Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт

з лабораторної роботи № 3 з дисципліни

«Розробка інтерфейсу користувача на C++»

Варіант 29

Виконав студент ІМ-11, Пономаренко Маргарита Альбертівна
(шифр, прізвище, ім'я, по батькові)

Перевірив Порєв В. М.
(прізвище, ім'я, по батькові)

Київ 2022

Мета роботи: отримати вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів C++, запрограмувавши графічний інтерфейс користувача.

Завдання:

1. Створити у середовищі Qt Creator проект типу Desktop Application.

2. Написати вихідний текст програми згідно варіанту завдання.

3. Скомпілювати вихідний текст і отримати виконуваний файл програми.

4. Перевірити роботу програми. Налагодити програму.

5. Проаналізувати та прокоментувати результати та вихідний текст програми.

6. Оформити звіт.

Варіант 29 + 1:

6. Масив вказівників для динамічних об’єктів типу Shape

- статичний масив Shape *pcshape[N];

7. "Гумовий" слід при вводі об’єктів

- суцільна лінія синього кольору для студентів, у яких (Ж mod 4 = 2)

8. Чотири геометричні форми (крапка, лінія, прямокутник, еліпс) можуть мати наступні різновиди вводу та відображення.

8.1. Прямокутник

Ввід прямокутника:

- по двом протилежним кутам для студентів, у яких (Ж mod 2 = 0)

Відображення прямокутника:

- чорний контур з білим заповненням для (Ж mod 5 = 0)

8.2. Еліпс

Ввід еліпсу:

- від центру до одного з кутів охоплюючого прямокутника для (Ж mod 2 = 0)

Відображення еліпсу:

- чорний контур еліпсу без заповнення для (Ж mod 5 = 0 або 2)

9. Позначка поточного типу об’єкту, що вводиться

- в заголовку вікна для

Текст головного файлу (main.cpp)

```
#include "mainwindow.h"

#include <QApplication>

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

Текст заголовку головного вікна (mainwindow.h)

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include "shapeobjectseditorview.h"
#include "toolbar.h"

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  MainWindow(QWidget *parent = nullptr);
  ~MainWindow();

private slots:
  void on_actionNew_triggered();

  void on_drawPoint_triggered();

  void on_drawLine_triggered();
```

```
  void on_drawRect_triggered();

  void on_drawElipse_triggered();

private:
  Ui::MainWindow *ui;
  ShapeObjectsEditorView *shapeEditorView;
  Toolbar *toolBar;
  void setUpToolBar();

};
#endif // MAINWINDOW_H
```

Текст реалізації головного вікна (mainwindow.cpp)

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "scenewindow.h"
#include "toolbar.h"

#include <QToolBar>

MainWindow::MainWindow(QWidget *parent)
  : QMainWindow(parent)
  , ui(new Ui::MainWindow)
{
  ui->setupUi(this);
  QApplication::instance()->setAttribute(Qt::AA_DontShowIconsInMenus, true);

  toolBar = new Toolbar(this);
  addToolBar(toolBar);
  setUpToolBar();
}

MainWindow::~MainWindow()
{
  delete ui;
}

void MainWindow::on_actionNew_triggered()
{
  shapeEditorView = createEditor(this);
}

void MainWindow::on_drawPoint_triggered()
{
  setWindowTitle("Режим малювання: крапка");
  ui->drawLine->setChecked(false);
  ui->drawRect->setChecked(false);
  ui->drawElipse->setChecked(false);
```

```
  shapeEditorView->drawPoint();
}


void MainWindow::on_drawLine_triggered()
{
  setWindowTitle("Режим малювання: лінія");
  ui->drawPoint->setChecked(false);
  ui->drawRect->setChecked(false);
  ui->drawElipse->setChecked(false);

  shapeEditorView->drawLine();
}


void MainWindow::on_drawRect_triggered()
{
  setWindowTitle("Режим малювання: прямокутник");
  ui->drawPoint->setChecked(false);
  ui->drawLine->setChecked(false);
  ui->drawElipse->setChecked(false);

  shapeEditorView->drawRect();
}


void MainWindow::on_drawElipse_triggered()
{
  setWindowTitle("Режим малювання: еліпс");
  ui->drawPoint->setChecked(false);
  ui->drawLine->setChecked(false);
  ui->drawRect->setChecked(false);

  shapeEditorView->drawElipse();
}

void MainWindow::setUpToolBar()
{
  toolBar->addAction(ui->actionNew);
  toolBar->addSeparator();
  toolBar->addAction(ui->drawPoint);
  toolBar->addAction(ui->drawLine);
  toolBar->addAction(ui->drawRect);
  toolBar->addAction(ui->drawElipse);
}
```

Текст заголовку файлу класу Point (pointShape.h)

