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

по домашнему задания № 3

Название: Программирование на C++ с использованием классов

Дисциплина: Объектно-ориентированное программирование

Студент	ИУ6-22Б		С.В. Астахов
	(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 – такси. Поля: номер автомобиля, количество пассажирских мест, наличие детского кресла, состояние (занят или свободен). Методы: процедура инициализации; процедура вывода информации на экран; процедура взятия заказа, изменяющая состояние на «занят»; процедура завершения заказа, изменяющая состояние на «свободен»; функция, определяющая, подходит ли данная машина для поступившего заказа (параметры заказа: число пассажиров и наличие среди них маленького ребенка); а также функции, возвращающие значения полей по запросу.

Объект – таксопарк. Включает в себя несколько машин. Методы должны позволять: инициализировать объект, выводить информацию обо всех машинах на экран, отправлять на вызов и снимать с вызова машину с заданным номером, определять номер подходящей для поступившего заказа машины (она должна быть свободна и удовлетворять параметрам заказа).

В отчете привести диаграмму разработанных классов и объектную декомпозицию.

Исходный код

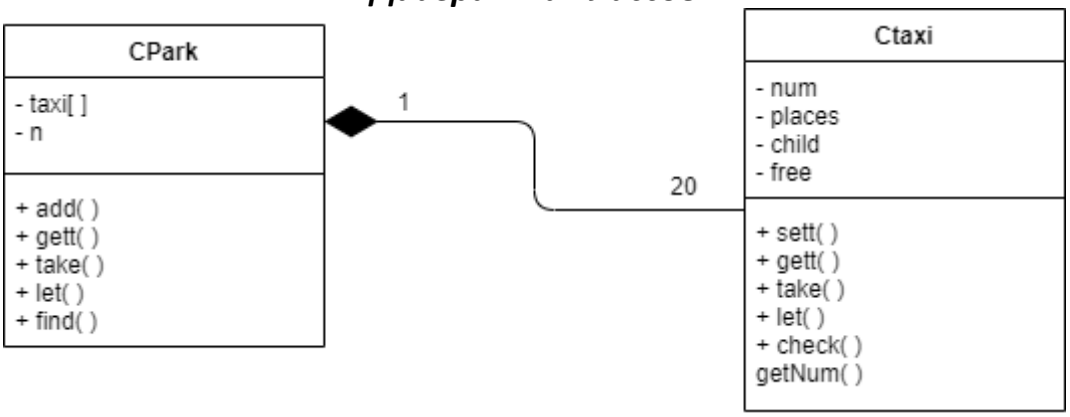
```
#include <iostream>

int main()
{
	class CTaxi {
	protected:
		int num;
		int places;
		bool child;
		bool free;
	public:
		CTaxi() {
			num = 0;
			places = 0;
			child = false;
			free = false;
		}
		void sett(int a1, int a2, int a3, int a4) {
			num = a1;
			places = a2;
			child = (a3 != 0);
			free = (a4 != 0);
		}
		void gett() {
```

```
                printf("\n num: %d \n places: %d \n child: %s \n free: %s \n", num, places,
child ? "yes" : "no", free ? "yes" : "no");
        }
        void take() {
                free = false;
        }
        void let() {
                free = true;
        }
        bool check(int a2, int a3) {
                return ((places >= a2) && ((child) || (a3 == 0)) && (free));
        }
        int getNum() {
                return num;
        }
        // + get_field()
};

class CPark {
protected:
        CTaxi taxi[20];
        int n;
public:
        CPark() {
                n = -1;
        }
        void add(int a1, int a2, int a3, int a4) {
                n++;
                taxi[n].sett(a1, a2, a3, a4);
        }
        void gett() {
                for (int i = 0; i <= n; i++) {
                        taxi[i].gett();
                }
                if (n < 0) {
                        puts("\n park is empty");
                }
        }
        void take(int target) {
                for (int i = 0; i <= n; i++) {
                        if (taxi[i].getNum() == target) {
                                taxi[i].take();
                        }
                }
        }
        void let(int target) {
                for (int i = 0; i <= n; i++) {
                        if (taxi[i].getNum() == target) {
                                taxi[i].let();
                        }
                }
```

