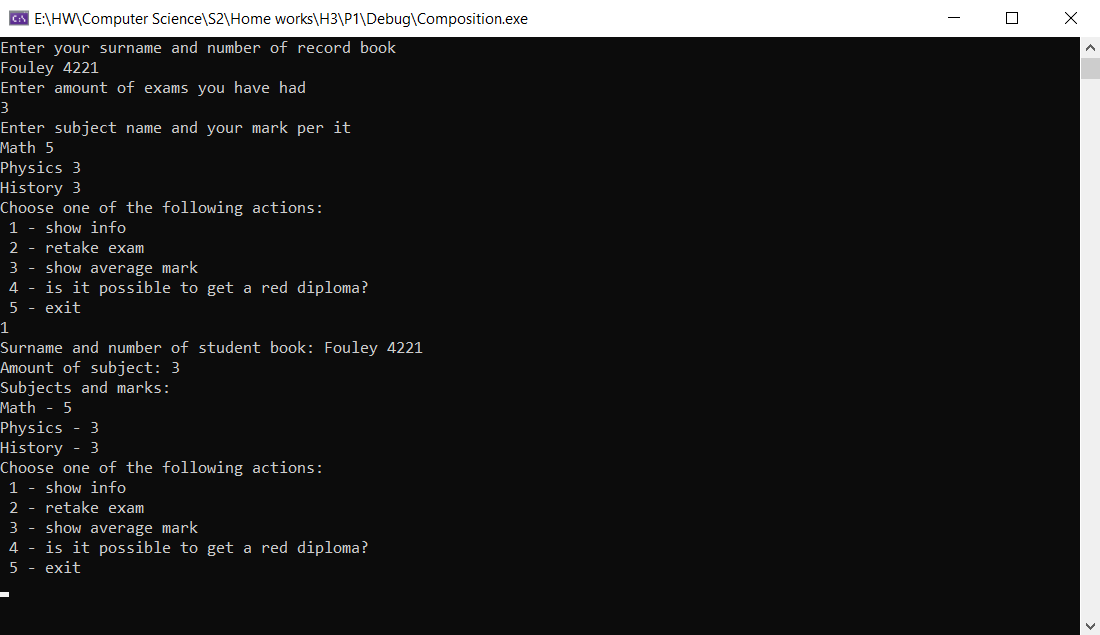
break;

}

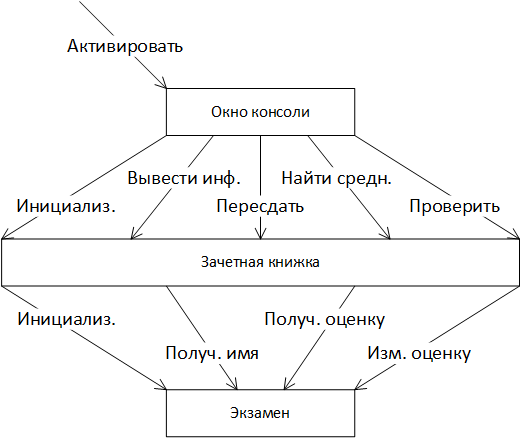
}

}

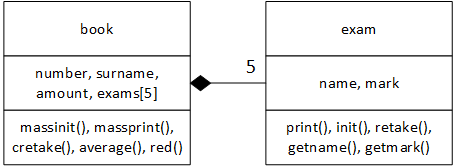
**Скриншоты:**

****

**Объектная декомпозиция:**

****

**Диаграмма классов:**

****

**Задание:**

Разработать программу, содержащую описание трех графических объектов: круг с вырезанной четвертью, спираль Архимеда, квадрат.

Реализуя механизм полиморфизма, привести объекты в одновременное вращение вокруг их геометрических центров с различными угловыми скоростями.

В отчете привести диаграмму используемых классов Qt и разработанных классов, граф состояний пользовательского интерфейса и объектную декомпозицию.

