Лабораторна робота №5

АБСТРАКТНІ КЛАСИ, ІНТЕРФЕЙСИ,СЕРІАЛІЗАЦІЯ.

***Мета роботи***: Навчитись застосовувати інтерфейси для роботи класів на прикладі задач серіалізації.

**1 ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Реалізувати збереження і завантаження даних за допомогою класу CFileStorage.У висновках відмітити недоліки класу CFileStorage.

**2 ОПИС ПРОГРАМИ**

**2.1 Ієрархія та структура класів**

**class button –** клас, який містить атрибути **data\_field1 data\_field2.**

**class View –** клас, який містить метод, що виводить значення полів **data\_field1 data\_field2**.

**class View2** – клас, який мітить методи info() ,Square() та атрибути **name, surname, num**.

**class BaseView -** клас, який виводить на екран **data\_field1 data\_field2.**

**class View2 -** клас, який виводить на екран об’єкт класу **Jar**

**class CFileStorage –** клас, який серіалізує дані програми

**class** MStorageInterface – інтерфейс для роботи з серіалізацією.

**2.2 Опис програми**

На рисунку 2.1 наведена структура розробленої програми

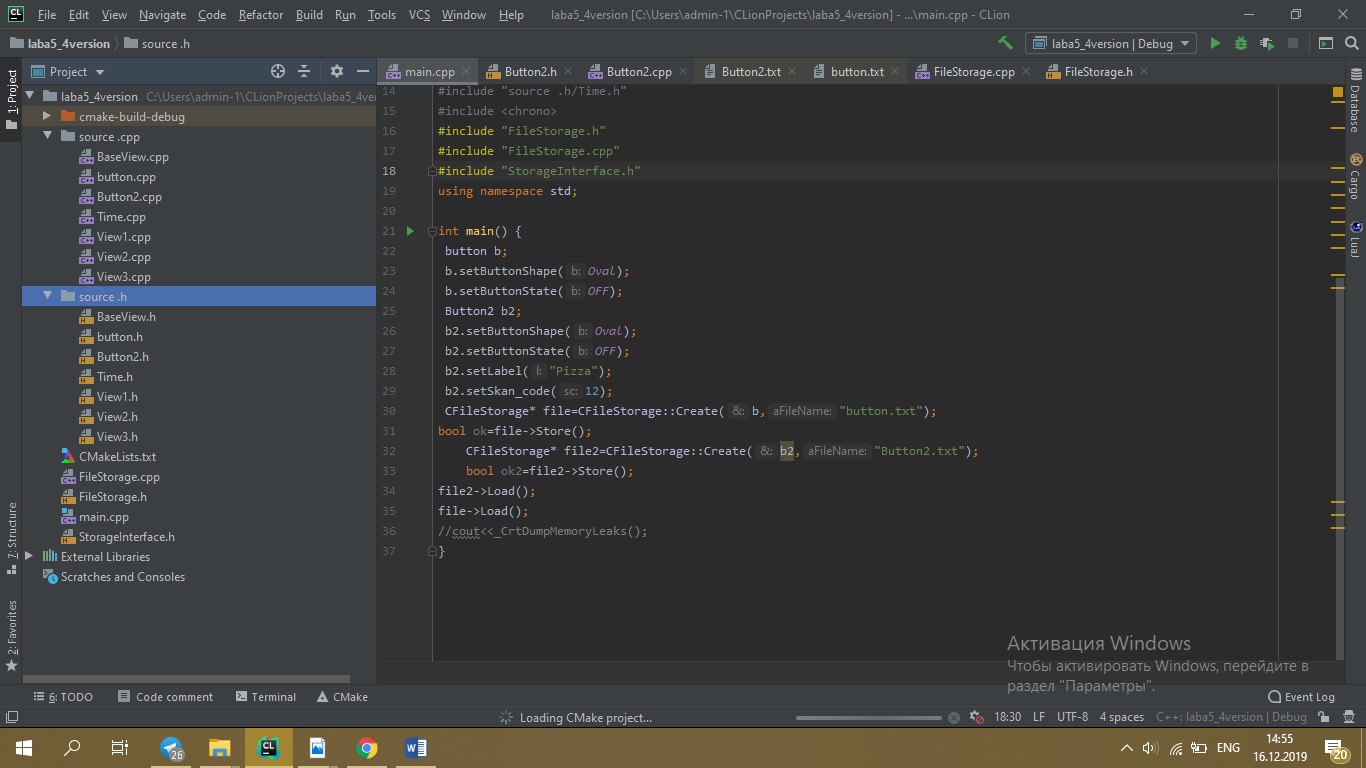


Рис 2.1 – структура програми

**2.3 Важливі фрагменти програми**

2.3.1 Файл*main.cpp*

#include "source .cpp/Time.cpp"  
#include "source .h/Time.h"  
#include <chrono>  
#include "FileStorage.h"  
#include "FileStorage.cpp"  
#include "StorageInterface.h"  
using namespace std;  
  
int main() {  
 button b;  
 b.setButtonShape(*Oval*);  
 b.setButtonState(*OFF*);  
 Button2 b2;  
 b2.setButtonShape(*Oval*);  
 b2.setButtonState(*OFF*);  
 b2.setLabel("Pizza");  
 b2.setSkan\_code(12);  
 CFileStorage\* file=CFileStorage::Create(b,"button.txt");  
bool ok=file->Store();  
 CFileStorage\* file2=CFileStorage::Create(b2,"Button2.txt");  
 bool ok2=file2->Store();  
file2->Load();  
file->Load();  
//cout<<\_CrtDumpMemoryLeaks();  
}

