Báo cáo đồ án 2

Đồ án – SimplePaint

Môn: Phương pháp lập trình hướng đối tượng

Nhóm 4 thực hành hướng đối tượng ca 1 sáng thứ 2

Họ Và Tên: Nguyễn Tuấn Khoa

Mssv: 19120547

Lớp: 19_3

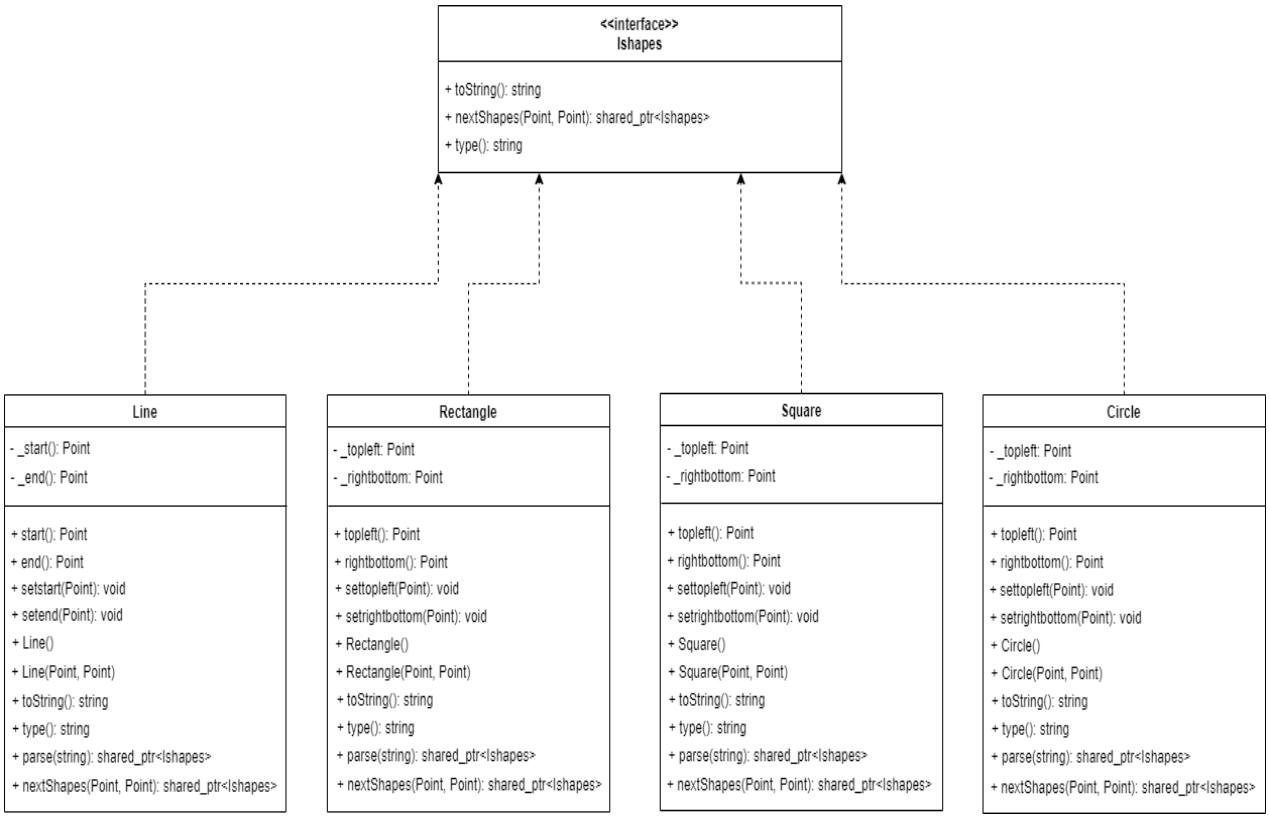
I. Phần trăm công việc

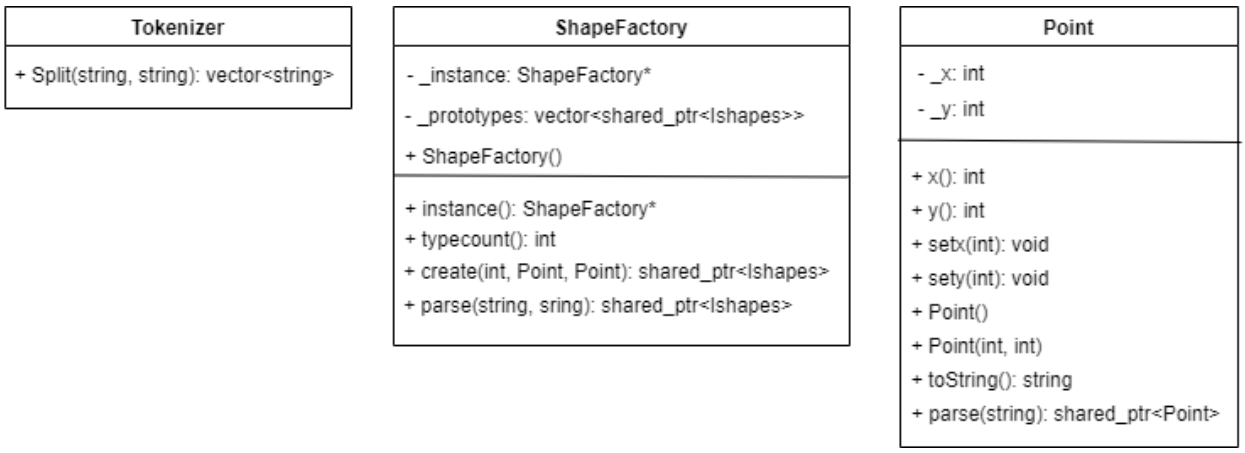
MSSV	Họ và Tên	Phần trăm
19120547	Nguyễn Tuấn Khoa	100%

II. Chức năng hoàn thành

Tạo các nút bấm cho phép chọn các đối tượng hình để vẽ bao gồm (đường thẳng, hình chữ nhật, hình tròn, hình vuông)
Tạo nút lưu cho phép người sử dụng lưu các hình đã vẽ xuống file txt
Tạo nút mở file để nạp các hình được lưu trong file lên màn hình
Tạo nút new file để tạo file mới cho phép người sử dụng thao tác trên file mới đó

III. Sơ đồ lớp UML các đối tượng cần vẽ





IV. Các hàm đã cài đặt

- **Các hàm quan trọng trong StaticLib OopLibShapes**

+ Các hàm lớp Tokenizer

- **Vector<string>Split(string haystack, string needdle):** chia chuỗi cha thành các tokens dựa trên chuỗi ngăn cách needdle

```
std::vector<std::string>Tokenizer::Split(std::string haystack, std::string needdle =
{
    std::vector<std::string>tokens;
    int startpos = 0;
    size_t  foundpos = 0;
    while (true){
        foundpos = haystack.find(needdle, startpos);
        if (foundpos != std::string::npos)
        {
            std::string token = haystack.substr(startpos, foundpos - startpos);
            tokens.push_back(token);
            startpos = (int)foundpos + (int)needdle.length();
        }
        else
        {