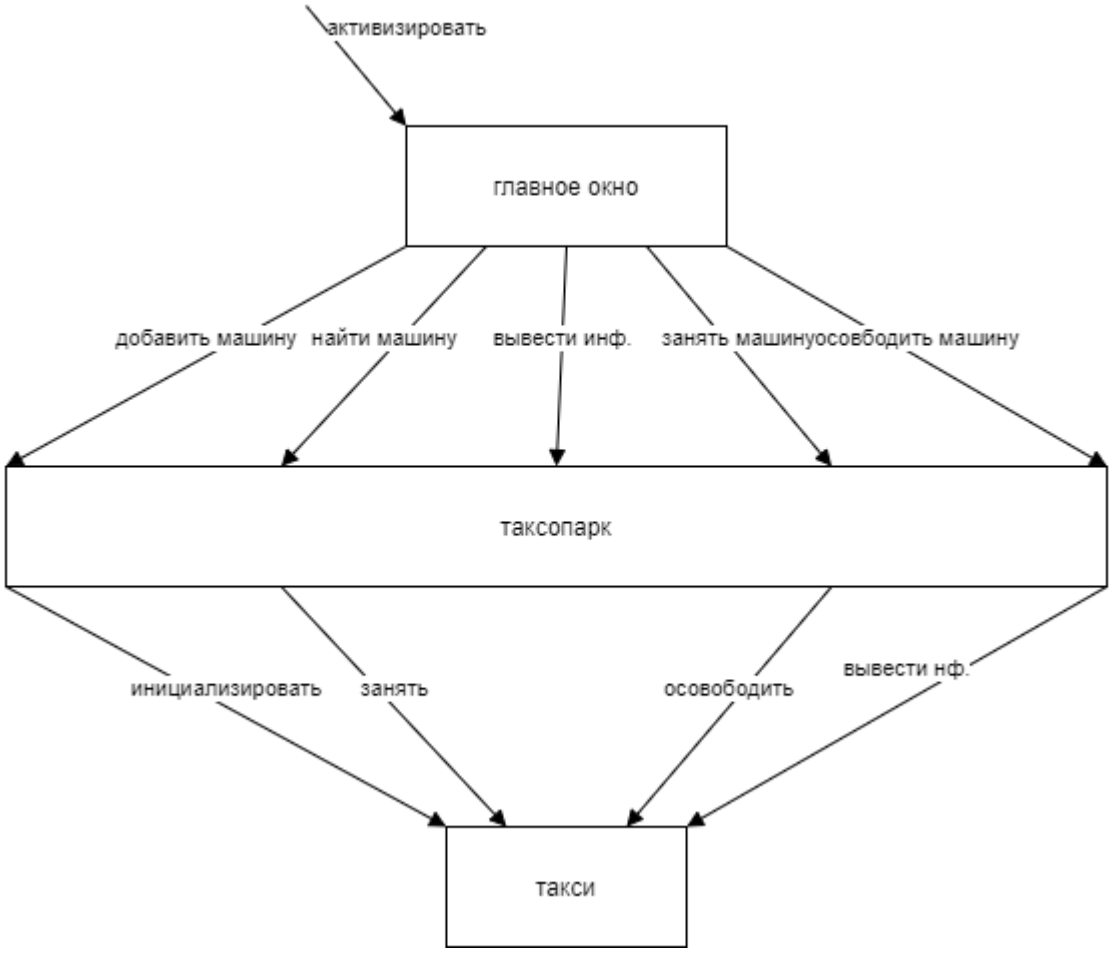
```
		}
		int find(int a2, int a3) {
			int res = -1;
			for (int i = 0; i <= n; i++) {
				if (taxi[i].check(a2, a3)) {
					res = taxi[i].getNum();
					break; // safe?
				}
			}
			return res;
		}
	};
	CPark yataxi;
	int inp1, inp2, inp3, inp4;
	int code = 73;
	while (code != 0) {
		puts("\n Enter command \n 1) add taxi \n 2) show all info \n 3) make busy by num \n 4)
make free by num \n 5) find taxi by places and child");
		scanf_s("%d", &code);
		switch (code) {
		case 0:
			puts("\n exit...");
			break;
		case 1:
			puts("\n enter num, places, childplace(0 - no, other - yes), free(0 - no, other -
yes)");
			scanf_s("%d %d %d %d", &inp1, &inp2, &inp3, &inp4);
			yataxi.add(inp1, inp2, inp3, inp4);
			break;
		case 2:
			yataxi.gett();
			break;
		case 3:
			puts("\n Enter num");
			scanf_s("%d", &inp1);
			yataxi.take(inp1);
			break;
		case 4:
			puts("\n Enter num");
			scanf_s("%d", &inp1);
			yataxi.let(inp1);
			break;
		case 5:
			puts("\n Enter places and childplace(0 - no, other - yes)");
			scanf_s("%d %d", &inp1, &inp2);
			if (yataxi.find(inp1, inp2) == -1) {
				puts("\n no such taxi, we are very sorry");
			}
			else {
				printf("\n nice car for you: %d", yataxi.find(inp1, inp2));
			}
```

```
                        break;
                default:
                        puts("\n no such option, retry");
                        break;
                }
        }


        return 0;
}
```