2.3.1 Файл*button.cpp*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
//  
// Created by admin-1 on 06.09.2019.  
//  
#pragma once  
#include "../source .h/button.h"  
#include <iostream>  
#include <fstream>  
 void button::setButtonState(ButtonState b) {  
 bt = b;  
 }  
  
  
  
  
bool button::IsRoundPressed() {  
 return getButtonState() == 0;  
}  
  
void button::setButtonShape(ButtonShape b) {  
bs=b;  
}  
  
ButtonShape button::getButtonShape() const {  
 return this->bs;  
}  
  
void button::ButtonOperation() {  
 if(IsRoundPressed()){  
 cout<<"System block works";  
 }  
 else  
 {  
 cout<<"System block doesn't work";  
 }  
  
}  
  
ButtonState button::getButtonState() const {  
 return bt;  
}  
  
void button::OnStore(std::ofstream &aStream) {  
  
 aStream << "Shape of this button is " <<bs<<endl;  
 aStream << "State of this button is " <<bt<<endl;  
  
  
  
}  
  
void button::OnLoad(std::ifstream &aStream) {  
 char line[50];  
 for (int j = 0; j < 2; ++j) {  
 aStream.getline(( char \*) line, sizeof(line));  
 for (int i = 0; i <50 ; ++i) {  
  
 cout<<line[i];  
 }  
 cout<<endl;  
 }  
}

2.3.1 Файл*button.h*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
#pragma once  
#include <string>  
#include "../StorageInterface.h"  
  
using namespace std;  
enum ButtonState  
{  
 *ON*,  
 *OFF*};  
enum ButtonShape  
 {  
*Oval*,  
*Rectangular* };  
  
 class button :public MStorageInterface{  
 protected:  
 ButtonState bt;  
 ButtonShape bs;  
 int sa;  
 public:  
 void OnStore(std::ofstream &aStream) override;  
  
 void OnLoad(std::ifstream &aStream) override;  
  
 public:  
  
 void setButtonState(ButtonState b);  
 void setButtonShape(ButtonShape b);  
 ButtonShape getButtonShape() const;  
 ButtonState getButtonState() const;  
  
 bool IsRoundPressed();  
  
 virtual void ButtonOperation();  
  
  
};

