[Лабораторная работа №6. Паттерн «Адаптер» 1](#_Toc451457759)

[Обязательные задания 1](#_Toc451457760)

[Задание 1 – Адаптер объекта – 100 баллов 1](#_Toc451457761)

[Бонус – 40 баллов за unit-тестирование адаптера 6](#_Toc451457762)

[Бонус – 20 баллов за UML-диаграмму классов 6](#_Toc451457763)

[Дополнительные задания 6](#_Toc451457764)

[Задание 2 – Адаптер класса – 30 баллов 6](#_Toc451457765)

[Бонус – 40 баллов за unit-тестирование адаптера 6](#_Toc451457766)

[Бонус – 10 баллов за UML-диаграмму классов 6](#_Toc451457767)

[Задание 3 – Поддержка цветных изображений – 60 баллов 6](#_Toc451457768)

[Бонус – 30 баллов за unit-тестирование адаптера 8](#_Toc451457769)

# Лабораторная работа №6. Паттерн «Адаптер»

На оценку «удовлетворительно» необходимо набрать 100 баллов.

На оценку «хорошо» необходимо набрать 200 баллов.

На оценку «отлично» необходимо набрать 300 баллов.

**Дополнительные задания принимаются только после выполнения обязательных заданий.**

## Обязательные задания

### Задание 1 – Адаптер объекта – 100 баллов

IT компания разработала приложение позволяющее рисовать треугольники и прямоугольники. В процессе разработки, команда использовала библиотеку **graphics\_lib**, предоставляющую API для рисования простейших графических примитивов.

// Пространство имен графической библиотеки (недоступно для изменения)

namespace graphics\_lib

{

// Холст для рисования

class ICanvas

{

public:

// Ставит "перо" в точку x, y

virtual void MoveTo(int x, int y) = 0;

// Рисует линию с текущей позиции, передвигая перо в точку x,y

virtual void LineTo(int x, int y) = 0;

virtual ~ICanvas() = default;

};

// Реализация холста для рисования

class CCanvas : public ICanvas

{

public:

void MoveTo(int x, int y) override

{

cout << "MoveTo (" << x << ", " << y << ")" << endl;

}

void LineTo(int x, int y) override

{

cout << "LineTo (" << x << ", " << y << ")" << endl;

}

};

}

Для рисования более сложных фигур команда решила использовать библиотеку **shape\_drawing\_lib**, позволяющую рисовать объекты, используя API библиотеки **graphics\_lib**.

Код библиотеки представлен ниже.

// Пространство имен библиотеки для рисования фигур (использует graphics\_lib)

// Код библиотеки недоступен для изменения

namespace shape\_drawing\_lib

{

struct Point

{

int x;

int y;

};

// Интерфейс объектов, которые могут быть нарисованы на холсте из graphics\_lib

class ICanvasDrawable

{

public:

virtual void Draw(graphics\_lib::ICanvas & canvas)const = 0;

virtual ~ICanvasDrawable() = default;

};

class CTriangle : public ICanvasDrawable

{

public:

CTriangle(const Point & p1, const Point & p2, const Point & p3)

{

// TODO: написать код конструктора

}

void Draw(graphics\_lib::ICanvas & canvas)const override

{

// TODO: написать код рисования треугольника на холсте

}

private:

// TODO: дописать приватную часть

};

class CRectangle : public ICanvasDrawable

{

public:

CRectangle(const Point & leftTop, int width, int height)

{

// TODO: написать код конструктора

}

void Draw(graphics\_lib::ICanvas & canvas)const override

{

// TODO: написать код рисования прямоугольника на холсте

}

private:

// TODO: дописать приватную часть

};

// Художник, способный рисовать ICanvasDrawable-объекты на ICanvas

class CCanvasPainter

{

public:

CCanvasPainter(graphics\_lib::ICanvas & canvas)

{

// TODO: дописать конструктор класса

}

void Draw(const ICanvasDrawable & drawable)

{

// TODO: дописать код рисования ICanvasDrawable на canvas, переданном в конструктор

}

private:

// TODO: дописать приватную часть

};

