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**

О Т Ч Е Т (В А Р И А Н Т 1 2)

по домашней работе № 3

Название: **Программирование на C++ с использованием классов**

Дисциплина: Объектно-ориентированное программирование

Студент ИУ6-23Б (Группа) 14.05.2023 (Подпись, дата) В.К. Залыгин (И.О. Фамилия)

Преподаватель (Подпись, дата) А.М.Минитаева (И.О. Фамилия)

Москва, 2023

Цель работы

Изучить основы работы с объектной моделью C++ и библиотеки графических интерфейсов QT.

Задание

Часть 3.1. Композиция

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 – продукт. Поля: название, дата изготовления, срок годности в днях. Методы: процедура инициализации, процедура вывода информации об объекте на экран, функции, возвращающие значения полей по запросу, и функция вычисления даты, до которой годен продукт.

Проект программы

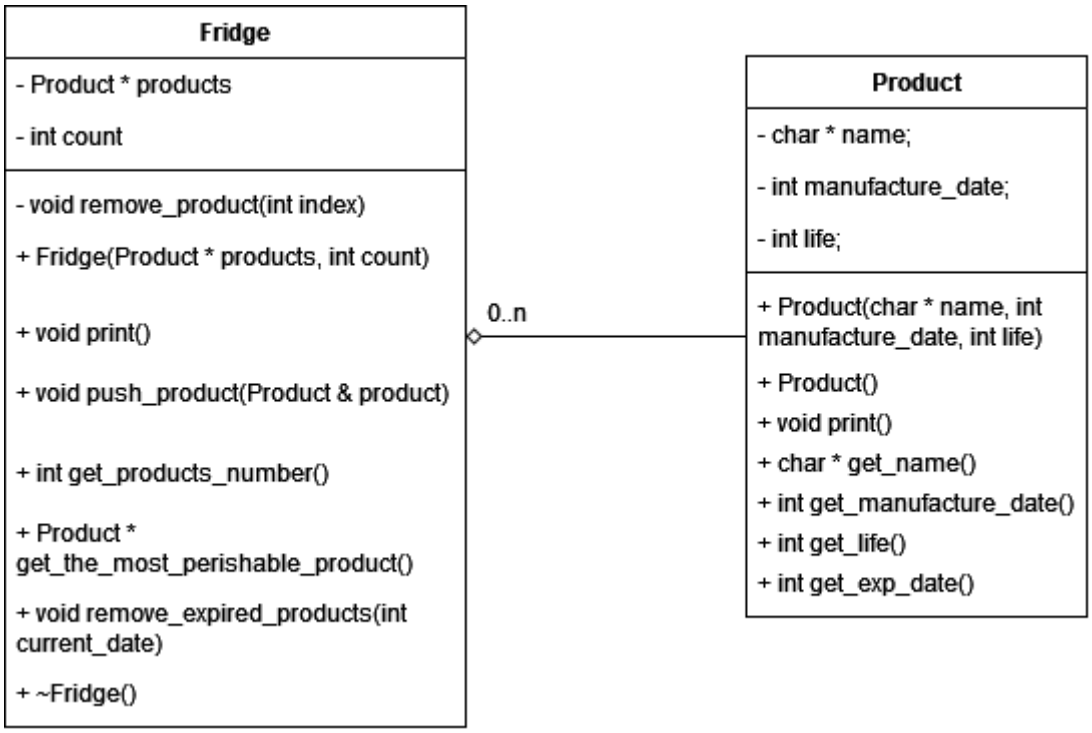


Рисунок 1 – диаграмма классов

Текст программы

```cpp
#include <iostream>
#include <cstring>
#include <stdlib.h>

using namespace std;

class Product {
private:
    char * name;
    int manufacture_date;
    int life;
public:
    Product(char * name, int manufacture_date, int life)
        : name(name), manufacture_date(manufacture_date), life(life) {}
    Product() {}

    void print() {
        cout << "Product; name: " << name << "; manufacture_date: " << manufacture_date << "; exp_date: " << life << ";";
    }

    char * get_name() { return name; }
    int get_manufacture_date() { return manufacture_date; }
    int get_life() { return life;}

    int get_exp_date() { return manufacture_date + life; }
};
```