            std::string token = haystack.substr(startpos, haystack.length() - found
            tokens.push_back(token);
            break;
        }
    }
    return tokens;
```

+ Các hàm của lớp Point

- **Shared_ptr<Point>parse(string buffer):** chuyển chuỗi lưu tọa độ hình sang kiểu dữ liệu tương ứng và gán cho từng điểm

```
std::shared_ptr<Point> Point::parse(std::string buffer)
{
    std::vector<std::string>tokens = Tokenizer::Split(buffer, ",");
    int x = stoi(tokens[0]);
    int y = stoi(tokens[1]);
    return std::make_shared<Point>(x, y);
}
```

+ Các hàm của các lớp đối tượng hình học

- **String type():** trả ra loại hình mà người dùng đã chọn để vẽ
- **Shared_ptr<Ishapes>nextShapes(Point topleft, Point rightbottom):** tạo ra một đối tượng hình học cụ thể hình học với 2 điểm cho trước
- **Shared_ptr<Ishapes>parse(string buffer):** chuyển chuỗi buffer lưu các thông tin của hình học bao gồm loại hình và tọa độ sang các kiểu dữ liệu tương ứng và gán cho đối tượng
- **Ví dụ lớp hình chữ nhật**

```
std::shared_ptr<Ishapes> Rectangle::parse(std::string buffer)
{
    std::vector<std::string>tokens = Tokenizer::Split(buffer, " ");
    std::shared_ptr<Point> a = Point::parse(tokens[0]);
    std::shared_ptr<Point> b = Point::parse(tokens[1]);
    return std::make_shared<Rectangle>(*a, *b);
}
std::shared_ptr<Ishapes> Rectangle::nextShapes(Point topleft, Point rightbottom)
{
    return std::make_shared<Rectangle>(topleft, rightbottom);
}
std::string Rectangle::type()
{
    return { "Rectangle" };
}
```

