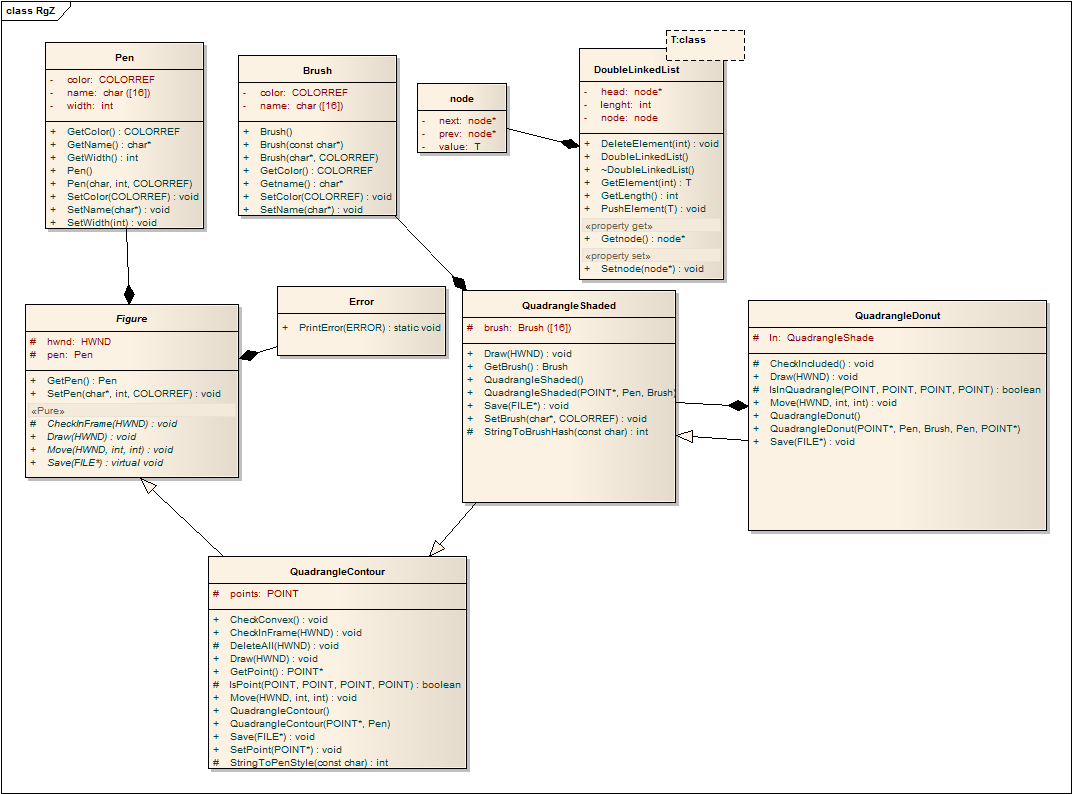
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Расчетно-графическое задание | | |
| по дисциплине «Объектно-ориентрованное программирование» | | |
|  | | |
| **Создание диаграммы классов и прототипа кода на языка C++ в системе “Enterprise architect”** Место для ввода текста. | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-81 |
|  |  |
| Студенты: | Воронкина Дарья |
|  | Кондратьев Игорь |
|  |  |
| Преподаватели: | Лисицин Даниил Валерьевич |
|  | Еланцева Ирина Леонидовна |
|  | | |
| Новосибирск | | |
| 2019 | | |

1. Постановка задачи

Ознакомиться с назначением и элементами диаграммы классов, технологией разработки диаграмм классов и создания прототипа кода классов на языке С++ в системе Enterprise Architect. Составить диаграмму классов, соответствующую графической системе, разработанной в лабораторных работах № 4, 5, и произвести автоматическую генерацию прототипа кода разработанных классов.