Рисунок 2 - код программы. Класс Product

```cpp
class Fridge {
private:
    Product * products;
    int count;

    void remove_product(int index) {
        Product * new_products = new Product[count-1];
        int j = 0;
        for (int i = 0; i < index; ++i, ++j)
            new_products[j] = products[i];
        for (int i = index+1; i < count; ++i, ++j)
            new_products[j] = products[i];
        free(products);
        products = new_products;
        count--;
    }
public:
    Fridge(Product * products = nullptr, int count = 0)
        : products(products), count(count) {}

    void print() {
        cout << "Fridge; number of the products: " << count << "; Products: " << endl;
        for (int i = 0; i < count; ++i) {
            products[i].print();
            cout << endl;
        }
        cout << ";" << endl;
    }

    void push_product(Product & product) {
        products = static_cast<Product *>(realloc(products, sizeof(Product)*(count+1)));
        products[count] = product;
        count++;
    }

    int get_products_number() { return count; }

    Product * get_the_most_perishable_product() {
        if (count != 0) {
            Product * res = &products[0];
            for (int i = 1; i < count; ++i)
                if (res->get_exp_date() > products[i].get_exp_date())
                    res = &products[i];

            return res;
        } else {
            return nullptr;
        }
    }

    void remove_expired_products(int current_date) {
        for (int i = 0; i < count; ++i) {
            if (products[i].get_exp_date() < current_date)
                remove_product(i);
        }
    }

    ~Fridge() {
        free(products);
    }
};
```

Рисунок 3 - код программы. Класс Fridge

```cpp
int main()
{
    Product p1("Milk", 1, 7), p2("Egg", 1, 14), p3("Potato", 1, 60);
    Fridge fridge(nullptr, 0);
    fridge.push_product(p2);
    fridge.push_product(p1);
    fridge.push_product(p3);

    fridge.print();

    cout << "a number of the products is " << fridge.get_products_number() << endl;
    cout << "the most perishable product is";
    fridge.get_the_most_perishable_product()->print();
    cout << endl;

    fridge.remove_expired_products(10);
    cout << "the list of non-expired products(day 10): " << endl;
    fridge.print();
    cout << endl;

    return 0;
}
```

Рисунок 4 - код программы. Функция Main

Тестовые данные

```
Fridge; number of the products: 3; Products:
Product; name: Egg; manufacture_date: 1; exp_date: 14;
Product; name: Milk; manufacture_date: 1; exp_date: 7;
Product; name: Potato; manufacture_date: 1; exp_date: 60;
;
a number of the products is 3
the most perishable product isProduct; name: Milk; manufacture_date: 1; exp_date: 7;
the list of non-expired products(day 10):
Fridge; number of the products: 2; Products:
Product; name: Egg; manufacture_date: 1; exp_date: 14;
Product; name: Potato; manufacture_date: 1; exp_date: 60;
;
```

Рисунок 5 - результат работы

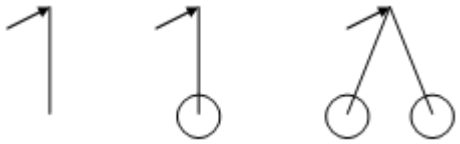
Вывод

Была разработана объектная декомпозиция задачи, реализована программа на языке C++ с использование базовых возможностей объектной модели.

Часть 3.2. Qt. Полиморфное наследование

Задание

Разработать программу, содержащую описание трех графических объектов:



Реализуя механизм полиморфизма, привести объекты в одновременное колебательное движение вокруг указанных точек с разными амплитудами и периодами колебаний.

