# Аннотация

Библиотека классов для создания оконного интерфейса в графическом режиме. – Челябинск: ЮУрГУ, ЕТ-212, 2019. Библиографический список – 2 наим. 1 прил.

В курсовой работе описывается разработка библиотека классов для создания оконного интерфейса в графическом режиме с помощью объектно ориентированного подхода. Работа содержит результаты объектно-ориентированного анализа и проектирования, инструкции по установке и использованию библиотеки.

В результате работы была разработана библиотека классов для создания оконного интерфейса в графическом режиме, код которой приводится в приложении.

Оглавление

[Аннотация 1](#_Toc28264281)

# Введение

**Актуальность темы** – Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.

**Цель работы** – Написать библиотеку для оконного интерфейса в графическом режиме.

**Задачи работы**:

* изучить приёмы объектно-ориентированного анализа;
* научиться разрабатывать программы в объектно-ориентированом стиле;
* овладеть технологиями объектно-ориентированного анализа и проектирования;
* изучить концепции объектно-ориентированного программирования; изучить особенности объектной модели языка программирования C++;
* научиться самостоятельно и творчески использовать знания и полученные практические навыки;
* овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

**Объект работы** – Графический интерфейс Borland (BGI)

**Предмет работы** – применение объектно-ориентированного подхода для разработки библиотеки.

**Результаты работы** можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика»

## Постановка задачи

Необходимо разработать библиотеку классов для оконного интерфейса в графическом режиме. Должны определяться следующие классы:

1. окно
   1. координаты
   2. размеры
   3. видимость
   4. цвет фона
   5. текст заголовка
   6. процедура, вызываемая для нераспознанных клавиш окна
   7. верхнее окно // свойство класса
   8. добавить(элемент) // метод
2. метка (надпись)
   1. координаты
   2. видимость
   3. цвет
   4. текст метки
3. кнопка
   1. координаты
   2. размеры
   3. видимость
   4. цвет кнопки
   5. текст кнопки
   6. цвет текста
   7. процедура, вызываемая при нажатии кнопки
4. ввод строки
   1. координаты
   2. ширина
   3. видимость
   4. цвет поля
   5. цвет текста
   6. текст

В качестве примера был рассмотрен оконный интерфейс Windows XP

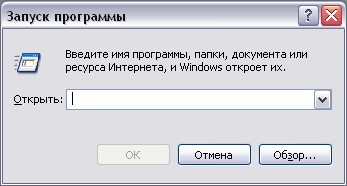


Рисунок пример окна

Анализ предметной области выявил, что объекты интерфейса могут быть двух типов, контейнеры и компоненты. Каждый объект интерфейса имеет такие свойства как видимость, ширина, высота, позиция по x и y относительно родительского элемента, цвет фона, цвет отрисовки.

## Описание программы

2.1 Для разработки программы были использованы:

компилятор MinGW GNU C/C++ 7.2

2.2 Библиотека состоит из 10 модулей:

Модуль Event (интерфейсная часть в файле .hpp, реализация в файле event.cpp) содержит следующие классы

class Event {//Интерфейс метка для стуктур данных событий

public:

virtual ~Event() {};

};

//Событие мыши

class MouseEvent : public Event {

public:

enum Type {

MOVE, DRAG, LEFT\_KEY\_DOWN, LEFT\_KEY\_UP

};

MouseEvent(int, int, Type, int = 0, int = 0);//Конструктор

int getX() const;//Возращает позицию x нажатия мыши

int getY() const;// Возращает позицию y нажатия мыши

int getPreX() const;// Возращает предыдущую позицию x нажатия мыши

int getPreY() const;// Возращает предыдущую позицию y нажатия мыши

Type getType() const;// Возращает тип события мыши

private:

int x, y, preX, preY;//Текущие x,y, предыдущие x,y

Type type;//тип события мыши

};

//Событие клавиатуры

class KeyboardEvent : public Event {

public:

KeyboardEvent(int, int = 0);//Конструктор

int getKeyCode() const;//Возращает код тукущей нажатой клавиши

int getPreKeyCode() const;//Возращает предыдущей нажатой клавишу

private:

int keyCode, preKeyCode;//Код клавиши, код предыдущей нажатой клавиши

};

Модуль Listener (интерфейсная часть в файле .hpp, реализация в файле Listener.cpp) содержит следующие классы

class Listener {//Слушатель

public:

virtual ~Listener(){}//Деструктор

virtual void onEvent(Event \*event){}//Действие при событии

};

class OnClickListener : public Listener {//Слушатель клика мыши

protected:

virtual void onClick(MouseEvent \*event){};//Действие при клике мышкой

private:

void onEvent(Event \*event);//Проверка было ли событие кликом мыши

};

class OnPressListener : public Listener {//Слуушатель зжатия кнопки мыши

protected:

virtual void onPress(MouseEvent \*event){};//Действие при зажатии кнопки мыши

private:

void onEvent(Event \*event);//Проверка было ли событие зажатием кнопки мыши

};

class OnDragListener : public Listener {//Слуушатель перетаскивания зажатой мыши

protected:

virtual void onDrag(MouseEvent \*event){};//Действие при перетаскивании зажатой мыши

private:

void onEvent(Event \*event);//Проверка было ли событие перетаскиванием зажатой мыши