- **Riêng constuctor của đối tượng hình vuông Square(Point a, Point b) cần có điều kiện để chiều dài = chiều rộng như sau:**

```
int d0 = abs(a.x() - b.x());
int d1 = abs(a.y() - b.y());
int d = abs(d0 - d1);
if (d0 < d1) {
    if (a.y() > b.y()) {
        b.sety(b.y() + d);
    }
    else {
        b.sety(b.y() - d);
    }
}
else {
    if (a.x() > b.x()) {
        b.setx(b.x() + d);
    }
    else {
        b.setx(b.x() - d);
    }
}
_topleft = a;
_rightbottom = b;
```

+ Các hàm của lớp ShapeFactory

- **ShapeFactory():** cho biết các đối tượng muốn vẽ bằng cách lưu chúng vào vector<Ishapes>_prototypes để dễ dàng truy cập khi tạo ra đối tượng bất kí hay nạp các hình để vẽ từ file txt lên màn hình
- **Typecount():** trả ra size của _prototypes hay trả ra số lượng đối tượng hình học đã cài đặt
- **Shared_ptr<Ishapes>create(int type, Point StartPoint, Point EndPoint):** tạo một đối tượng hình học dựa trên loại hình học muốn tạo và hai điểm có sẵn truyền vào, int type ở đây là thứ tự đối tượng nằm trong vector _prototypes

```
ShapeFactory::ShapeFactory()
{
    _prototypes.push_back(std::make_shared<Line>());
    _prototypes.push_back(std::make_shared<Rectangle>());
    _prototypes.push_back(std::make_shared<Circle>());
    _prototypes.push_back(std::make_shared<Square>());
}
int ShapeFactory::typecount()
{
    return (int)_prototypes.size();
}
std::shared_ptr<Ishapes>ShapeFactory::create(int type, Point start, Point end)
{
    return _prototypes[type]->nextShapes(start, end);
}
```

- **Instance():** tạo và trả ra con trỏ ShapeFactory* _instance

```
ShapeFactory* ShapeFactory::instance()
{
    if (_instance == NULL) {
        _instance = new ShapeFactory();
    }
    return _instance;
}
```

- **Shared_ptr<Ishapes>parse(string type, string buffer):** chuyển chuỗi buffer chứa các thông tin của hình sang các kiểu dữ liệu tương ứng với loại hình được truyền vào

```
std::shared_ptr<Ishapes> ShapeFactory::parse(std::string type, std::string value)
{
    std::shared_ptr<Ishapes>shape = NULL;
    if (type == "Line") {
        shape = Line::parse(value);
    }
    else if (type == "Rectangle") {
        shape = Rectangle::parse(value);
    }
    else if (type == "Square") {
        shape = Square::parse(value);
    }
    else {
        shape = Circle::parse(value);
    }
    return shape;
}
```

- **Các biến toàn cục quan trọng trong ProjectSimplePaint**

```
HWND txtfile;// một handle của cửa sổ con và gián tiếp đến các lớp cửa sổ con khi new hoặc load file
HWND hWndWinmain;// một handle của cửa sổ mẹ

//Biến toàn cục xác định hình sẽ thao tác
int LINE = 0;
int RECTANGLE = 1;
int CIRCLE = 2;
int SQUARE = 3;

//Biến toàn cục xử lý vẽ, nạp, tải hình
Point start; // điểm nhấp chuột
Point End; //điểm nhả chuột
bool isPreview;// có nhấn chuột hay không
bool isDrawing;// có đang vẽ hay không
int TypeOfShape = 0; // loại hình
int Loadfile = 0;// xác định có load file hay ko
vector<shared_ptr<Ishapes>>shapes;//lưu các hình đã vẽ
vector<DWORD>saveRGB;// lưu lại màu viền của hình
shared_ptr<Ishapes>shape = NULL;// lưu 1 hình hiện tại
```

```
//Biến toàn cục để xử lí newfile và loadfile
int fileAction = 0; // trạng thái file load hay new file
vector<string>saveFilename = { "filename.txt", "data.txt" };// vector lưu những tên file tồn tại
string currentFile = "data.txt";// tên file lưu mặc định nếu không newfile
const int BUFFER_SIZE = 255;
WCHAR buffer[BUFFER_SIZE]; // chứa chuỗi lấy từ hàm GetWindowsText của hệ thống
```