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

Проект программы

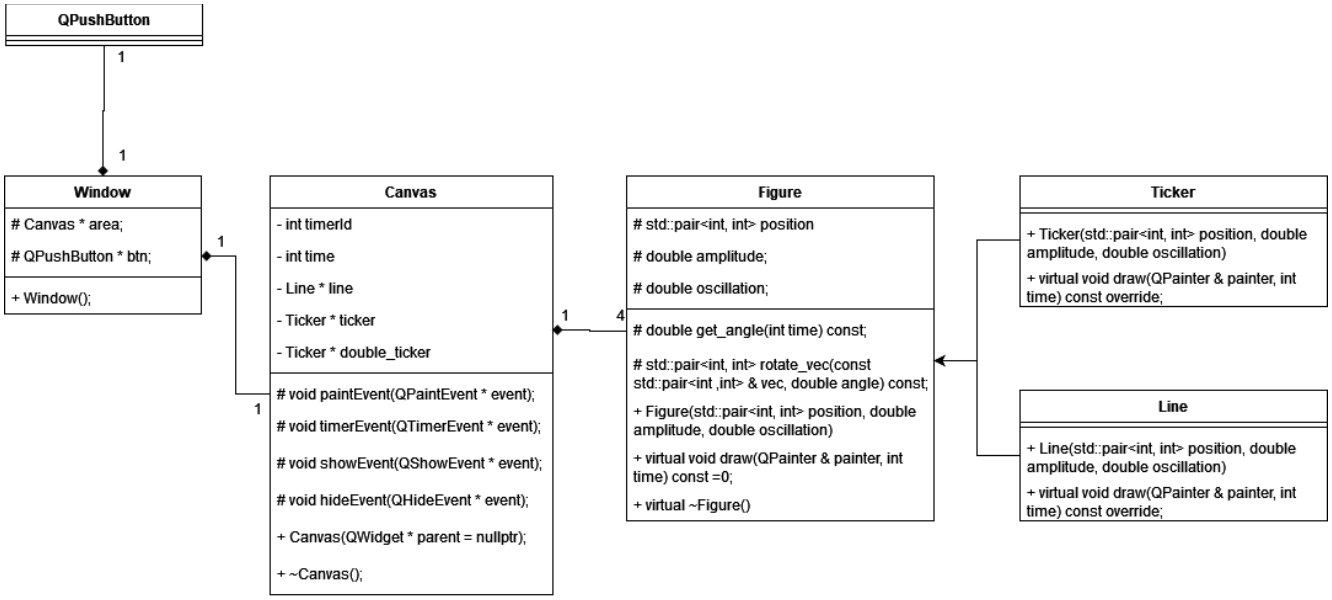


Рисунок 6 - диаграмма классов

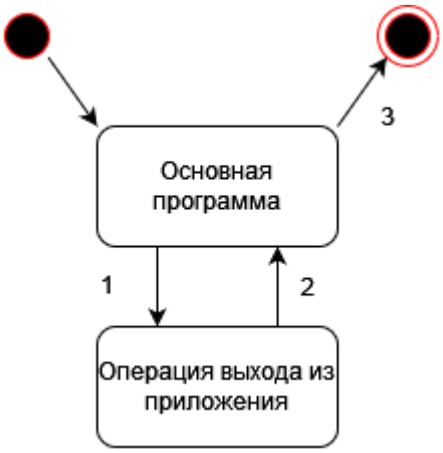


Рисунок 7 - граф состояний

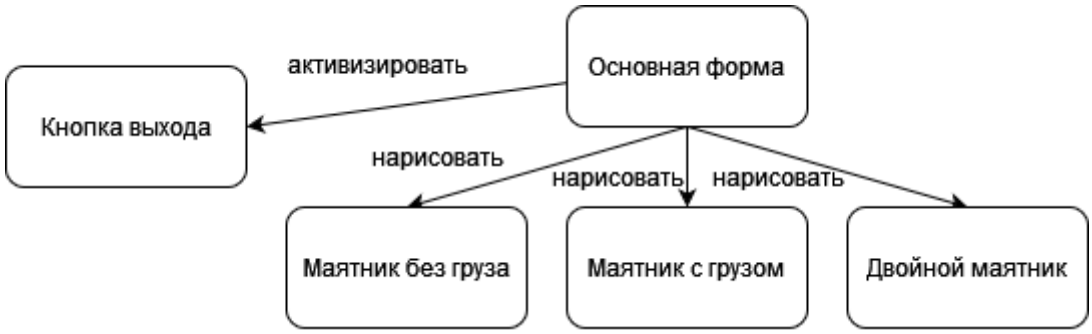


Рисунок 8 - объектная декомпозиция

Текст программы

```
#ifndef CANVAS_H
#define CANVAS_H

#include <QWidget>
#include <vector>

#include <figure.h>
#include <line.h>
#include <ticker.h>

class Canvas : public QWidget {
private:
    int timerId;
    int time;
    Line * line;
    Ticker * ticker;
    Ticker * double_ticker;
protected:
    void paintEvent(QPaintEvent * event);
    void timerEvent(QTimerEvent * event);
    void showEvent(QShowEvent * event);
    void hideEvent(QHideEvent * event);
public:
    Canvas(QWidget * parent = nullptr);
    ~Canvas() {
        delete line;
        delete ticker;
        delete[] double_ticker;
    };
};

#endif // CANVAS_H
```