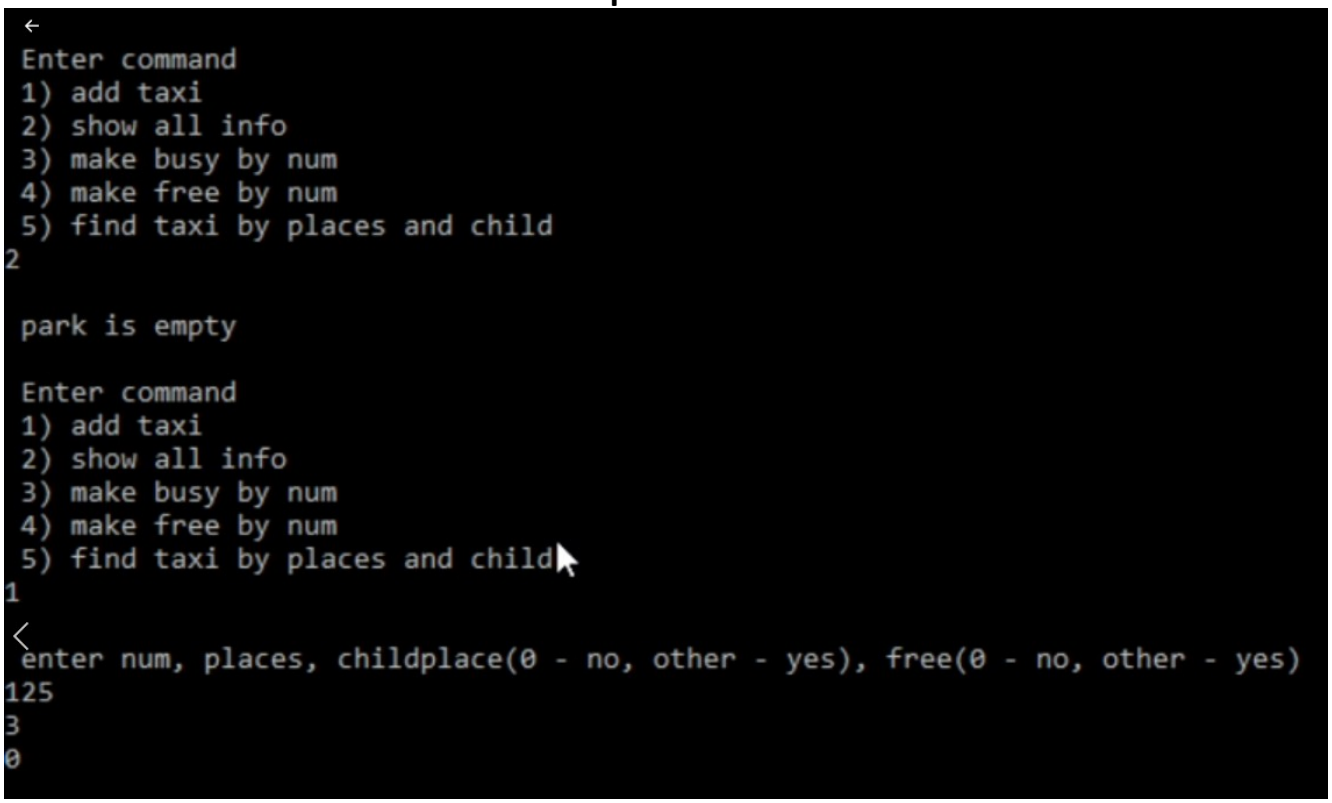
Диаграмма классов



Объектная декомпозиция



Скриншоты


Enter command
1) add taxi
2) show all info
3) make busy by num
4) make free by num
5) find taxi by places and child
2
park is empty
Enter command
1) add taxi
2) show all info
3) make busy by num
4) make free by num
5) find taxi by places and child
1
enter num, places, childplace(0 - no, other - yes), free(0 - no, other - yes)
125
3
0

Задание

Разработать программу, содержащую описание трех графических объектов: круг с вырезанной четвертью, эллипс, 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.