};

class OnMoveListener : public Listener {//Слуушатель передвижения мыши

protected:

virtual void onMove(MouseEvent \*event){};//Действие при передвижении мыши

private:

void onEvent(Event \*event);//Проверка было ли событие передвижением мыши

};

class OnKeyUpListener : public Listener {//Слуушатель событий клавиатуры

protected:

virtual void onKeyUp(KeyboardEvent \*event){};// Действие при событии клавиатуры

private:

void onEvent(Event \*event);//Проверка является ли событие событием клавиатуры

};

Модуль Component (интерфейсная часть в файле .hpp, реализация в файле Component.cpp) содержит следующие классы

class Component { //Компонент

public:

virtual ~Component(); //Деструктор

void addListener(Listener \*listener); //Добавляет слушателя

void notify(Event \*event); //Сообщить слушателям компонента о событии

void render(int rootX, int rootY, int rootWidth, int rootHeight);//Визуализация и обновление абсолютной позиции

virtual void draw(int rootWidth, int rootHeight) {} //Метод отрисовки компонента

//Возвращает…

bool isVisible() const; // true если компонент видимый

int getWidth() const; // ширину компонента

int getHeight() const; // высоту компонента

int getX() const; // позицию x компонента в родительском контейнере

int getY() const; // позицию y компонента в родительском контейнере

int getAbsolutX() const; // абсолютную позицию x

int getAbsolutY() const; // абсолютную позицию y

string getTag() const; // тег элемента

//Устанавливает

void setVisible(bool visible); // Видимость элемента

void setWidth(int width); // Ширину элемента

void setHeight(int height); // Высоту элемента

void setX(int x); // Позицию x компонента в родительском контейнере

void setY(int y); // Позицию y компонента в родительском контейнере

void setPosition(int x, int y);// Позицию x и y компонента в родительском контейнере

void setBgColor(int color);// Цвет фона компонента

void setColor(int color); // Основной цвет отрисовки компонента

void setTag(string tag); // Тег элемента

protected:

string tag = "untag"; // Тег элемента

bool visible = true; //Видимость объекта

int width = 0, height = 0, x = 0, y = 0,absolutX = 0, absolutY = 0;// ширина, высота, позиция компонента относительно родительского контейнера, абсолютная позиция компонента

int bgColor = LIGHTGRAY; //Цвет фона, по умолчанию светло-серый

int color = BLACK; //Основной цвет рисования, по умолчанию чёрный

private:

vector<Listener\*> listeners; //Динамический массив слушателей

};

Модуль Container (интерфейсная часть в файле .hpp, реализация в файле Container.cpp) содержит следующие классы

class Container : public Component {//Контейнер

public:

Container(); //Конструктор

virtual ~Container(); //Деструктор

void addComponent(Component \*component); //Добавить компонент в контейнер

void notifyAll(Event \*event); //Вызывает свой notify, notify компонентов в контейнере и notifyAll контейнеров в контейнере

void setVisible(bool visible);//Устанавливает видимость контейнера и дочерних элементов

void renderAll(int rootX, int rootY, int rootWidth, int rootHeight); //Вызывает свой render, render компонентов в контейнере и renderAll контейнеров в контейнере

vector<Component \*> \*getComponents() const; //Возращает динамический массив компонентов

protected:

void draw(int rootWidth, int rootHeight) {} //Метод отрисовки контейнера

private:

vector<Component \*> \*components; //Динамический массив компонентов контейнера

};

Модуль Screen (интерфейсная часть в файле .hpp, реализация в файле Screen.cpp) содержит следующие классы

class Screen : public Container {//Экран, основа отображения элементов

public:

Screen();//Конструктор

~Screen() {}; //Деструктор

void start();//Инициализация графического интерфейса

private:

void draw(int rootWidth, int rootHeight);//Метод отрисовки

bool run;//Флаг, при false приложение завершает свою работу

void checkMouse();//Проверка событий мыши, уведомляет о них слушателей

void checkKeyboard();//Проверка событий клавиатуры, уведомляет о них слушателей

class ScreenOnKeyUpListener : public OnKeyUpListener {

public:

ScreenOnKeyUpListener(Screen \*screen);//Конструктор

private:

Screen \*screen;//Родительский экран

void onKeyUp(KeyboardEvent \*event);//Реализация завершения работы Screen при нажатии клавиши ESC

};

class ScreenOnSelectListner : public OnClickListener {

public:

ScreenOnSelectListner(Screen \*screen);//Конструктор

private:

Screen \*screen;//Родительский экран

void replaceSelectables(int pos);//Перемещение окна в верх массива

void onClick(MouseEvent \*event);//Выбор текущего активного окна

};

class AboutButton : public Button {

public:

AboutButton(Screen \*screen);//Конструктор

virtual ~AboutButton() {}; //Деструктор

private:

class AboutButtonOnClickListener : public OnClickListener {

public:

AboutButtonOnClickListener(AboutButton \*button,Screen \*screen);//Конструктор

private:

Screen \*screen; //Кореной элемент

AboutButton \*button;//Родительская кнопка

void onClick(MouseEvent \*event); //Открытие окна "о библиотеке"

};

};

};

Модуль Window (интерфейсная часть в файле .hpp, реализация в файле Window.cpp) содержит следующие классы

/\*\*

\*Окно, является контейнером для других компонентов.