Рисунок 9 - код программы

```
#ifndef FIGURE_H
#define FIGURE_H

#include <QPainter>
#include <utility>

class Figure {
protected:
    std::pair<int, int> position;
    double amplitude;
    double oscillation;

    double get_angle(int time) const;
    std::pair<int, int> rotate_vec(const std::pair<int ,int> & vec, double angle) const;
public:
    Figure(std::pair<int, int> position, double amplitude, double oscillation)
        : position(position), amplitude(amplitude), oscillation(oscillation) {};

    virtual void draw(QPainter & painter, int time) const =0; // ога, =0 - отдельная целостная синтаксическая единица
    virtual ~Figure() {}
};
#endif // FIGURE_H

```

Рисунок 10 - код программы

```
#ifndef LINE_H
#define LINE_H

#include <QPainter>
#include <utility>

#include <figure.h>

class Line : public Figure {
public:
    Line(std::pair<int, int> position, double amplitude, double oscillation)
        : Figure(position, amplitude, oscillation) {};

    virtual void draw(QPainter & painter, int time) const override;
};

#endif // LINE_H
```

Рисунок 11 - код программы

```
#ifndef LINE_H
#define LINE_H

#include <QPainter>
#include <utility>

#include <figure.h>

class Line : public Figure {
public:
    Line(std::pair<int, int> position, double amplitude, double oscillation)
        : Figure(position, amplitude, oscillation) {};

    virtual void draw(QPainter & painter, int time) const override;
};

#endif // LINE_H
```

Рисунок 12 - код программы

```
#ifndef WINDOWS_H
#define WINDOWS_H

#include "canvas.h"
#include <QWidget>
#include <QTextCodec>
#include <QPushButton>

class Window : public QWidget
{
protected:
    Canvas * area;
    QPushButton * btn;
public:
    Window();
};
#endif
```

Рисунок 13 - код программы

```cpp
#include <QTimerEvent>

#include "canvas.h"

Canvas::Canvas(QWidget * parent)
    : QWidget(parent), timerId(0), time(0) {
    setFixedSize(QSize(400, 400));

    line = new Line({70, 100}, 50, 2000);
    ticker = new Ticker({200, 100}, 15, 1000);
    double_ticker = new Ticker[2] {
            Ticker({320, 100}, 30, 1500),
            Ticker({320, 100}, -30, 1500)
    };
}

void Canvas::showEvent(QShowEvent *) {
    timerId = startTimer(50);
}

void Canvas::paintEvent(QPaintEvent *) {
    QPainter painter(this);
    painter.setPen(Qt::red);
    line->draw(painter, time);
    ticker->draw(painter, time);
    double_ticker[0].draw(painter, time);
    double_ticker[1].draw(painter, time);
}

void Canvas::timerEvent(QTimerEvent * event) {
    if (event->timerId() == timerId) {
        time += 50; // в идеале вынести в константу, но мне лень
        update();
    } else {
        QWidget::timerEvent(event);
    }
}

void Canvas::hideEvent(QHideEvent *) {
    killTimer(timerId);
}

```

Рисунок 14 - код программы

```cpp
#include <cmath>

#include "figure.h"

double Figure::get_angle(int time) const {
    double pi = 3.14; // Потому что Pi нет в стандартной библиотеке :/
    double alpha = 2 * pi * (time / oscillation);
    alpha = sin(alpha) * amplitude / 180 * pi;
    return alpha;
}

std::pair<int, int> Figure::rotate_vec(const std::pair<int, int> & vec, double alpha) const {
    return {
        static_cast<int>(+ cos(alpha)*vec.first + sin(alpha)*vec.second),
        static_cast<int>(- sin(alpha)*vec.first + cos(alpha)*vec.second)
    };
}