1. Диаграмма класса  
   

4. Сгенерированный код

**Исходный код**

|  |
| --- |
| Class Error |
|  | { |
|  | public: |
|  | static void PrintError(ERROR error); |
|  | }; |
|  |  |

**Сгенерированный код**

class Error

{

public:

Error();

virtual ~Error();

static void PrintError(ERROR errror);

};

**Исходный код**

|  |
| --- |
| class Brush |
|  | { |
|  | private: |
|  | char name[16]; |
|  | COLORREF color; |
|  | public: |
|  | Brush(); |
|  | Brush(const char\* name); |
|  | Brush(char\* name, COLORREF color); |
|  | char\* GetName(); |
|  | void SetName(char\* name); |
|  | COLORREF GetColor(); |
|  | void SetColor(COLORREF color); |
|  | }; |

**Сгенерированный код**

public:

virtual ~Brush();

Brush();

Brush(const char\* name);

Brush(char\* name, COLORREF color);

COLORREF GetColor();

char\* Getname();

void SetColor(COLORREF color);

void SetName(char\* name);

private:

COLORREF color;

char name[16];

};

**Исходный код**

|  |
| --- |
| class Pen |
|  | { |
|  | private: |
|  | char name[16]; |
|  | int width; |
|  | COLORREF color; |
|  | public: |
|  | Pen(); |
|  | Pen(char\* name, int width, COLORREF color); |
|  | char\* GetName(); |
|  | void SetName(char\* name); |
|  | int GetWidth(); |
|  | void SetWidth(int width); |
|  | COLORREF GetColor(); |
|  | void SetColor(COLORREF color); |
|  | }; |

**Сгенерированный код**

class Pen

{

public:

virtual ~Pen();

Pen();

Pen(char \*name, int width, COLORREF color);

COLORREF GetColor();

char\* GetName();

int GetWidth();

void SetColor(COLORREF color);

void SetName(char\* name);

void SetWidth(int width);

private:

COLORREF color;

char name[16];

int width;

};

**Исходный код**

|  |
| --- |
| class Figure |
|  | { |
|  | protected: |
|  | Pen pen; |
|  | HWND hwnd; |
|  | virtual void CheckInFrame(HWND hwnd) = 0; |
|  | public: |
|  | virtual void Save(FILE\* saveFile) = 0; |
|  | virtual void Draw(HWND hwnd) = 0; |
|  | virtual void Move(HWND hwnd, int x, int y) = 0; |
|  | Pen GetPen(); |
|  | void SetPen(char\* name, int width, COLORREF color); |
|  | }; |

**Сгенерированный код**

#include "Error.h"

#include "Pen.h"

class Figure

{

public:

Figure();

virtual ~Figure();

Error m\_Error;

Pen m\_Pen;

virtual void Draw(HWND hwnd) =0;

Pen GetPen();

virtual void Move(HWND hwnd, int x, int y) =0;

virtual virtual void Save(FILE\* saveFile) =0;

void SetPen(char\* name, int width, COLORREF color);

protected:

HWND hwnd;

Pen pen;

virtual void CheckInFrame(HWND hwnd) =0;

};

**Исходный код**

|  |
| --- |
| template <class T> |
|  | class DoubleLinkedList |
|  | { |
|  | public: |
|  | DoubleLinkedList(); |
|  | ~DoubleLinkedList(); |
|  |  |
|  | T GetElement(int index); |
|  | void PushElement(T elemnent); |
|  | void DeleteElement(int indexs); |
|  | int GetLength(); |
|  |  |
|  | private: |
|  | struct node |
|  | { |
|  | T value; |
|  | node\* prev; |
|  | node\* next; |
|  | }; |
|  |  |
|  | node\* head; |
|  | int length; |
|  | }; |

**Сгенерированный код**

template<class T>

class DoubleLinkedList

{

public:

node m\_node;

DoubleLinkedList();

~DoubleLinkedList();

void DeleteElement(int indexs);

T GetElement(int index);

int GetLength();

node\* Getnode();

void PushElement(T element);

void Setnode(node\* newVal);

private:

node\* head;

int lenght;

node node;

};