```
#ifndef POINTSHAPE_H
#define POINTSHAPE_H
```

```
#include "shape.h"


class PointShape: public Shape
{
public:
  PointShape();
  void Show(QColor color) override;  //метод відображення фігури на сцені
  void startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene * scene) override;
  void drawOnMove(QGraphicsSceneMouseEvent *event) override;
  void endDrawing() override;

  // QGraphicsItem interface
  QRectF boundingRect() const override;  //виділення "фізичної" області на сцені для
малювання
  void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
override;  //малювання фігури
  QPainterPath shape() const override;
};

#endif // POINTSHAPE_H
```

Текст реалізації файлу класу Point (pointShape.cpp)

```
#include "pointshape.h"

#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QGraphicsScene>

PointShape::PointShape()
{

}

void PointShape::Show(QColor color)
{
  this->color = color;
  this->setPos(xs1, ys1);
}

void PointShape::startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene * scene)
{
  this->scene = scene;
  int x = event->scenePos().x();  //ініціалізація позиції x та y курсора
  int y = event->scenePos().y();
  Set(x, y, x, y);
  scene->addItem(this);  //додавання об'єкту крапка
  Show(Qt::blue);
}
```

```
void PointShape::drawOnMove(QGraphicsSceneMouseEvent *event)
{
}

void PointShape::endDrawing()
{
}

QRectF PointShape::boundingRect() const  //виділенян на сцені прямокутної області для відображення фігури
{
  return QRectF(0,0,2,2);
}

void PointShape::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)  //метод малювання крапки
{
  qDebug()<<"Paint Point"<< xs1<<", "<<ys1;
  Q_UNUSED(option);
  Q_UNUSED(widget);
  painter->setBrush(Qt::black);
  painter->setPen(Qt::NoPen);
  painter->drawEllipse(0,0,2,2);
}

QPainterPath PointShape::shape() const
{
  QPainterPath path;
  path.addRect(0,0,2,2);
  return path;
}
```

Текст заголовку файлу класу Line (lineShape.h)

```
#ifndef LINESHAPE_H
#define LINESHAPE_H
#include "shape.h"

class LineShape: public Shape
{
public:
  LineShape();
  void Show(QColor color) override;
  void startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene * scene) override;
  void drawOnMove(QGraphicsSceneMouseEvent *event) override;
  void endDrawing() override;

  QRectF boundingRect() const override;
  void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget) override;
  QPainterPath shape() const override;
```

```
private:
  int xSize() const;
  int ySize() const;
};

#endif // LINESHAPE_H
```

Текст реалізації файлу класу Line (lineShape.cpp)

```
#include "lineshape.h"
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QGraphicsScene>

LineShape::LineShape()
{

}

void LineShape::Show(QColor color)
{
  this->color = color;
  this->setPos(std::min(xs1,xs2), std::min(ys1,ys2));
}

void LineShape::startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene *scene)
{
  int x = event->scenePos().x();  //ініціалізація позиції x та y курсора
  int y = event->scenePos().y();
  Set(x, y, x, y);
  scene->addItem(this);  //додавання об'єкту лінія
  Show(Qt::blue);
}

void LineShape::drawOnMove(QGraphicsSceneMouseEvent *event)
{
  int x1 = getXs1();
  int y1 = getYs1();
  int x2 = event->scenePos().x();
  int y2 = event->scenePos().y();
  Set(x1, y1, x2, y2);
  Show(Qt::blue);
}

void LineShape::endDrawing()
{
  Show(Qt::black);
}

QRectF LineShape::boundingRect() const
```

```
{
  qreal adjust = 0.5;
  QPointF point1(0 - adjust, 0 - adjust);
  QPointF point2(xSize() + adjust, ySize() + adjust);
  qDebug()<<"Bounding rect"<<xSize()<<"; "<<ySize();
  return QRectF(point1, point2);
}