- **Các hàm quan trọng trong Project SimplePaint**

+ Các hàm quan trọng của cửa sổ mẹ

- **ATOM MyRegisterClass(HINSTANCE hinstance):** đăng kí hay khởi tạo lớp cửa sổ chính

```
ATOM MyRegisterClass(HINSTANCE hInstance)
{
    WNDCLASSEXW wcex;

    wcex.cbSize = sizeof(WNDCLASSEX);

    wcex.style          = CS_HREDRAW | CS_VREDRAW;
    wcex.lpfnWndProc    = WndProc;
    wcex.cbClsExtra     = 0;
    wcex.cbWndExtra     = 0;
    wcex.hInstance      = hInstance;
    wcex.hIcon          = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));
    wcex.hCursor        = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground  = (HBRUSH)(CTLCOLOR_BTN + 1);
    wcex.lpszMenuName   = MAKEINTRESOURCEW(IDC_SIMPLEPAINT);
    wcex.lpszClassName  = szWindowClass;
    wcex.hIconSm        = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_ICON2));

    return RegisterClassExW(&wcex);
}
```

- **BOOL InitInstance(HINSTANCE hinstance, int CmdShow**): lưu các trường hợp xử lý và tạo cửa sổ chính

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
   hInst = hInstance; // Store instance handle in our global variable

   hWndWinmain = CreateWindowW(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
      CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

   if (!hWndWinmain)
   {
      return FALSE;
   }

   ShowWindow(hWndWinmain, nCmdShow);
   UpdateWindow(hWndWinmain);

   return TRUE;
}
```

- **LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM):** tạo quy trình nhận lệnh từ WM cho cửa sổ chính

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        HANDLE_MSG(hWnd, WM_CREATE, OnCreate);
        HANDLE_MSG(hWnd, WM_COMMAND, OnCommand);
        HANDLE_MSG(hWnd, WM_DESTROY, OnDestroy);
        HANDLE_MSG(hWnd, WM_PAINT, OnPaint);
        HANDLE_MSG(hWnd, WM_LBUTTONDOWN, OnLButtonDown);
        HANDLE_MSG(hWnd, WM_LBUTTONUP, OnLButtonUp);
        HANDLE_MSG(hWnd, WM_MOUSEMOVE, OnMouseMove);
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

- **INT_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM):** thông báo xử lý cho hộp thoại

```
INT_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    UNREFERENCED_PARAMETER(lParam);
    switch (message)
    {
    case WM_INITDIALOG:
        return (INT_PTR)TRUE;

    case WM_COMMAND:
        if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)
        {
            EndDialog(hDlg, LOWORD(wParam));
            return (INT_PTR)TRUE;
        }
        break;
    }
    return (INT_PTR)FALSE;
}
```

- **BOOL OnCreate(HWND hwnd, LPCREATESTRUCT lpCreateStruct):** Tạo giao diện, nút bấm cho cửa sổ chính

```
BOOL OnCreate(HWND hwnd, LPCREATESTRUCT lpCreateStruct)
{
    InitCommonControls();
    TBBUTTON tbButtons[] =
    {
        { STD_FILENEW,  ID_FILE_NEW, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
        { STD_FILEOPEN, ID_FILE_LOAD, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
        { STD_FILESAVE, ID_FILE_SAVE, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 }
    };
    HWND hToolBarWnd = CreateToolbarEx(hwnd,
        WS_CHILD | WS_VISIBLE | CCS_ADJUSTABLE | TBSTYLE_TOOLTIPS,
        ID_TOOLBAR, sizeof(tbButtons) / sizeof(TBBUTTON), HINST_COMMCTRL,
        0, tbButtons, sizeof(tbButtons) / sizeof(TBBUTTON),
        BUTTON_WIDTH, BUTTON_HEIGHT, IMAGE_WIDTH, IMAGE_HEIGHT,
        sizeof(TBBUTTON));
```

```
TBBUTTON userButtons[] =
{
    { 0, 0, TBSTATE_ENABLED, TBSTYLE_SEP, 0, 0 },
    { 0, ID_DRAW_CIRCLE, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
    { 1, ID_DRAW_RECTANGLE, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
    { 2, ID_DRAW_LINE, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
    { 3, ID_DRAW_SQUARE, TBSTATE_ENABLED, TBSTYLE_BUTTON, 0, 0 },
};
TBADDBITMAP tbBitmap[] = { {hInst, IDB_BITMAP1} };
int idx = (int)SendMessage(hToolBarWnd, TB_ADDBITMAP, (WPARAM)sizeof(tbBitmap) / sizeof(TBADDBITMAP),
    (LPARAM)(LPTBADDBITMAP)&tbBitmap);
// Xác định chỉ mục hình ảnh của mỗi button từ ảnh bự liên hoàn nhiều tấm
userButtons[1].iBitmap += idx;
userButtons[2].iBitmap += idx;
userButtons[3].iBitmap += idx;
userButtons[4].iBitmap += idx;
// Thêm nút bấm vào toolbar
SendMessage(hToolBarWnd, TB_ADDBUTTONS, (WPARAM)sizeof(userButtons) / sizeof(TBBUTTON),
    (LPARAM)(LPTBBUTTON)&userButtons);
return TRUE;
```

- **void OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify):** xử lý sự kiện sau khi người dùng kích chuột vào các nút ở cửa sổ mẹ