Исходный код

(файл front.cpp)

```
#include <QApplication>
#include "back.h"
//#include "que.h"

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   FormDialog *dialog = new FormDialog();
   dialog->show(); // отображаем окно
   return app.exec(); // запускаем цикл обработки сообщений
}
```

(файл back.h)

```
#ifndef BACK_H_
#define BACK_H_
#include <QDialog>
#include <QLineEdit>
#include <QSignalMapper>
#include <QTextEdit>
#include <QString>

class FormDialog: public QDialog
{
   Q_OBJECT
public:
   FormDialog( QWidget * parent = 0);
   virtual ~FormDialog(){};

private slots:
   void starter();
};
#endif
```

(файл back.cpp)

```
#include <QPushButton>
#include <QVBoxLayout>
#include <QTextEdit>
#include <QLineEdit>
#include <iostream>
#include <QString>
#include <QFrame>
#include <QPoint>
#include <QTimer>
#include "back.h"
#include "drawer.h"
#include "que.h"

#include <QPainter>
using namespace std;

CSmartQ qobj;
bool startb = true;
QTimer *timer1;

FormDialog::FormDialog(QWidget * parent){
  timer1 = new QTimer(this);
  QVBoxLayout *mainLayout = new QVBoxLayout();
  QPushButton *button1 = new QPushButton("Start / Stop");
  CDrawer* drawer1 = new CDrawer(this);
  bool lower = true, isOut = false;


  connect(button1, SIGNAL(clicked()), this, SLOT(starter()));
  connect(timer1, SIGNAL(timeout()), this, SLOT(repaint()));
  mainLayout->addWidget(drawer1);
  mainLayout->addWidget(button1);
  setLayout(mainLayout);
};

void FormDialog::starter(){
  if(startb)
  {
        timer1->start(150);
  }
  else{
        timer1->stop();
  }
```

```
  startb = !(startb);
};
```

(файл drawer.h)

```
#ifndef drawer_h
#define drawer_h
#include <QWidget>
#include "figures.h"
#include <QPainter>
class CDrawer : public QWidget {

  public:
    CDrawer(QWidget *parent = 0);

  protected:
    void paintEvent(QPaintEvent *event);
    void drawFigure(QPainter *qp, int id);
    CCircle *f1;
    CSquare *f2;
    CEllipse *f3;
};

#endif
```

(файл drawer.cpp)

```
//#include <QPainter>
#include "drawer.h"
#include "figures.h"
#include <QPainter>

CDrawer::CDrawer(QWidget *parent): QWidget(parent)
{
setFixedSize(QSize(800,300));
f1 = new CCircle();
f2 = new CSquare();
f3 = new CEllipse();
f1->sett(125,125,1);
f2->sett(375,125,3);
f3->sett(625,125,1);
}

void CDrawer::paintEvent(QPaintEvent *e) {

  Q_UNUSED(e);

  QPainter qp(this);
  drawFigure(&qp,0);
```

```
  drawFigure(&qp,1);
  drawFigure(&qp,2);
}

void CDrawer::drawFigure(QPainter *qp, int id) {

  QPen pen(Qt::black, 2, Qt::SolidLine);
  qp->setPen(pen);
  switch(id){
    case 0:
      f1->redraw(qp);
    break;
    case 1:
      f2->redraw(qp);
    break;
    case 2:
      f3->redraw(qp);
    break;
  }
}
```

(файл figures.h)

```
#ifndef figures_h
#define figures_h
#include <QPainter>
#include <QRectF>
#include <QWidget>
#include <cmath>

class CFigure{
  protected:
   int xc, yc, t, dt;
    void tick();

  public:
    void sett(int a1, int a2, int a3);
    CFigure();
    void redraw(QPainter *qpp);
};

class CCircle: public CFigure{

  public:
    void sett(int a1, int a2, int a3);
    void redraw(QPainter *qpp);
};
```

```
class CSquare: public CFigure{

  public:
    void sett(int a1, int a2, int a3);
    void redraw(QPainter *qpp);
};

class CEllipse: public CFigure{

  public:
    void sett(int a1, int a2, int a3);
    void redraw(QPainter *qpp);
};

#endif
```

(файл figures.cpp)