void LineShape::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget
*widget)
{
  Q_UNUSED(option);
  Q_UNUSED(widget);
  qDebug()<<"Paint Line"<< xs1<<"; "<<ys1<<"; "<<xs2<<"; "<<ys2;
  int x1,x2,y1,y2;

  if ((xs2 >xs1 ) && (ys2>ys1)){
     x1 = 0;
     y1 = 0;
     x2 = xSize();
     y2 = ySize();
  }
  if ((xs2 <= xs1 ) && (ys2>ys1)){
     x1 = xSize();
     y1 = 0;
     x2 = 0;
     y2 = ySize();
  }
  if ((xs2 >xs1 ) && (ys2<=ys1)){
     x1 = 0;
     y1 = ySize();
     x2 = xSize();
     y2 = 0;
  }
  if ((xs2 <=xs1 ) && (ys2<=ys1)){
     x1 = xSize();
     y1 = ySize();
     x2 = 0;
     y2 = 0;
  }

  QPointF point1(x1, y1);
  QPointF point2(x2, y2);

  painter->setPen(QPen(color, 2));
  painter->setBrush(color);
  painter->drawLine(point1, point2);
}

QPainterPath LineShape::shape() const
{
  QPainterPath path;
```

```
  QPointF point1(0, 0);
  QPointF point2(xSize(), ySize());
  path.addRect(QRectF(point1, point2));
  return path;
}

int LineShape::xSize() const
{
  return abs(xs1 - xs2);
}

int LineShape::ySize() const
{
  return abs(ys1 - ys2);
}
```

Текст заголовку файлу класу Rect (rectShape.h)

```
#ifndef RECTSHAPE_H
#define RECTSHAPE_H
#include "shape.h"

class RectShape: public Shape
{
public:
  RectShape();

  void Show(QColor color) override;
  void startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene * scene) override;
  void drawOnMove(QGraphicsSceneMouseEvent *event) override;
  void endDrawing() override;

  QRectF boundingRect() const override;
  void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
override;
  QPainterPath shape() const override;

private:
  int xSize() const;
  int ySize() const;
};

#endif // RECTSHAPE_H
```

Текст реалізації файлу класу Rect (rectShape.cpp)

```
#include "rectshape.h"
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QGraphicsScene>
```

```
RectShape::RectShape()
{

}

void RectShape::Show(QColor color)
{
  this->color = color;
  this->setPos(std::min(xs1,xs2), std::min(ys1,ys2));
}

void RectShape::startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene *scene)
{
  int x = event->scenePos().x();  //ініціалізація позиції x та y курсора
  int y = event->scenePos().y();
  Set(x, y, x, y);
  scene->addItem(this);  //додавання об'єкту лінія
  Show(Qt::blue);
}

void RectShape::drawOnMove(QGraphicsSceneMouseEvent *event)
{
  int x1 = getXs1();
  int y1 = getYs1();
  int x2 = event->scenePos().x();
  int y2 = event->scenePos().y();
  Set(x1, y1, x2, y2);
  Show(Qt::blue);
}

void RectShape::endDrawing()
{
  Show(Qt::black);
}

QRectF RectShape::boundingRect() const
{
  qreal adjust = 0.5;
  QPointF point1(0 - adjust - xSize() / 2, 0 - adjust - ySize() / 2);
  QPointF point2(xSize() / 2 + adjust, ySize() / 2 + adjust);
  qDebug()<<"Bounding rect"<<xSize()<<"; "<<ySize();
  return QRectF(point1, point2);
}

void RectShape::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget
*widget)
{
  Q_UNUSED(option);
  Q_UNUSED(widget);
  qDebug()<<"Paint Line"<< xs1<<"; "<<ys1<<"; "<<xs2<<"; "<<ys2;
  int x1,x2,y1,y2;
```

```
    if ((xs2 >xs1 ) && (ys2>ys1)){
        qDebug()<<"1 path";
        x1 = -xSize();
        y1 = -ySize();
        x2 = xSize();
        y2 = ySize();
    }
    if ((xs2 <= xs1 ) && (ys2>ys1)){
        qDebug()<<"2 path";
        x1 = 0;
        y1 = -ySize();
        x2 = xSize();
        y2 = ySize()  + ySize() / 2;
    }
    if ((xs2 >xs1 ) && (ys2<=ys1)){
        qDebug()<<"3 path";
        x1 = -xSize();
        y1 = 0;
        x2 = xSize() + xSize() / 2;
        y2 = ySize();
    }
    if ((xs2 <=xs1 ) && (ys2<=ys1)){
        qDebug()<<"4 path";
        x1 = 0;
        y1 = 0;
        x2 = xSize();
        y2 = ySize();
    }