```
void OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{
    switch (id)
    {
    case ID_CHOOSE_COLOR:
    {
        ZeroMemory(&cc, sizeof(CHOOSECOLOR));
        cc.lStructSize = sizeof(CHOOSECOLOR);
        cc.hwndOwner = hwnd;
        cc.lpCustColors = (LPDWORD)acrCustClr;
        cc.rgbResult = rgbCurrent;
        cc.Flags = CC_FULLOPEN | CC_RGBINIT;
        if (ChooseColor(&cc))
        {
            hbrush = CreateSolidBrush(cc.rgbResult);
            rgbCurrent = cc.rgbResult;
        }
        break;
    }
```

```
case ID_FILE_NEW:
{
    fileAction = ID_FILE_NEW;
    displayDialog(hwnd);
    break;
}
case ID_FILE_LOAD:
{
    fileAction = ID_FILE_LOAD;
    displayDialogLoad(hwnd);
    break;
}
```

```
case ID_FILE_SAVE:
{
    int value = MessageBox(hwnd, L"Chọn Yes để lưu", L"Bạn có muốn lưu file", MB_YESNO | MB_ICONQUESTION);
    if (IDYES == value)
    {
        WCHAR bufferIn[100];
        bool f_Save = saveShape(currentFile, shapes, saveRGB);
        if (f_Save) {
            wsprintf(bufferIn, L"Lưu file thành công");
        }
        else {
            wsprintf(bufferIn, L"Lưu file thất bại");
        }
        MessageBox(hwnd, bufferIn, L"Lưu file", MB_OK);
    }
    else if (IDNO == value)
    {
        //do nothing
    }
    break;
}
```

```
case ID_DRAW_LINE:
{
    TypeOfShape = ID_DRAW_LINE;
    break;
}
case ID_DRAW_RECTANGLE:
{
    TypeOfShape = ID_DRAW_RECTANGLE;
    break;
}
case ID_DRAW_CIRCLE:
{
    TypeOfShape = ID_DRAW_CIRCLE;
    break;
}
case ID_DRAW_SQUARE:
{
    TypeOfShape = ID_DRAW_SQUARE;
    break;
}
```

- **void OnDestroy(HWND hwnd):** hủy cửa sổ

```
void OnDestroy(HWND hwnd)
{
    PostQuitMessage(0);
}
```

- **void OnPaint(HWND hwnd):** xử lý vẽ hình

```
void OnPaint(HWND hwnd)
{
    PAINTSTRUCT ps;
    hdc = BeginPaint(hwnd, &ps);

    HPEN hPen = CreatePen(PS_DASHDOT, 3, rgbCurrent);
    SelectObject(hdc, hPen);
    if (IDC_LOADFILE == Loadfile)
    {
        for (int i = 0; i < (int)shapes.size(); i++)
        {
            HPEN hPenLoad = CreatePen(PS_DASHDOT, 3, saveRGB[i]);
            SelectObject(hdc, hPenLoad);
            string type = shapes[i]->type();
            string ShapesInformation = shapes[i]->toString();
            vector<string>tokens = Tokenizer::Split(ShapesInformation, " ");
            Point startPoint = getPointOfShapes(tokens[0]);
            Point endPoint = getPointOfShapes(tokens[1]);
```

```
if (type == "Line")
{
    MoveToEx(hdc, startPoint.x(), startPoint.y(), NULL);
    LineTo(hdc, endPoint.x(), endPoint.y());
}
else if (type == "Rectangle")
{
    //vẽ hình chữ nhật từ file
    Rectangle(hdc, startPoint.x(), startPoint.y(), endPoint.x(), endPoint.y());
}
else if (type == "Square")
{
    //vẽ hình vuông từ file
    Square(hdc, startPoint.x(), startPoint.y(), endPoint.x(), endPoint.y());
}
else
{
    //vẽ hình tròn từ file
    Circle(hdc, startPoint.x(), startPoint.y(), endPoint.x(), endPoint.y());
}
```