**Текст программы:**

(draw.h)

#ifndef DRAW\_H

#define DRAW\_H

#include <QDialog>

#include <ui\_draw.h>

#include <QPainter>

#include <QTimer>

class Form:public QDialog, public Ui::Draw

{

Q\_OBJECT

public:

Form(QWidget\* parent = 0);

protected:

QTimer \*timer;

QPainter painter;

private slots:

void *paintEvent*(QPaintEvent \*event);

};

#endif

(figures.h)

#ifndef FIGURES\_H

#define FIGURES\_H

#define \_USE\_MATH\_DEFINES

#include <math.h>

#include <QPainter>

class figure

{

public:

void drawing(QPainter \*canvas, double n);

};

class archi:public figure

{

public:

void drawing(QPainter \*canvas, double n);

};

class square:public figure

{

public:

void drawing(QPainter \*canvas, double n);

};

class circle:public figure

{

public:

void drawing(QPainter \*canvas, double n);

};

#endif

(draw.cpp)

#include "draw.h"

#include "figures.h"

archi fig;

square fig1;

circle fig2;

double n[3] = {0, 0, 0};

Form::Form(QWidget\* parent):QDialog(parent)

{

setupUi(this);

setFixedSize(1800, 600);

timer = new QTimer(this);

connect(timer,SIGNAL(timeout()),this,SLOT(repaint()));

timer->start(50);

}

void Form::*paintEvent*(QPaintEvent \*event)

{

Q\_UNUSED(event)

for (int i = 0; i < 3; i++)

{

n[i] += 0.05 \* pow(2,i);

if (n[i] >= 2\*M\_PI)

n[i] = 0;

}

QPainter painter(this);

fig.drawing(&painter, n[1]);

fig1.drawing(&painter, n[2]);

fig2.drawing(&painter, n[0]);

}

(figures.cpp)

#include "figures.h"

void figure::drawing(QPainter \*canvas, double n)

{

//abstract

}

void square::drawing(QPainter \*canvas, double n)

{

int b[4][2];

for (int i = 0; i < 4; i++)

{

b[i][0] = 900+round(180\*(-1)\*cos(n+i\*M\_PI/2));

b[i][1] = 300+round(180\*sin(n+i\*M\_PI/2));

}

for (int i = 0; i < 4; i++)

canvas->drawLine(b[i][0], b[i][1], b[(i+1)%4][0], b[(i+1)%4][1]);

}

void circle::drawing(QPainter \*canvas, double n)

{

int b;

b = -(n\*180\*16/M\_PI);

canvas->drawPie(1350, 150, 300, 300, b, 16\*270);

}

void archi::drawing(QPainter \*canvas, double n)

{

int b[6][3];

b[0][0] = 300;

b[0][1] = 300;

b[4][0] = 300;

b[4][1] = 300;

for (int i = 1; i < 4; i++)

{

if (i == 2)

{

b[i][0] = 300+round(sqrt(1800)\*cos(n-((i+1)-2)\*M\_PI/4));

b[i][1] = 300+round(sqrt(1800)\*sin(n-((i+1)-2)\*M\_PI/4));

}

else

{

b[i][0] = 300 + round(30\*cos(n-((i+1)-2)\*M\_PI/4));

b[i][1] = 300 + round(30\*sin(n-((i+1)-2)\*M\_PI/4));

}

}

b[5][0] = b[1][0];

b[5][1] = b[1][1];

for (int i = 0; i < 6; i++)

b[i][2] = 16\*90\*((i+1)%4)-(n\*180\*16/M\_PI);

for(int i = 0; i < 6; i++)

canvas->drawArc(b[i][0]-30\*(i+1), b[i][1]-30\*(i+1), 60\*(i+1), 60\*(i+1), b[i][2], 16\*90);

}

(start.cpp)

#include <QApplication>

#include "draw.h"

int main(int argc, char\* argv[])

{

QApplication app(*argc*, argv);