    QPointF point1(x1, y1);
    QPointF point2(x2, y2);

    if(color == Qt::blue){
        painter->setPen(QPen(color, 2));
    }
    else{
        painter->setPen(QPen(color, 2));
        painter->setBrush(Qt::white);
    }
    painter->drawRect(x1, y1, x2, y2);
}

QPainterPath RectShape::shape() const
{
    QPainterPath path;
    QPointF point1(0, 0);
    QPointF point2(xSize(), ySize());
    path.addRect(QRectF(point1, point2));
    return path;
}
```

```
int RectShape::xSize() const
{
  return abs(xs1 - xs2) * 2;
}

int RectShape::ySize() const
{
  return abs(ys1 - ys2) * 2;
}
```

Текст заголовку файлу класу Elipse (elipseShape.h)

```
#ifndef ELIPSESHAPE_H
#define ELIPSESHAPE_H
#include "shape.h"

class ElipseShape: public Shape
{
public:
  ElipseShape();

  void Show(QColor color) override;
  void startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene * scene) override;
  void drawOnMove(QGraphicsSceneMouseEvent *event) override;
  void endDrawing() override;


  QRectF boundingRect() const override;
  void paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
override;
  QPainterPath shape() const override;

private:
  int xSize() const;
  int ySize() const;
};

#endif // ELIPSESHAPE_H
```

Текст реалізації файлу класу Elipse (elipseShape.cpp)

```
#include "elipseshape.h"
#include <QPainter>
#include <QGraphicsSceneMouseEvent>
#include <QGraphicsScene>

ElipseShape::ElipseShape()
{

}
```

```
void ElipseShape::Show(QColor color)
{
  this->color = color;
  this->setPos(std::min(xs1,xs2), std::min(ys1,ys2));
}

void ElipseShape::startDrawing(QGraphicsSceneMouseEvent *event, QGraphicsScene *scene)
{
  int x = event->scenePos().x();  //ініціалізація позиції x та y курсора
  int y = event->scenePos().y();
  Set(x, y, x, y);
  scene->addItem(this);  //додавання об'єкту лінія
  Show(Qt::blue);
}

void ElipseShape::drawOnMove(QGraphicsSceneMouseEvent *event)
{
  int x1 = getXs1();
  int y1 = getYs1();
  int x2 = event->scenePos().x();
  int y2 = event->scenePos().y();
  Set(x1, y1, x2, y2);
  Show(Qt::blue);

}

void ElipseShape::endDrawing()
{
  Show(Qt::black);
}

QRectF ElipseShape::boundingRect() const
{
  qreal adjust = 0.5;
  QPointF point1(0 - adjust, 0 - adjust);
  QPointF point2(xSize() + adjust, ySize() + adjust);
  qDebug()<<"Bounding rect"<<xSize()<<"; "<<ySize();
  return QRectF(point1, point2);
}

void ElipseShape::paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget
*widget)
{
  Q_UNUSED(option);
  Q_UNUSED(widget);
  qDebug()<<"Paint Elipse"<< xs1<<"; "<<ys1<<"; "<<xs2<<"; "<<ys2;
  int x1,x2,y1,y2;

  if ((xs2 >xs1 ) && (ys2>ys1)){
     qDebug()<<"1 path";
     x1 = -xSize();
```

```
        y1 = -ySize();
        x2 = xSize();
        y2 = ySize();
    }
    if ((xs2 <= xs1 ) && (ys2>ys1)){
        qDebug()<<"2 path";
        x1 = 0;
        y1 = -ySize();
        x2 = xSize();
        y2 = ySize()  + ySize() / 2;
    }
    if ((xs2 >xs1 ) && (ys2<=ys1)){
        qDebug()<<"3 path";
        x1 = -xSize();
        y1 = 0;
        x2 = xSize() + xSize() / 2;
        y2 = ySize();
    }
    if ((xs2 <=xs1 ) && (ys2<=ys1)){
        qDebug()<<"4 path";
        x1 = 0;
        y1 = 0;
        x2 = xSize();
        y2 = ySize();
    }