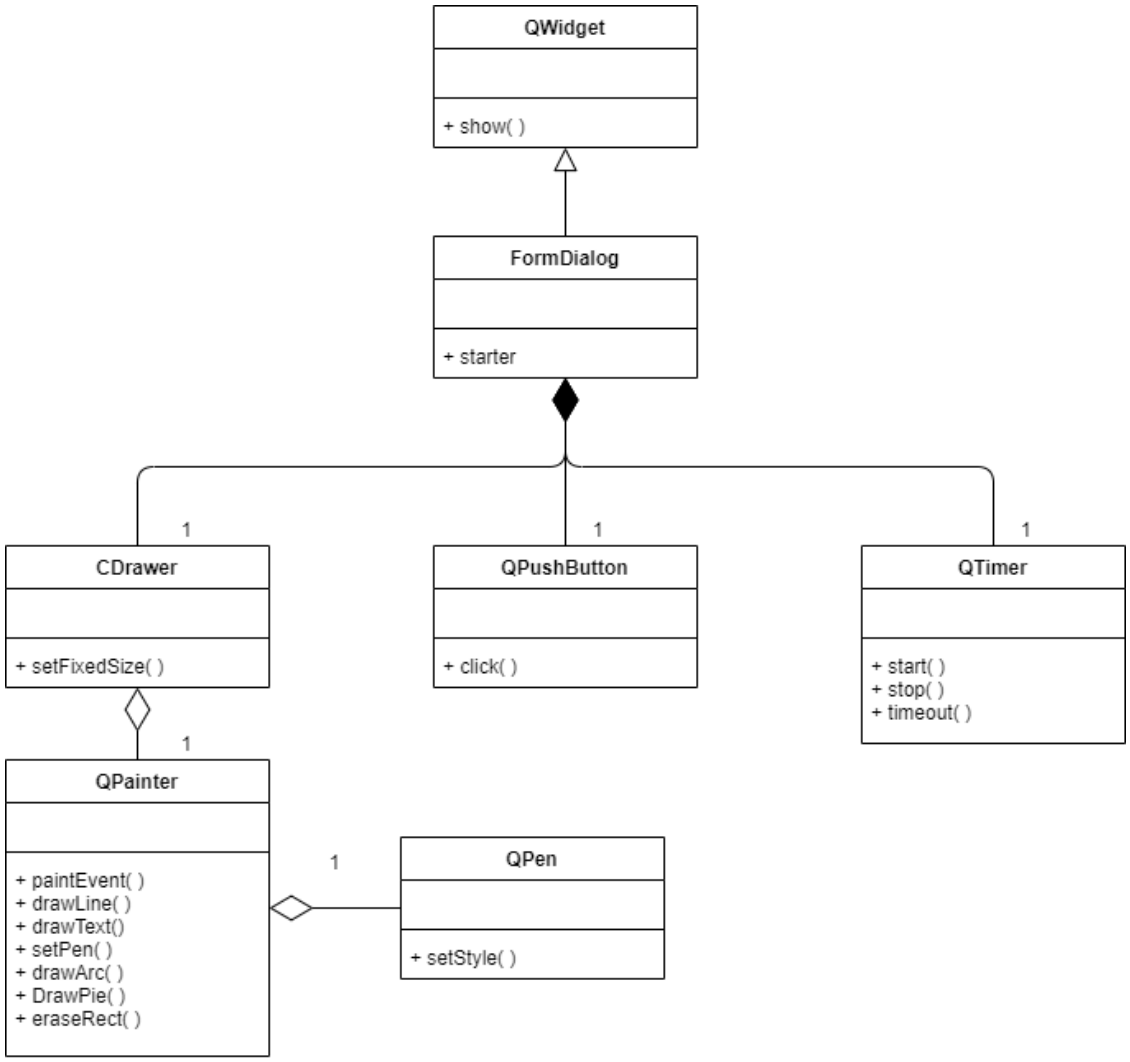
```
#include "figures.h"
using namespace std;

void CFigure::sett(int a1, int a2, int a3){
  xc = a1;
  yc = a2;
  t = 0;
  dt = a3;
}
CFigure::CFigure(){
  xc = 300;
  yc = 300;
  t = 0;
  dt = 0;
}

void CFigure::tick(){
  t += dt;
}

void CFigure::redraw(QPainter *qpp){

}

void CCircle::redraw(QPainter *qpp){
 QRectF rect(xc - 110, yc - 110, xc + 110, yc + 110);
 qpp->eraseRect(rect);
```

```
 qpp->drawPie(rect, t*16 , 270 *16);
 tick();
}

void CSquare::redraw(QPainter *qpp){
int x1, y1, x2, y2, x3, y3, x4, y4;
 QRectF rect(xc - 125, yc - 125, xc + 125, yc + 125);
qpp->eraseRect(rect);
  x1 = xc - int(90 * cos(t));
  y1 = yc - int(90 * sin(t));
  x2 = xc - int(90 * cos(t + 3.14 / 2));
  y2 = yc - int(90 * sin(t + 3.14 / 2));
  x3 = xc - int(90 * cos(t + 3.14));
  y3 = yc - int(90 * sin(t + 3.14));
  x4 = xc - int(90 * cos(t + 3 * 3.14 / 2));
  y4 = yc - int(90 * sin(t + 3 * 3.14 / 2));
  qpp->drawLine(x1, y1, x2, y2);
  qpp->drawLine(x2, y2, x3, y3);
  qpp->drawLine(x3, y3, x4, y4);
  qpp->drawLine(x4, y4, x1, y1);
  tick();
}

void CEllipse::redraw(QPainter *qpp){
 int x1, y1, x2, y2, x3, y3, x4, y4, xc1, yc1, xc2, yc2;
  QRectF rect(xc - 125, yc - 125, xc + 125, yc + 125);
  qpp->eraseRect(rect);
  x1 = xc - int(60 * cos(t));
  y1 = yc - int(60 * sin(t));
  x2 = xc - int(60 * cos(t + 3.14 / 2));
  y2 = yc - int(60 * sin(t + 3.14 / 2));
  x3 = xc - int(60 * cos(t + 3.14));
  y3 = yc - int(60 * sin(t + 3.14));
  x4 = xc - int(60 * cos(t + 3 * 3.14 / 2));
  y4 = yc - int(60 * sin(t + 3 * 3.14 / 2));
  xc1 = int((x2 + x3) / 2);
  yc1 = int((y2 + y3) / 2);
  xc2 = int((x4 + x1) / 2);
  yc2 = int((y4 + y1) / 2);

  qpp->drawLine(x1, y1, x2, y2);
  qpp->drawLine(x3, y3, x4, y4);
```