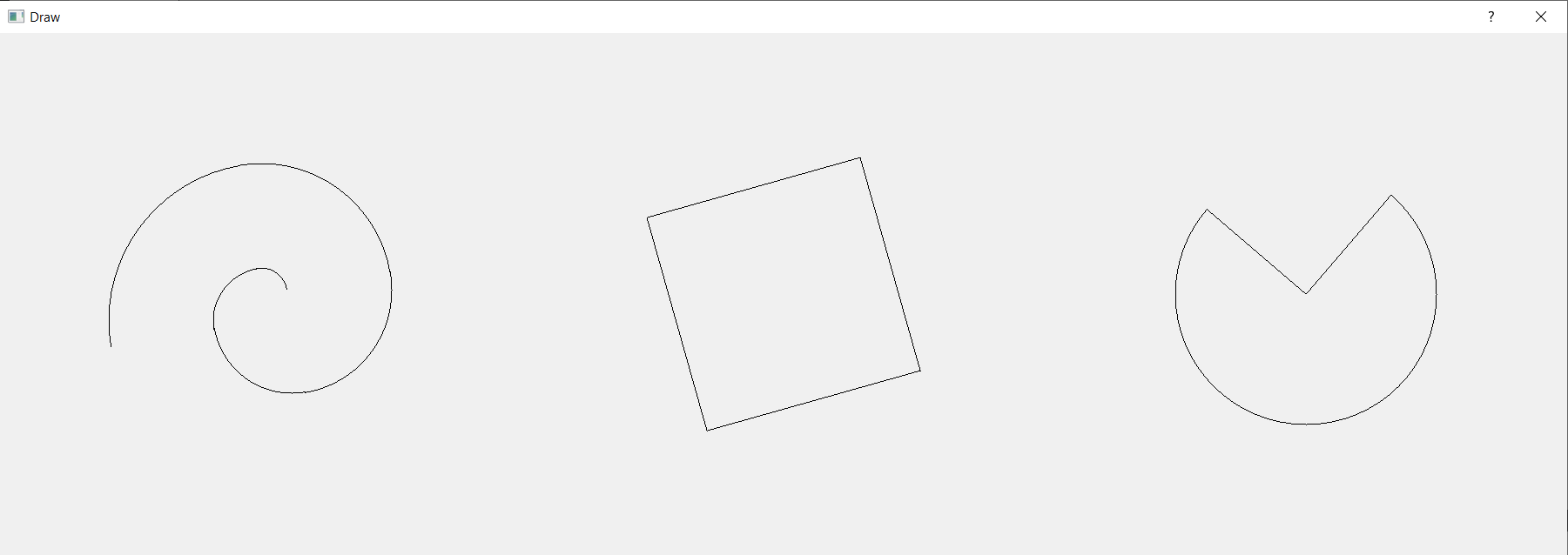
Form\* dialog = new Form();

dialog->show();

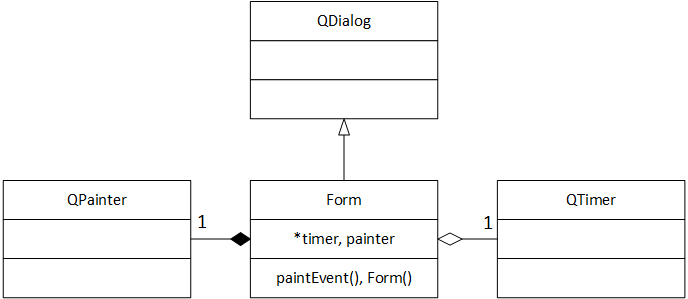
return app.exec();

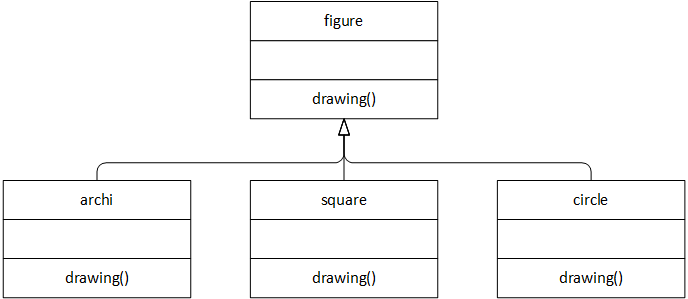
}

**Скриншоты:**

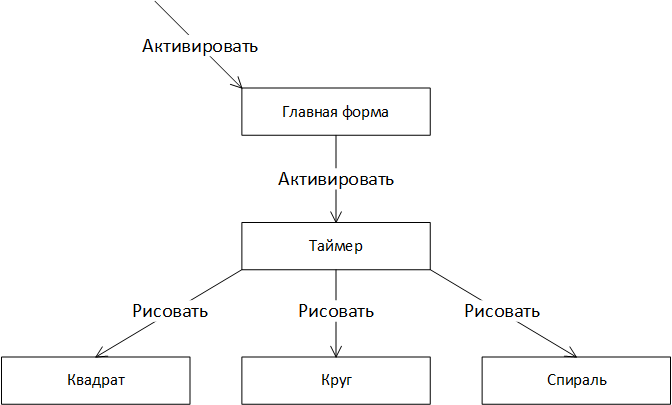
****

**Диаграмма классов:**

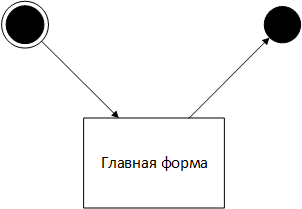
****

****

**Объектная декомпозиция:**

****

**Диаграмма состояний интерфейса:**

****

**Выводы:**

Получены навыки применения на языке С++ таких методов построения классов, как композиция и полиморфное наследование. Также изучены инструменты, которые Qt предлагает для работы с графикой.