    QPointF point1(x1, y1);
    QPointF point2(x2, y2);

    painter->setPen(QPen(color, 2));
    painter->drawEllipse(x1, y1, xSize(), ySize());
}

QPainterPath ElipseShape::shape() const
{
    QPainterPath path;
    QPointF point1(0, 0);
    QPointF point2(xSize(), ySize());
    path.addRect(QRectF(point1, point2));
    return path;
}

int ElipseShape::xSize() const
{
    return abs(xs1 - xs2);
}

int ElipseShape::ySize() const
{
    return abs(ys1 - ys2);
}
```

Текст заголовку файлу класу Toolbar (toolbar.h)

```
#ifndef TOOLBAR_H
#define TOOLBAR_H

#include <QToolBar>
#include <QApplication>

class Toolbar : public QToolBar
{
  Q_OBJECT

public:
  Toolbar(QWidget *parent = nullptr);
  ~Toolbar();

private:
};

#endif // TOOLBAR_H
```

Текст реалізації файлу класу Toolbar (toolbar.cpp)

```
#include <QToolBar>
#include <QIcon>
#include <QAction>
#include "toolbar.h"

Toolbar::Toolbar(QWidget *parent): QToolBar(parent) {
 qDebug()<<"Creating Toolbar";
}

Toolbar::~Toolbar()
{
}
```

Результати тестування програми

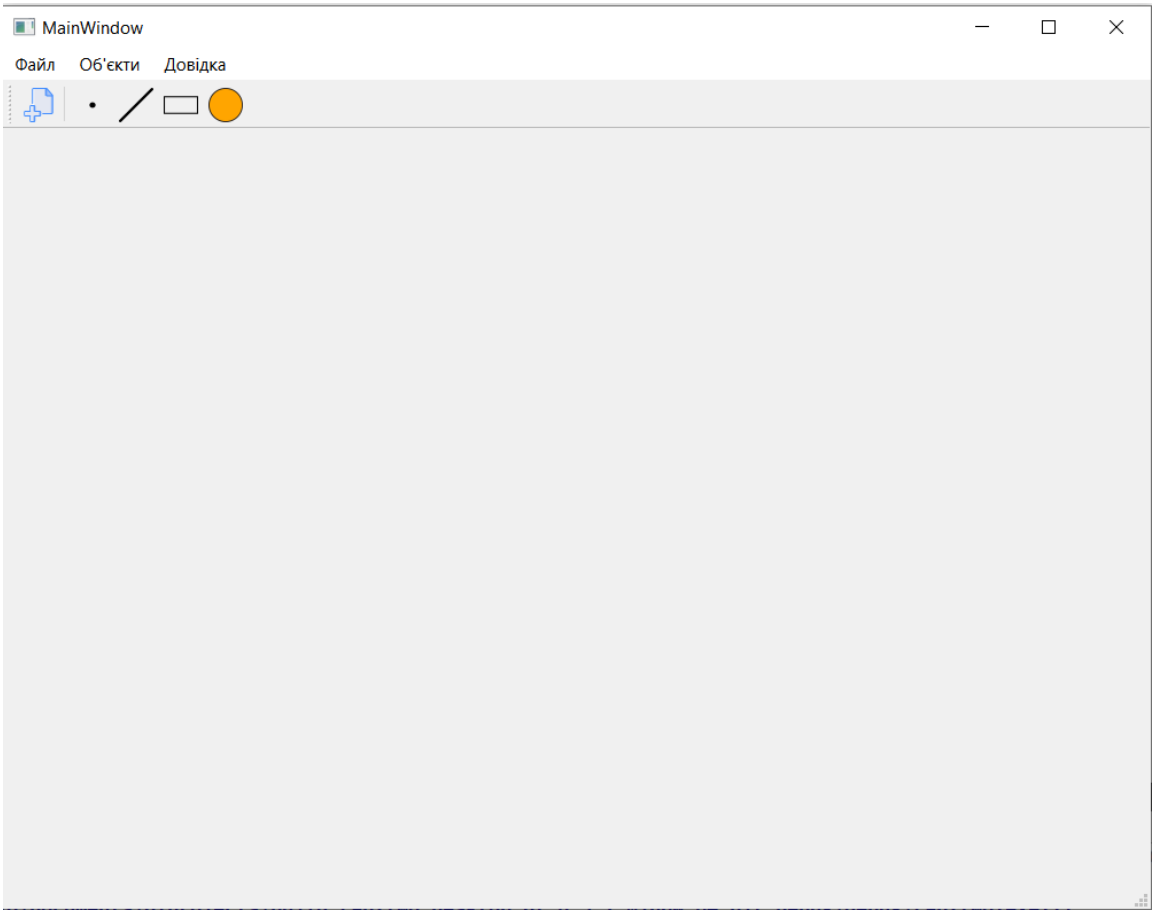