```
else
{
    switch (TypeOfShape)
    {
    case ID_DRAW_LINE:
    {
        shape = ShapeFactory::instance()->create(LINE, start, End);
        MoveToEx(hdc, start.x(), start.y(), NULL);
        LineTo(hdc, End.x(), End.y());
        break;
    }
    case ID_DRAW_RECTANGLE:
    {
        shape = ShapeFactory::instance()->create(RECTANGLE, start, End);
        Rectangle(hdc, start.x(), start.y(), End.x(), End.y());
        break;
    }
       case ID_DRAW_CIRCLE:
       {
           shape = ShapeFactory::instance()->create(CIRCLE, start, End);
           Circle(hdc, start.x(), start.y(), End.x(), End.y());
           break;
       }
       case ID_DRAW_SQUARE:
       {
           shape = ShapeFactory::instance()->create(SQUARE, start, End);
           Square(hdc, start.x(), start.y(), End.x(), End.y());
           break;
       }
       }
    }
    EndPaint(hwnd, &ps);
}
```

- **void OnLButtonDown(HWND hwnd, BOOL fDoubleClick, int x, int y, UINT keyFlags):** xử lý nhấn chuột để vẽ

```
void OnLButtonDown(HWND hwnd, BOOL fDoubleClick, int x, int y, UINT keyFlags)
{
    isPreview = true;
    isDrawing = true;
    start = Point(x, y);
}
```

- **void OnLButtonUp(HWND hwnd, int x, int y, UINT keyFlags):** xử lý nhả chuột sau khi vẽ

```
void OnLButtonUp(HWND hwnd, int x, int y, UINT keyFlags)
{
    shapes.push_back(shape);
    saveRGB.push_back(rgbCurrent);
    isDrawing = false;
    // Báo hiệu cần xóa đi toàn bộ màn hình & vẽ lại
    InvalidateRect(hwnd, NULL, TRUE);
}
```

- **void OnMouseMove(HWND hwnd, int x, int y, UINT keyFlags):** xử lý di chuyển chuột khi vẽ

```
void OnMouseMove(HWND hwnd, int x, int y, UINT keyFlags)
{
    if (isDrawing) {
        End = Point(x, y);

        // Báo hiệu cần xóa đi toàn bộ màn hình
        InvalidateRect(hwnd, NULL, TRUE);
    }
}
```

+ Các hàm quan trọng của cửa sổ con (dialog box)

- **void registerDialogClass(HINSTANCE hinst)**: đăng kí hay khởi tạo lớp cửa sổ con

```
void registerDialogClass(HINSTANCE hinst)
{
    WNDCLASSW dialog = { 0 };

    dialog.hInstance = hinst;
    dialog.hCursor = LoadCursor(nullptr, IDC_CROSS);
    dialog.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    dialog.lpszClassName = L"myDialogClass";
    dialog.lpfnWndProc = dialogWndProc;
    dialog.hIcon = LoadIcon(dialog.hInstance, MAKEINTRESOURCE(IDI_SMALL));

    RegisterClassW(&dialog);
}
```

- **LRESULT CALLBACK dialogWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam):** tạo quy trình nhận lệnh từ WM cho cửa sổ con

```
LRESULT CALLBACK dialogWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        HANDLE_MSG(hWnd, WM_CREATE, OnCreateDialog);
        HANDLE_MSG(hWnd, WM_COMMAND, OnCommandDialog);
        HANDLE_MSG(hWnd, WM_CLOSE, OnCloseDialog);
    default:
        return DefWindowProc(hWnd, message, wParam, lParam);
    }
    return 0;
}
```

- **void displayDialog(HWND hwnd):** tạo ra cửa sổ con khi nhấn newfile

```
void displayDialog(HWND hwnd)
{
    HWND hw = CreateWindowW(L"myDialogClass", L"New File", WS_VISIBLE | WS_OVERLAPPEDWINDOW,
        500, 200, 400, 200, hwnd, nullptr, nullptr, nullptr);
    EnableWindow(hwnd, false);
}
```

- **void displayDialogLoad(HWND hwnd):** tạo ra cửa sổ con khi nhấn loadfile

```
void displayDialogLoad(HWND hwnd)
{
    HWND hw = CreateWindowW(L"myDialogClass", L"Load File", WS_VISIBLE | WS_OVERLAPPEDWINDOW,
        500, 200, 400, 200, hwnd, nullptr, nullptr, nullptr);
    EnableWindow(hwnd, false);
}
```

- **BOOL OnCreateDialog(HWND hwnd, LPCREATESTRUCT lpCreateStruct):** tạo giao diện, nút bấm cho cửa sổ con