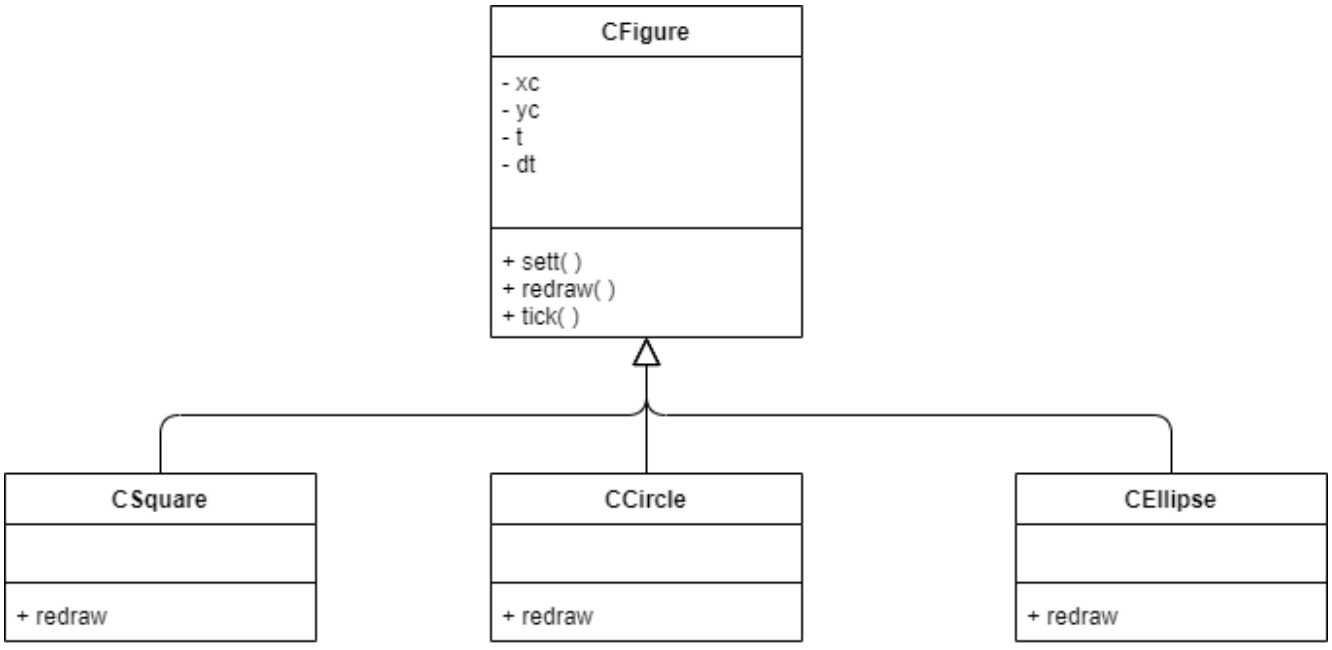
```
  qpp->drawArc(xc1 - 40, yc1 - 40, 80, 80, (-t*180*16)/3.14 - 45*16, +180*16);
  qpp->drawArc(xc2 - 40, yc2 - 40, 80, 80, (-t*180*16)/3.14- 45*16, -180*16);
  tick();
}

void CCircle::sett(int a1, int a2, int a3){
  xc = a1;
  yc = a2;
  t = 0;
  dt = a3;
}
void CSquare::sett(int a1, int a2, int a3){
  xc = a1;
  yc = a2;
  t = 0;
  dt = a3;
}
void CEllipse::sett(int a1, int a2, int a3){
  xc = a1;
  yc = a2;
  t = 0;
  dt = a3;
}
```