**Исходный код**

|  |
| --- |
| class QuadrangleContour : public Figure |
|  | { |
|  | protected: |
|  | POINT points[4]; |
|  | int StringToPenStyle(const char string[]); |
|  | bool IsPoint(POINT k1, POINT k2, POINT k3, POINT k4); |
|  | void DeleteAll(HWND hwnd); |
|  | public: |
|  | void CheckInFrame(HWND hwnd); |
|  | void CheckConvex(); |
|  | QuadrangleContour(); |
|  | QuadrangleContour(POINT\* points, Pen qd\_pen); |
|  | void Save(FILE \*saveFile); |
|  | void Draw(HWND hwnd); |
|  | void Move(HWND hwnd, int x, int y); |
|  | POINT\* GetPoint(); |
|  | void SetPoint(POINT\* points); |
|  | }; |

**Сгенерированный код**

class QuadrangleContour : public Figure

{

public:

virtual ~QuadrangleContour();

QuadrangleContour();

QuadrangleContour(POINT\* points, Pen qd\_pen);

void CheckConvex();

void CheckInFrame(HWND hwnd);

void Draw(HWND hwnd);

POINT\* GetPoint();

void Move(HWND hwnd, int x, int y);

void Save(FILE\* saveFile);

void SetPoint(POINT\* points);

protected:

POINT points;

void DeleteAll(HWND hwnd);

boolean IsPoint(POINT k1, POINT k2, POINT k3, POINT k4);

int StringToPenStyle(const char string[]);

};

**Исходный код**

|  |
| --- |
| class QuadrangleShaded : public QuadrangleContour |
|  | { |
|  | protected: |
|  | Brush brush; |
|  | int StringToBrushHash(const char string[]); |
|  | public: |
|  | QuadrangleShaded(); |
|  | QuadrangleShaded(POINT\* points, Pen qd\_pen, Brush qd\_brush); |
|  | void Draw(HWND hwnd); |
|  | void Save(FILE\* saveFile); |
|  | Brush GetBrush(); |
|  | void SetBrush(char\* name, COLORREF color); |
|  | }; |

**Сгенерированный код**

class QuadrangleShaded : public QuadrangleContour

{

public:

virtual ~QuadrangleShaded();

Brush m\_Brush;

QuadrangleShaded();

QuadrangleShaded(POINT\* points, Pen qd\_pen, Brush qd\_brush);

void Draw(HWND hwnd);

Brush GetBrush();

void Save(FILE\* saveFile);

void SetBrush(char\* name, COLORREF color);

protected:

Brush brush[16];

int StringToBrushHash(const char string[]);

};

**Исходный код**

|  |
| --- |
| class QuadrangleDonut : public QuadrangleShaded |
|  | { |
|  | protected: |
|  | QuadrangleShaded In; |
|  | void CheckIncluded (); |
|  | bool IsInQuadrangle(POINT P1, POINT P2, POINT P3, POINT P4, POINT PTest); |
|  | public: |
|  | QuadrangleDonut(); |
|  | QuadrangleDonut(POINT\* points, POINT\* pointsIn, Pen qd\_pen, Pen qd\_penIn, Brush qd\_brush); |
|  | void Save(FILE\* saveFile); |
|  | void Draw(HWND hwnd); |
|  | void Move(HWND hwnd, int x, int y); |
|  | }; |

**Сгенерированный код**

class QuadrangleDonut : public QuadrangleShaded

{

public:

virtual ~QuadrangleDonut();

QuadrangleShaded m\_QuadrangleShaded;

QuadrangleDonut();

QuadrangleDonut(POINT\* points, Pen qd\_pen, Brush qd\_brush, Pen qd\_penIn, POINT\* pointsIn);

void Draw(HWND hwnd);

void Move(HWND hwnd, int x, int y);

void Save(FILE\* saveFile);

protected:

QuadrangleShade In;

void CheckIncluded();

boolean IsInQuadrangle(POINT P1, POINT P2, POINT P3, POINT P4);

};

1. Вывод  
   Сгенерированный код совпал с кодом лабораторной работы, что говорит о том, что диаграмма классов построена верно.