2.3.2 Файл*View2.cpp*

//  
// Created by admin-1 on 07.09.2019.  
//  
#pragma once  
#include "../source .h/View2.h"  
#include <iostream>  
#include "../source .h/button.h"  
#include "../source .cpp/button.cpp"  
void View2::setData(const button &b)  
{  
 c=b;  
}  
  
void View2::ShowContent(button b ) {  
  
 int len = 31;  
 int len\_str1 = 25 + strlen("Kit-27B");  
 int len\_str2 = strlen("Gushin") + strlen("Danil") + strlen("Sergeyevich") + 6;  
 int len\_str3 = strlen("ButtonState = ")+5;  
 int len\_str4 = strlen("ButtonShape = ")+5;  
  
  
 cout << "\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
  
 cout << "\n\t\t\t\* Laba #2 ";  
 for (int counter = 0; counter < len - 25; counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* Completed by: st. gr." << "Kit-27B";  
 for (int counter = 0; counter <= (len - len\_str1); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "Gushin" << " " << "Danil" << " " << "Sergeyevich";  
 for (int counter = 0; counter <= (len - len\_str2); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "ButtonState = " <<b.getButtonState();  
 for (int counter = 0; counter <= (len - len\_str3); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "ButtonShape = " << b.getButtonShape() ;  
 for (int counter = 0; counter <= (len - len\_str4); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
 cout << endl;  
  
  
  
}  
  
void View2::ShowHeader() {  
cout<<"it's full info about button"<<endl;  
  
}  
  
void View2::ShowFooter() {  
cout<<"Version 2.0";  
  
}  
  
void View2::Display(button b) {  
  
 ShowHeader();  
 cout<<endl;  
 ShowContent(b);  
 cout<<endl;  
 ShowFooter();  
 cout<<endl;  
}  
  
 void View2::OnTimerAction(Button2 b2) {  
 cout<<"ButtonState() = "<<b2.getButtonState()<<endl;  
 cout<<"ButtonShape = "<<b2.getButtonShape()<<endl;  
 cout<<"Label = "<<b2.getLabel()<<endl;  
 cout<<"skan\_code = "<<b2.getSkan\_code()<<endl;  
 b2.ButtonOperation();  
}  
  
void View2::setData(const Button2 &b2) {  
b=b2;  
}

}

2.3.3 Файл*View2.h*

#include <iostream>  
#include "BaseView.h"  
  
using namespace std;  
  
  
class View2: public BaseView  
{  
  
  
public:  
 void ShowHeader() override ;  
 void ShowContent(button b) override ;  
 void ShowFooter() override ;  
 void Display(button b) override ;  
  
 void setData(const button &b);  
 void setData(const Button2 &b2);  
  
 static void OnTimerAction(Button2 b2) ;  
};

2.3.8 Файл*View3.h*

//  
// Created by admin-1 on 07.09.2019.  
//  
#pragma once  
  
  
class View3 : public BaseView {  
public:  
 void InfoButton2(Button2 b2);  
};

}

2.3.9 Файл*view3.срр*

//  
// Created by admin-1 on 07.09.2019.  
  
2.3.10 Файл*BaseView.h*

//  
// Created by admin-1 on 07.09.2019.  
//  
#pragma once  
  
class BaseView {  
public:  
 void CheckFuncOfTheButton( button\* px);  
 void INFO(button b);  
 button c;  
 Button2 b;  
 void PrintData(const button& b );  
  
 virtual void Display(button b) ;  
  
  
 virtual void ShowHeader() ;  
 virtual void ShowContent(button b) ;  
 virtual void ShowFooter();  
};

2.3.11 Файл*BaseView.cpp*

//  
// Created by admin-1 on 07.09.2019.  
//  
#pragma once  
#include "../source .h/BaseView.h"  
  
void BaseView::CheckFuncOfTheButton(button \*px){  
 px->ButtonOperation();  
  
}  
  
void BaseView::Display(button b) {  
ShowHeader();  
cout<<endl;  
ShowContent(b);  
cout<<endl;  
ShowFooter();  
cout<<endl;  
  
}  
  
void BaseView::ShowHeader() {  
cout<<"Info about button and her work";  
}  
  
void BaseView::ShowContent(button b) {  
 cout<<"it's content of button in the future when i create first version 1.0 View";  
  