}

Для рисования приложение использует вспомогательный класс CCanvasPainter:

// Пространство имен приложения (доступно для модификации)

namespace app

{

void PaintPicture(shape\_drawing\_lib::CCanvasPainter & painter)

{

using namespace shape\_drawing\_lib;

CTriangle triangle({ 10, 15 }, { 100, 200 }, { 150, 250 });

CRectangle rectangle({ 30, 40 }, 18, 24);

// TODO: нарисовать прямоугольник и треугольник при помощи painter

}

void PaintPictureOnCanvas()

{

graphics\_lib::CCanvas simpleCanvas;

shape\_drawing\_lib::CCanvasPainter painter(simpleCanvas);

PaintPicture(painter);

}

}

int main()

{

app::PaintPictureOnCanvas();

return 0;

}

Вскоре вышла графическая библиотека **modern\_graphics\_lib**, позволяющая рисовать графические примитивы со более высоким качеством.

// Пространство имен современной графической библиотеки (недоступно для изменения)

namespace modern\_graphics\_lib

{

class CPoint

{

public:

CPoint(int x, int y) :x(x), y(y) {}

int x;

int y;

};

// Класс для современного рисования графики

class CModernGraphicsRenderer

{

public:

CModernGraphicsRenderer(ostream & strm) : m\_out(strm)

{

}

~CModernGraphicsRenderer()

{

if (m\_drawing) // Завершаем рисование, если оно было начато

{

EndDraw();

}

}

// Этот метод должен быть вызван в начале рисования

void BeginDraw()

{

if (m\_drawing)

{

throw logic\_error("Drawing has already begun");

}

m\_out << "<draw>" << endl;

m\_drawing = true;

}

// Выполняет рисование линии

void DrawLine(const CPoint & start, const CPoint & end)

{

if (!m\_drawing)

{

throw logic\_error("DrawLine is allowed between BeginDraw()/EndDraw() only");

}

m\_out << boost::format(R"( <line fromX="%1%" fromY="%2" toX="%3%" toY="%4%"/>)")

<< endl;

}

// Этот метод должен быть вызван в конце рисования

void EndDraw()

{

if (!m\_drawing)

{

throw logic\_error("Drawing has not been started");

}

m\_out << "</draw>" << endl;

m\_drawing = false;

}

private:

ostream & m\_out;

bool m\_drawing = false;

};

}

Использование новой графической библиотеки вместо старой осложняется следующими факторами:

* Используемая библиотека shape\_drawing\_lib основана на библиотеке graphics\_lib, а не modern\_graphics\_lib
* Разработчики библиотеки shape\_drawing\_lib пока не заинтересованы в поддержке библиотеки modern\_graphics\_lib по следующим причинам:
  + И shape\_drawing\_lib, и graphics\_lib являются кроссплатформенными, а modern\_graphics\_lib – нет
  + У них есть свои планы развития своей библиотеки
* Библиотека modern\_graphics\_lib требует для своей работы ОС, начиная с Windows 7, в то время как приложение все еще должно поддерживать и более старые операционные системы: Windows XP и Windows Vista, которые прекрасно поддерживаются graphics\_lib

Принимая во внимание важность качественного отображения фигур, а также распространенность различных версий Windows, команда приняла решение использовать modern\_graphics\_lib на современных ОС, а на XP и Vista – graphics\_lib.

С использованием адаптера объектов реализуйте данный функционал, не изменяя код ни одной из библиотек, а также код функции PaintPicture.

// Пространство имен приложения (доступно для модификации)

namespace app

{

void PaintPicture(shape\_drawing\_lib::CCanvasPainter & painter);

void PaintPictureOnCanvas();

void PaintPictureOnModernGraphicsRenderer()

{

modern\_graphics\_lib::CModernGraphicsRenderer renderer(cout);

(void)&renderer; // устраняем предупреждение о неиспользуемой переменной

// TODO: при помощи существующей функции PaintPicture() нарисовать

// картину на renderer

// Подсказка: используйте паттерн "Адаптер"

}

}

int main()

{

cout << "Should we use new API (y)?";

string userInput;

if (getline(cin, userInput) && (userInput == "y" || userInput == "Y"))

{

app::PaintPictureOnModernGraphicsRenderer();

}

else

{

app::PaintPictureOnCanvas();

}

return 0;

}

#### Бонус – 40 баллов за unit-тестирование адаптера

Бонус начисляется за покрытие функционала разработанного адаптера объекта автоматическими тестами

#### Бонус – 20 баллов за UML-диаграмму классов

Нарисуйте UML-диаграмму классов системы, выделите на ней адаптер.