\*Может иметь заголовок.

\*/

class Window : public Container {

public:

Window(string title = " ", int width = 250, int height = 200);//Конструктор

virtual ~Window(){};//Деструктор

bool isSelected() const;//Возвращает true, если окно является активным

void setSelected(bool selected);//Устанавливает статус активности окна

string getTitle() const;//возвращает текст заголовка окна

protected:

void draw(int rootWidth, int rootHeight);//отрисовывает окно, унаследовано Container

private:

bool selected = false;//статус активности окна

bool dragedNow = false;//флаг перемещения окна

string title;//текст заголовка окна

class WindowOnDragListener : public OnDragListener { //вложеный класс, реализующий слушатель перетаскивания мышью

public:

WindowOnDragListener(Window \*window);//Конструктор

private:

Window \*window;//окно слушателя

void onDrag(MouseEvent \*event);//перемещение окна

};

class WindowOnClickListener : public OnClickListener { //вложеный класс, реализующий слушатель шелчка левой кнопки мыши

public:

WindowOnClickListener(Window \*window);//Конструктор

private:

Window \*window;//окно слушателя

void onClick(MouseEvent \*event);//снятие флага перемещения

};

};

Модуль Button (интерфейсная часть в файле .hpp, реализация в файле Button.cpp) содержит следующие классы

class Button : public Container {

public:

Button(int x = 0, int y = 0, int width = 100, int height = 25);// Конструктор

virtual ~Button() {}; // Деструктор

void setCheckColor(int color); // Установка цвета устанавливаемого при наведении мыши на кнопку

void setPressedColor(int color); // Установка цвета устанавливаемого при зажатии левой кнопки мыши на кнопке

protected:

void draw(int rootWidth, int rootHeight); // Отрисовка

private:

int checkColor = RGB(220,220,220); //Цвет устанавливаемый при наведении мыши на кнопку, по умолчанию светлосерый

int pressedColor = DARKGRAY; //Цвет устанавливаемый прри зажатии кнопки

enum Stage {

UP, CHECK, DOWN

};

Stage stage = UP;//Стадия кнопки, по умолчанию UP

class onMoveInButtonListener : public OnMoveListener {

public:

onMoveInButtonListener(Button \*button); //Конструктор

private:

Button \*button;//Родительская кнопка

void onMove(MouseEvent \*event); //Установка стадии на CHECK при движении на кнопке

};

class onPressedButtonListener : public OnPressListener {

public:

onPressedButtonListener(Button \*button);//Конструктор

private:

Button \*button;//Родительская кнопка

void onPress(MouseEvent \*event);//Изменение стадии кнопки на DOWN когда кнопка зажата

};

class ButtonOnClickListener : public OnClickListener {

public:

ButtonOnClickListener(Button \*button);//Конструктор

private:

Button \*button;//Родительская кнопка

void onClick(MouseEvent \*event); //Установка стадии на CHECK при клике на кнопке

};

};

class TextButton : public Button{

public:

TextButton(string text = " " ,int x = 0, int y = 0, int width =100, int height = 25);//Конструктор

virtual ~TextButton(){};//Деструктор

string getText() const;//Возвращает текст кнопки

void setText(string text);//Установка текста кнопки

private:

Label \*buttonLabel;//Метка текстовой кнопки

};

Модуль Image (интерфейсная часть в файле .hpp, реализация в файле Image.cpp) содержит следующие классы

/\*\*

\*Метка (текст)

\*/

class Image : public Component {

public:

Image(string path, int x = 0, int y = 0, bool useMask = false);//Конструктор

virtual ~Image();//Деструктор

void draw(int rootWidth, int rootHeight);//Отрисовка изображения

private:

bool useMask;//флаг использования маски (альтернатива альфа канала)

IMAGE \*img, \*mask;//изображение и маска

IMAGE \*createmask(IMAGE \*p);//создание маски

};

Модуль TextInputField (интерфейсная часть в файле .hpp, реализация в файле TextInputField.cpp) содержит следующие классы

/\*

\*Поле ввода текста

\*/

class TextInputField : public Component {

public:

TextInputField(string text = " ", int x = 0, int y = 0);//Конструктор

virtual ~TextInputField() {}; //Деструктор

void setText(string text);//Устанавливает текст поля

string getText() const;//Возвращает текст поля

protected:

void draw(int rootWidth, int rootHeight);//Отрисовка

private:

string text;//Текст поля

class InputOnKeyListener : public OnKeyUpListener {

public:

InputOnKeyListener(TextInputField \*field);//Конструктор

private:

TextInputField \*field;//Родительское поле ввода

void onKeyUp(KeyboardEvent \*event);//Обработка события ввода

};

};