Рис 1.1: Демонстрація інтерфейсу програми з toolBar

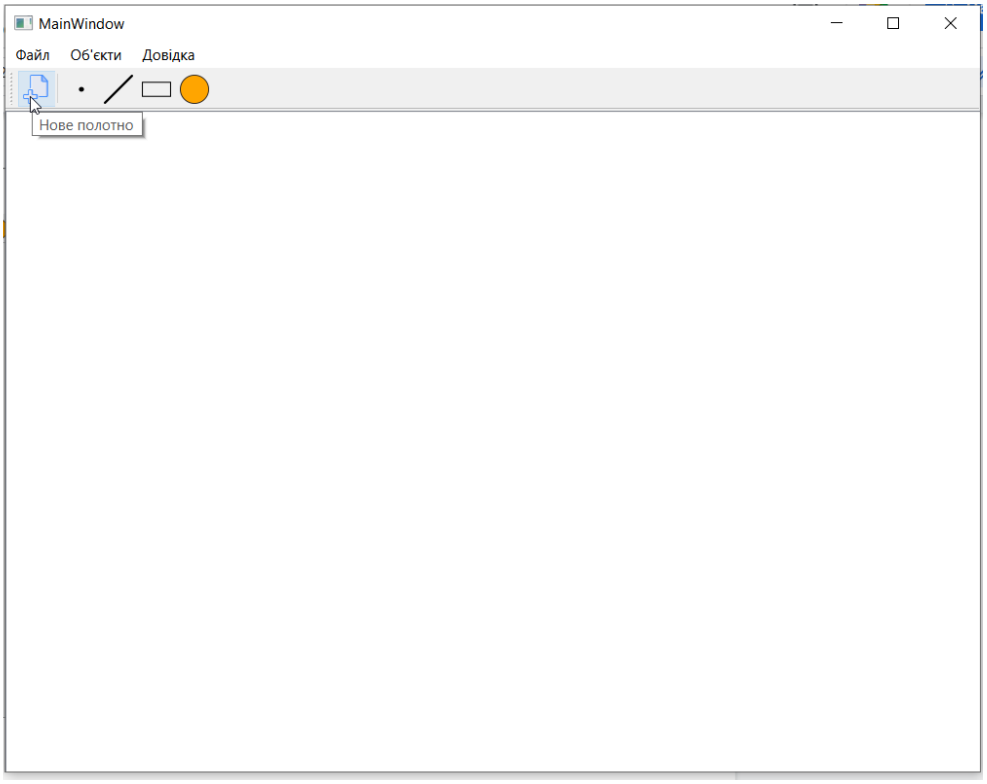


Рис 1.2: Демонстрація інструменту “ Нове полотно” (створює полотно для малювання)

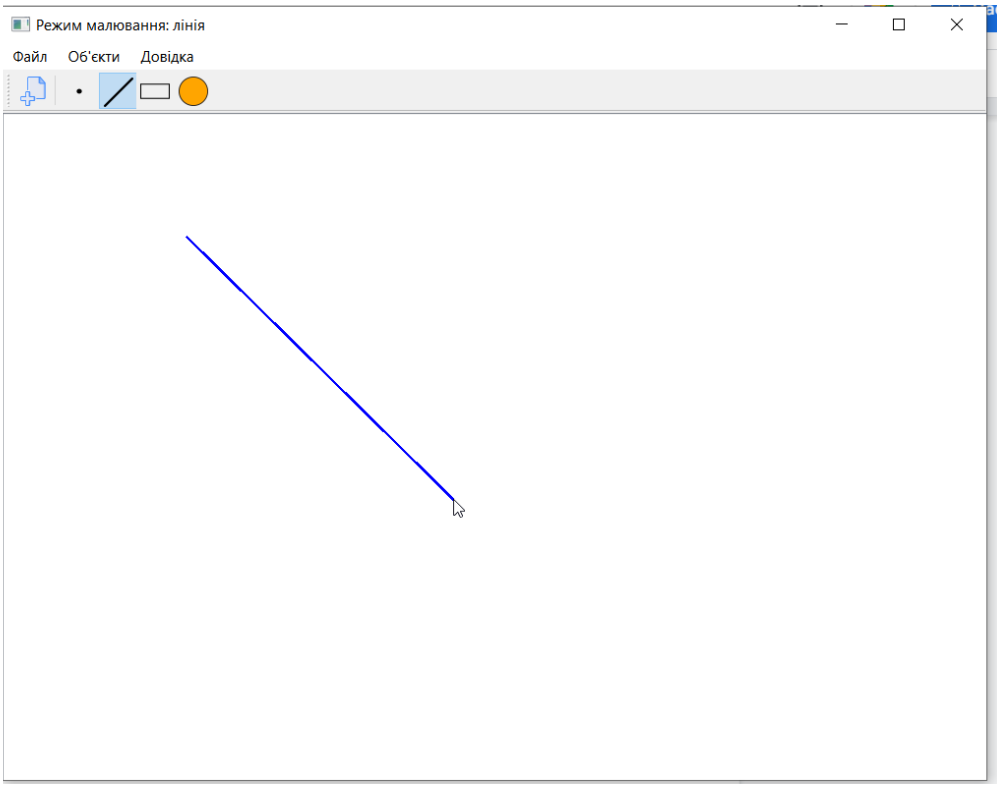


Рис 1.3: Демонстрація роботи інструменту лінія

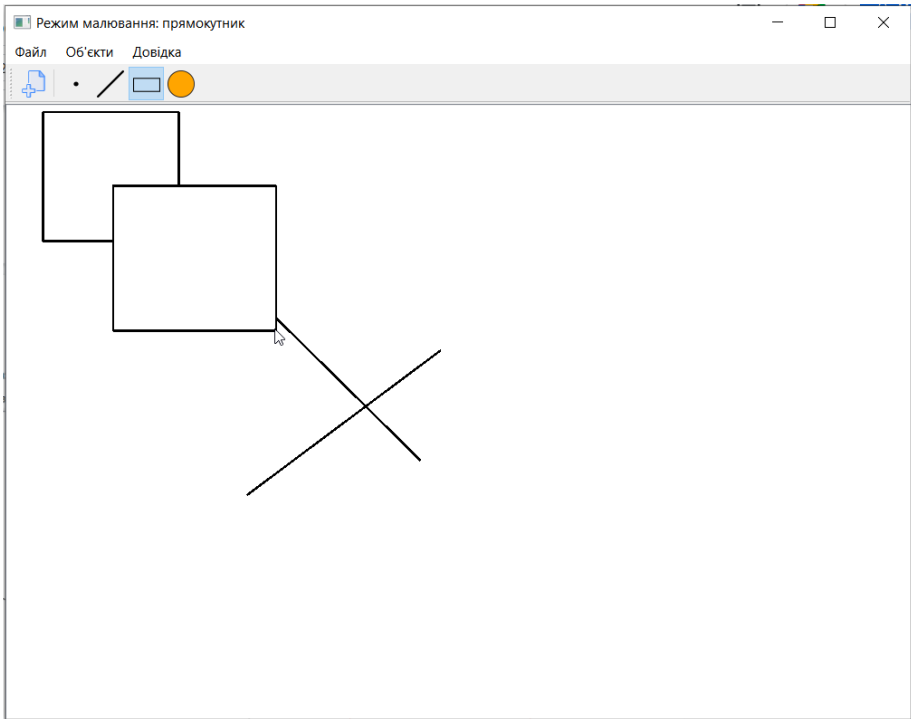


Рис 1.4: Демонстрація роботи інструменту прямокутник з білим заповненням

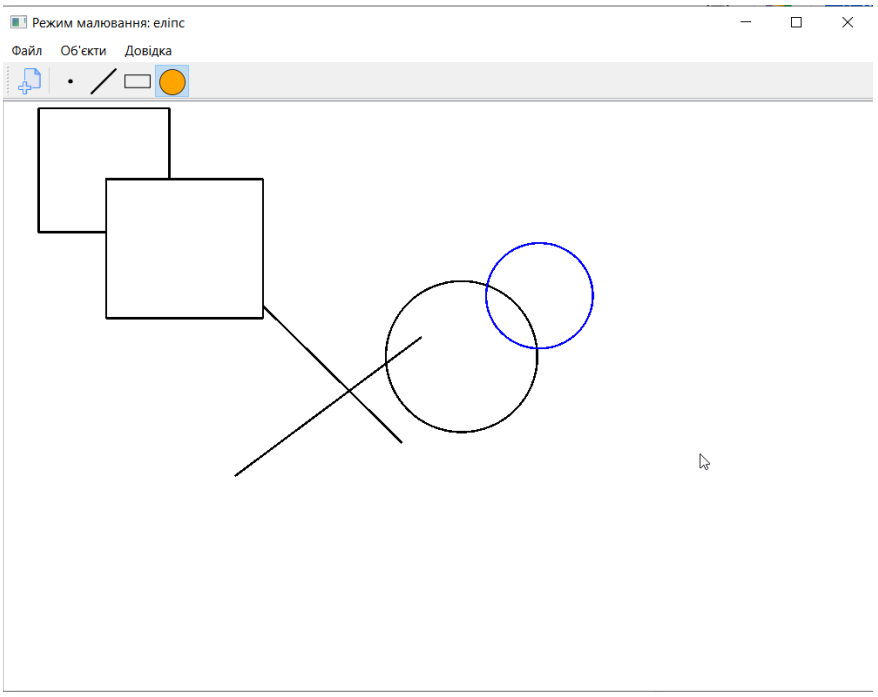
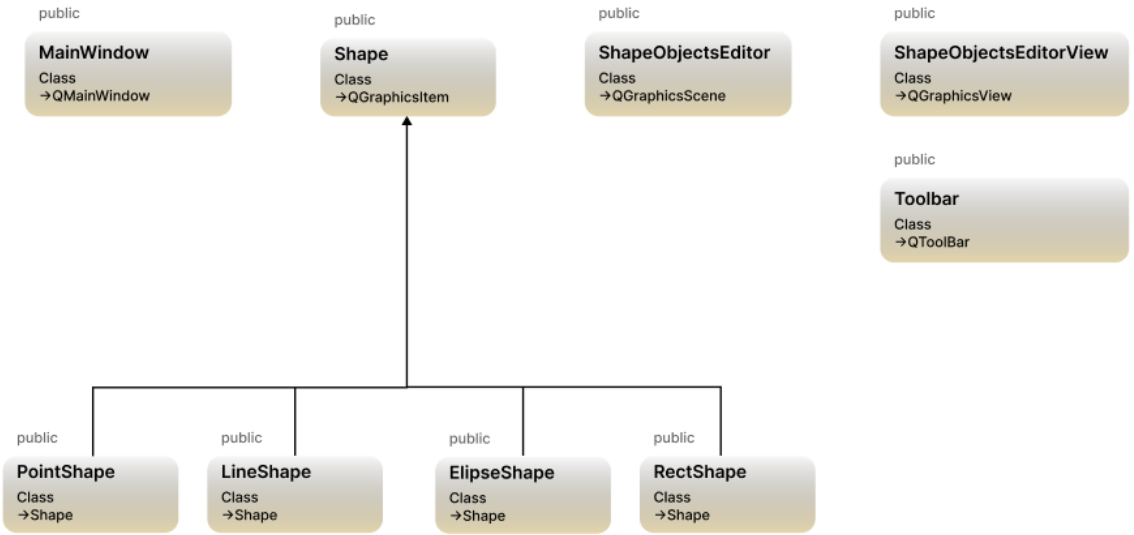


Рис 1.5: Демонстрація роботи інструменту еліпс без заповнення

Діаграма класів

*з урахуванням особливості розробки в середовищі Qt Creator



Висновок

У цій лабораторній роботі було створено програму для Windows на основі проєктів для Qt Creator з використанням інкапсуляції, абстракції типів, успадкування та поліморфізму на основі класів C++, запрограмовано простий графічний редактор в

об’єктно-орієнтованому стилі, в якому є можливість намалювати такі фігури: крапка, лінія, прямокутник та еліпс. Також створенно клас Toolbar та реалізовано такі методи: масштабування вікна, toolBar з кнопками малювання фігур та підказки до назв кнопок.