## Дополнительные задания

### Задание 2 – Адаптер класса – 30 баллов

На основе программы из задания «Адаптер объекта» реализуйте задачу при помощи адаптера класса. Сравните оба решения. Отметьте их достоинства и недостатки.

#### Бонус – 40 баллов за unit-тестирование адаптера

Бонус начисляется за покрытие функционала разработанного адаптера объекта автоматическими тестами

#### Бонус – 10 баллов за UML-диаграмму классов

В дополнение к диаграмме классов из задания «Адаптер объекта» нарисуйте диаграмму классов

### Задание 3 – Поддержка цветных изображений – 60 баллов

Разработчики библиотеки **graphics\_lib** решили поддержать в ней возможность рисования цветных примитивов и доработали возможности холста.

namespace graphics\_lib

{

// Холст для рисования

class ICanvas

{

public:

// Установка цвета в формате 0xRRGGBB

virtual void SetColor(uint32\_t rgbColor) = 0;

virtual void MoveTo(int x, int y) = 0;

virtual void LineTo(int x, int y) = 0;

virtual ~ICanvas() = default;

};

// Реализация холста для рисования

class CCanvas : public ICanvas

{

public:

void SetColor(uint32\_t rgbColor) override

{

// TODO: вывести в output цвет в виде строки SetColor (#RRGGBB)

}

void MoveTo(int x, int y) override

{

// Реализация остается без изменения

}

void LineTo(int x, int y) override

{

// Реализация остается без изменения

}

};

}

Разработчики библиотеки **shape\_drawing\_lib** также поддержали возможность создания цветных фигур – конструктор прямоугольников и треугольников принимает опциональный параметр color (тип uint32\_t), равный по умолчанию 0x000000.

Разработчики modern\_graphics\_lib также поддержали цвета в своей библиотеке, но, в отличие от graphics\_lib, поддерживающей цвет в формате **#RRGGBB**, поддерживает цвет в формате RGBA, причем каждый компонент цвета принимает значение с плавающей запятой в диапазоне от 0.0 до 1.0. Четвертый компонент цвета задает альфа-составляющую, где значение 0.0 соответствует полностью прозрачному цветау а 1.0 – полностью непрозрачному:

namespace modern\_graphics\_lib

{

class CPoint

{

// Реализация остается без изменения

};

// Цвет в формате RGBA, каждый компонент принимает значения от 0.0f до 1.0f

class CRGBAColor

{

public:

CRGBAColor(float r, float g, float b, float a):r(r), g(g), b(b), a(a){}

float r, g, b, a;

};

// Класс для современного рисования графики

class CModernGraphicsRenderer

{

public:

CModernGraphicsRenderer(ostream & strm);

~CModernGraphicsRenderer();

void BeginDraw();

// Выполняет рисование линии

void DrawLine(const CPoint & start, const CPoint & end, const CRGBAColor& color)

{

// TODO: выводит в output инструкцию для рисования линии. Пример:

// <line fromX="3" fromY="5" toX="5" toY="17">

// <color r="0.35" g="0.47" b="1.0" a="1.0" />

// </line>

// Можно вызывать только между BeginDraw() и EndDraw()

}

void EndDraw();

private:

// Реализация остается без изменений

};

}

Допишите недостающий код для библиотек graphics\_lib, shape\_drawing\_lib и modern\_graphics\_lib. Поддержите в приложении цветные фигуры, доработайте соответствующим образом адаптер холста.

#### Бонус – 30 баллов за unit-тестирование адаптера

Бонус начисляется за unit-тестирование адаптера