```
BOOL OnCreateDialog(HWND hwnd, LPCREATESTRUCT lpCreateStruct)
{
    GetObject(GetStockObject(DEFAULT_GUI_FONT), sizeof(LOGFONT), &lf);
    HFONT hFont = CreateFont(lf.lfHeight, lf.lfWidth,
        lf.lfEscapement, lf.lfOrientation, lf.lfWeight,
        lf.lfItalic, lf.lfUnderline, lf.lfStrikeOut, lf.lfCharSet,
        lf.lfOutPrecision, lf.lfClipPrecision, lf.lfQuality,
        lf.lfPitchAndFamily, lf.lfFaceName);
```

```
if (ID_FILE_NEW == fileAction)
{
    HWND lbl = CreateWindowEx(NULL, L"BUTTON", L"Nhập tên file.txt(không dấu)",
        WS_CHILD | WS_VISIBLE | BS_GROUPBOX,
        10, 30, 365, 70, hwnd,
        NULL, NULL, NULL);
    SetWindowFont(lbl, hFont, TRUE);

    txtfile = CreateWindowEx(WS_EX_CLIENTEDGE, L"EDIT", NULL,
        WS_CHILD | WS_VISIBLE | ES_AUTOHSCROLL,
        20, 58, 330, 25, hwnd,
        NULL, NULL, NULL);
    SetWindowFont(txtfile, hFont, TRUE);

    HWND btnAdd = CreateWindowEx(NULL, L"BUTTON", L"Thêm file",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        150, 90, 75, 30, hwnd, (HMENU)IDC_ADDFILE, NULL, NULL);
    SetWindowFont(btnAdd, hFont, TRUE);
}
else if (ID_FILE_LOAD == fileAction)
{
    HWND lblload = CreateWindowEx(NULL, L"BUTTON", L"Nhập tên file.txt(không dấu)",
        WS_CHILD | WS_VISIBLE | BS_GROUPBOX,
        10, 30, 365, 70, hwnd,
        NULL, NULL, NULL);
    SetWindowFont(lblload, hFont, TRUE);

    txtfile = CreateWindowEx(WS_EX_CLIENTEDGE, L"EDIT", NULL,
        WS_CHILD | WS_VISIBLE | ES_AUTOHSCROLL,
        20, 58, 330, 25, hwnd,
        NULL, NULL, NULL);
    SetWindowFont(txtfile, hFont, TRUE);

    HWND btnLoad = CreateWindowEx(NULL, L"BUTTON", L"Mở file",
        WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
        150, 90, 75, 30, hwnd, (HMENU)IDC_LOADFILE, NULL, NULL);
    SetWindowFont(btnLoad, hFont, TRUE);
}
fileAction = 0;
return TRUE;
```

- **void OnCommandDialog(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify):** xử lý sự kiện sau khi nhấn nút cho cửa sổ con

```
void OnCommandDialog(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{
    WCHAR bufferinform[100];
    string tempfile = "";
    readfilename("filename.txt", saveFilename);
 if (IDC_ADDFILE == id)
 {
     GetWindowText(txtfile, buffer, BUFFER_SIZE);
     tempfile = ws2s(buffer);
     if (!checkfilename(tempfile, saveFilename) || tempfile == "")
     {
         wsprintf(bufferinform, L"Tên file đã tồn tại(nhập lại)");
         MessageBox(hwnd, bufferinform, L"Lỗi", MB_OK | MB_ICONWARNING);
     }
     else
     {
         wsprintf(bufferinform, L"Tạo file thành công");
         currentFile = tempfile;
         saveFilename.push_back(currentFile);
         writefilename("filename.txt", saveFilename);
         if ((int)shapes.size() > 0 || (int)saveRGB.size() > 0)
         {
             shapes.clear();
             saveRGB.clear();
         }
         rgbCurrent = RGB(0, 0, 0);
```

```
            rgbCurrent = RGB(0, 0, 0);
            MessageBox(hwnd, bufferinform, L"Tạo file mới", MB_OK);
        }
    }
    else if (IDC_LOADFILE == id)
    {
        GetWindowText(txtfile, buffer, BUFFER_SIZE);
        tempfile = ws2s(buffer);
        if (checkfilename(tempfile, saveFilename) || tempfile == "")
        {
            wsprintf(bufferinform, L"Không tìm thấy tên file(nhập lại)");
            MessageBox(hwnd, bufferinform, L"Lỗi", MB_OK | MB_ICONWARNING
        }
        else if (tempfile == "filename.txt")
        {
            wsprintf(bufferinform, L"Không được truy cập file này(nhập lạ
            MessageBox(hwnd, bufferinform, L"Lỗi", MB_OK | MB_ICONERROR);
        }
        else
        {
            currentFile = tempfile;
```