Диаграмма классов

Визуальные классы



Невизуальные классы



Объектная декомпозиция

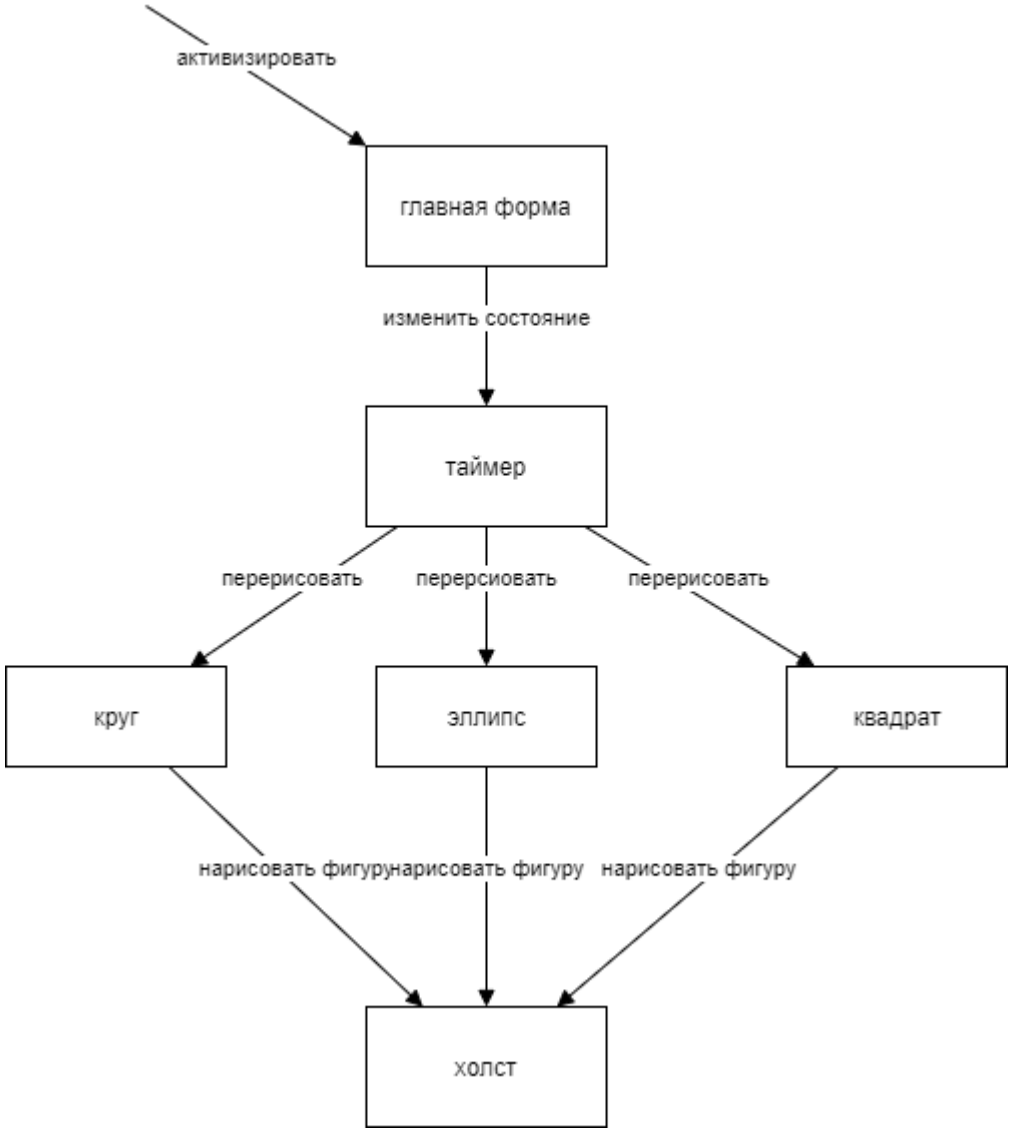
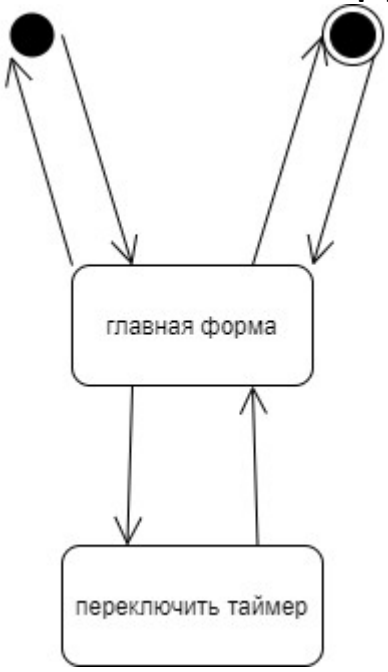
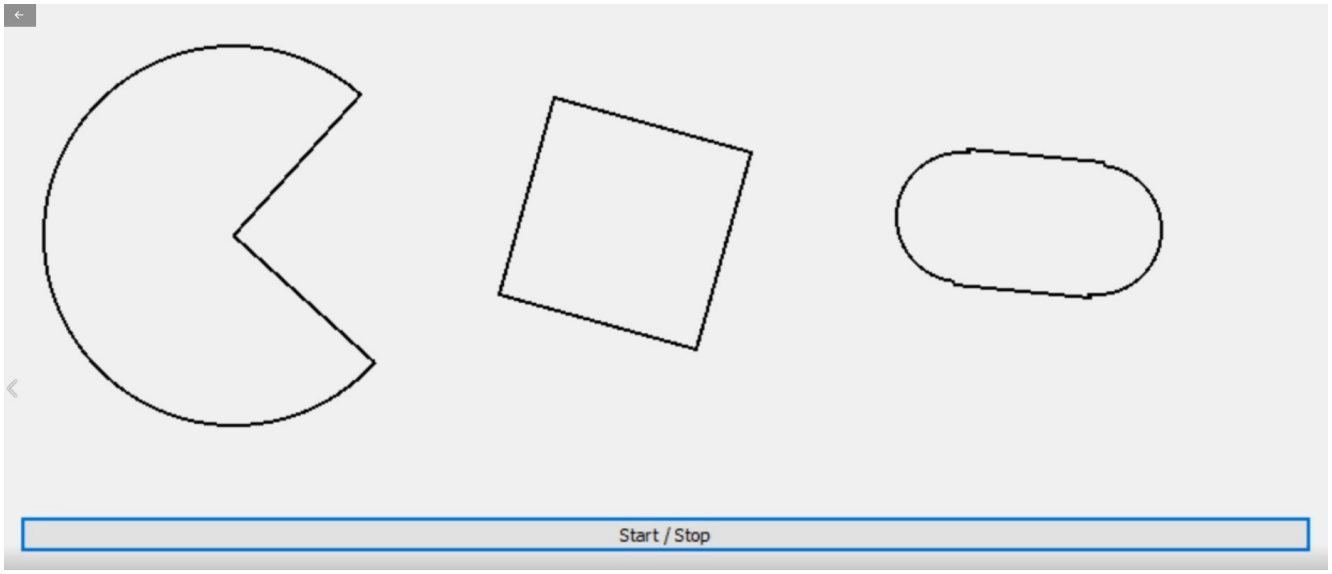


Диаграмма состояний интерфейса



Скриншоты



Вывод

- Композиция позволяет создавать сложные классы(таксопарк), вынося описание классов-частей(машина такси) за их рамки, что упрощает написание и сопровождение программ.
- Qt предоставляет широкий набор средств для работы с графикой.
- Полиморфное наследование с использованием абстрактного класса удобно, когда необходимо создать несколько классов одной природы(фигуры), но с разной реализацией методов(разные методы рисования).