```

Рисунок 15 - код программы

```cpp
#include <QPainter>

#include <line.h>

void Line::draw(QPainter & painter, int time) const {
    std::pair<int, int> vec = rotate_vec({0, 100}, get_angle(time));

    painter.drawLine(
                position.first,
                position.second,
                position.first + vec.first,
                position.second + vec.second
    );
}

```

Рисунок 16 - код программы

```cpp
#include "window.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Window win;
    win.show();
    return a.exec();
}

```

Рисунок 17 - код программы

```cpp
#include <cmath>

#include "ticker.h"

void Ticker::draw(QPainter & painter, int time) const {
    std::pair<int, int> vec = rotate_vec({0, 100}, get_angle(time));

    painter.drawLine(
                position.first,
                position.second,
                position.first + vec.first,
                position.second + vec.second
    );
    painter.drawEllipse(
                position.first + vec.first - 10,
                position.second + vec.second - 10,
                20, 20
    );
}

```

Рисунок 18 - код программы

```cpp
#include "window.h"
#include <QVBoxLayout>

Window::Window()
{
    this->setWindowTitle("HW 3");
    area = new Canvas(this);
    btn = new QPushButton("Exit", this);
    QVBoxLayout *layout = new QVBoxLayout(this);
    layout->addWidget(area);
    layout->addWidget(btn);
    connect(btn, SIGNAL(clicked(bool)),this,SLOT(close()));
};


```

Рисунок 19 - код программы

Тестовые данные

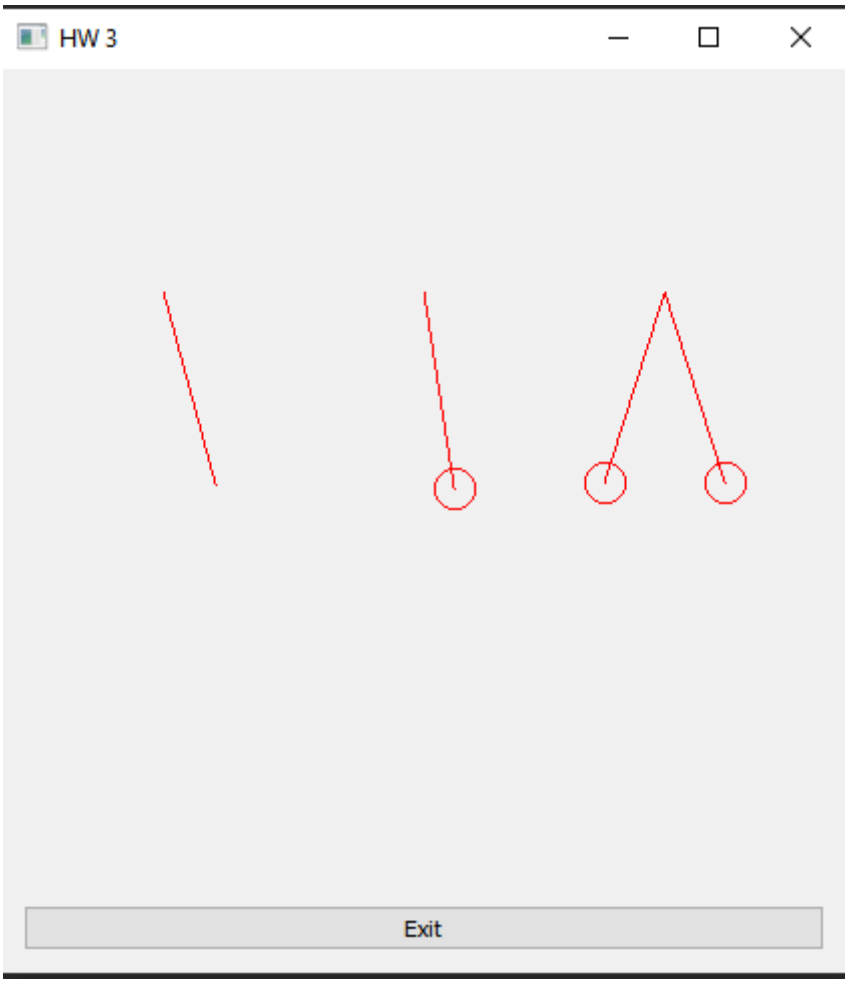


Рисунок 20 - пример работы

Вывод

Были изучены основы работы с QT.