```
            currentFile = tempfile;
            if ((int)shapes.size() > 0 || (int)saveRGB.size() > 0)
            {
                shapes.clear();
                saveRGB.clear();
            }
            bool f_Load = loadShape(currentFile, shapes, saveRGB);
            if (f_Load){
                wsprintf(bufferinform, L"Tải file thành công");
                Loadfile = IDC_LOADFILE;
                rgbCurrent = RGB(0, 0, 0);
                InvalidateRect(hWndWinmain, NULL, TRUE);
            }
            else{
                wsprintf(bufferinform, L"Tải file thất bại");
            }
            MessageBox(hwnd, bufferinform, L"Tải file", MB_OK);
        }
    }
    EnableWindow(hWndWinmain, true);
}
```

- **void OnCloseDialog(HWND hwnd):** đóng cửa sổ con

```
void OnCloseDialog(HWND hwnd)
{
    DestroyWindow(hwnd);
    EnableWindow(hWndWinmain, true);
}
```

+ Các hàm phụ trợ

- **bool saveShape(string filename, const vector<shared_ptr<Ishapes>>&, const vector<DWORD>&):** lưu các hình đã vẽ bao gồm loại hình, tọa độ, màu của hình đó xuống file txt

```
bool saveShape(string filename, const vector<shared_ptr<Ishapes>>& arrShapes, const vector<DWORD>& SAVERGB)
{
    if ((int)arrShapes.size() == 0)return false;
    ofstream writter;
    writter.open(filename);
    writter << (int)arrShapes.size() << endl;
    for (int i = 0; i < arrShapes.size(); i++)
    {
        writter << arrShapes[i]->type() << ": ";
        writter << arrShapes[i]->toString() << ": ";
        writter << saveRGB[i] << endl;
    }
    writter.close();
    return true;
}
```

- **bool loadShape(string filename, vector<shared_ptr<Ishapes>>&, vector<DWORD>&):** nạp các hình đã vẽ lên màn hình từ file txt

```
bool loadShape(string filename, vector<shared_ptr<Ishapes>>& arrShapes, vector<DWORD>& LOADRGB)
{
    ifstream reader;
    string lines;
    vector<string>tokens;
    reader.open(filename);
    getline(reader, lines);
    int size = stoi(lines);
    for (int i = 0; i < size; i++)
    {
        getline(reader, lines);
        tokens = Tokenizer::Split(lines, ": ");
        shape = ShapeFactory::parse(tokens[0], tokens[1]);
        DWORD rgbShape = (DWORD)stoi(tokens[2]);
        arrShapes.push_back(shape);
        LOADRGB.push_back(rgbShape);
    }
    reader.close();
    return true;
}
```

- **Point getPointOfShapes(string):** lấy tọa độ của hình

```
Point getPointOfShapes(string token)
{
    shared_ptr<Point>temp = Point::parse(token);
    Point result = *temp;
    return result;
}
```

- **string ws2s(const wstring& s):** hàm chuyển WCHAR sang string để lấy tên file qua từ windowstext

```
string ws2s(const wstring& s)
{
    int len = 0;
    int slength = (int)s.length();
    len = WideCharToMultiByte(CP_UTF8, 0, s.c_str(), slength, 0, 0, 0, 0);
    string r(len, '\0');
    WideCharToMultiByte(CP_UTF8, 0, s.c_str(), slength, &r[0], len, 0, 0);
    return r;
}
```

- **void readfilename(string filename, vector<string>& save):** đọc những tên file đã tạo

```
void readfilename(string filename, vector<string>& save)
{
    if ((int)save.size() > 0) {
        save.clear();
    }
    ifstream reader;
    string line;
    reader.open(filename);
    getline(reader, line);
    int count = stoi(line);
    for (int i = 0; i < count; i++) {
        getline(reader, line);
        save.push_back(line);
    }
    reader.close();
}
```

- **void writefilename(string filename, vector<string>& save):** lưu tên file đã tạo xuống file txt

```
void writefilename(string filename, vector<string>& save)
{
    ofstream writter;
    writter.open(filename);
    writter << save.size() << endl;
    for (int i = 0; i < (int)save.size(); i++)
    {
        writter << save[i] << endl;
    }
    writter.close();
}
```

- **bool checkfilename(string filename, vector<string>& save):** kiểm tra tên đã tồn tại hay chưa

```
bool checkfilename(string filename, vector<string>& save)
{
    for (int i = 0; i < (int)save.size(); i++) {
        if (filename == save[i])
            return false;
    }
    return true;
}
```