}  
  
void BaseView::ShowFooter() {  
 cout<<"Warnings: don't press this button if you installing any programs on your laptop";  
}  
void BaseView::INFO(button b) {  
  
 cout <<"ButtonState = "<<b.getButtonState()<<endl;  
  
  
 cout <<"ButtonState = " <<b.getButtonShape()<<endl;  
  
  
}  
  
  
void BaseView::PrintData(const button& b )  
{  
  
  
  
  
 int len = 31;  
 int len\_str1 = 25 + strlen("Kit-27B");  
 int len\_str2 = strlen("Gushin") + strlen("Danil") + strlen("Sergeyevich") + 6;  
 int len\_str3 = strlen("ButtonState = ")+5;  
 int len\_str4 = strlen("ButtonShape = ")+5;  
  
  
 cout << "\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
  
 cout << "\n\t\t\t\* Laba #2 ";  
 for (int counter = 0; counter < len - 25; counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* Completed by: st. gr." << "Kit-27B";  
 for (int counter = 0; counter <= (len - len\_str1); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "Gushin" << " " << "Danil" << " " << "Sergeyevich";  
 for (int counter = 0; counter <= (len - len\_str2); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "ButtonState = " <<b.getButtonState();  
 for (int counter = 0; counter <= (len - len\_str3); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t\* " << "ButtonShape = " << b.getButtonShape() ;  
 for (int counter = 0; counter <= (len - len\_str4); counter++ )  
 cout << " ";  
  
 cout << "\*\n\t\t\t";  
 for (int counter = 0; counter < len; counter++ )  
 cout << "\*";  
 cout << endl;  
  
  
  
}

2.3.12 Файл*FileStorage.cpp*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
#include <share.h>  
#include <stdio.h>  
#include <iostream>  
#include <fstream>  
#include <tchar.h>  
  
#include "FileStorage.h"  
using namespace std;  
  
#include "StorageInterface.h"  
  
// -----------------------------------------------------------------------------  
// CFileStorage\* CFileStorage::Create(MStorageInterface& aInterface,   
// const char\* aFileName)  
// Creates instance of class  
// -----------------------------------------------------------------------------  
//  
CFileStorage\* CFileStorage::Create(MStorageInterface& aInterface,  
 const char\* aFileName) {  
 CFileStorage\* self = new CFileStorage(aInterface);  
 self->Construct(aFileName);  
  
 return self;  
}  
  
// -----------------------------------------------------------------------------  
// CFileStorage::CFileStorage(MStorageInterface& aInterface)  
// First stage constructor  
// -----------------------------------------------------------------------------  
//  
CFileStorage::CFileStorage(MStorageInterface& aInterface) :  
 iInterface(aInterface),  
 iFileName(NULL),  
 iLastError(0) {  
}  
  
// -----------------------------------------------------------------------------  
// void CFileStorage::Construct(const char\* aFileName)   
// Second stage constructor  
// -----------------------------------------------------------------------------  
//  
void CFileStorage::Construct(const char\* aFileName) {  
 iFileName = new char[strlen(aFileName) + 1];  
 strcpy(iFileName, aFileName);  
}  
  
// -----------------------------------------------------------------------------  
// CFileStorage::~CFileStorage()  
// Destructor  
// -----------------------------------------------------------------------------  
//  
CFileStorage::~CFileStorage() {  
 delete[] iFileName;  
}  
  
// -----------------------------------------------------------------------------  
// bool CFileStorage::Store()  
// Stores data by calling MStorageInterface::OnStore() callback  
// -----------------------------------------------------------------------------  
//  
bool CFileStorage::Store() {  
 ofstream fileStream;  
 fileStream.open(iFileName,  
 ios\_base::out | ios\_base::binary | ios\_base::trunc);  
 if (!fileStream) {  
 iLastError = *EFileOpen*;  
 return false;  
 }  
  
 iInterface.OnStore(fileStream);  
 fileStream.close();  
 iLastError = *ENoError*;  
  
 return true;  
}  
  
// -----------------------------------------------------------------------------  
// bool CFileStorage::Load()  
// Loads data by calling MStorageInterface::OnLoad() callback  
// -----------------------------------------------------------------------------  
//  
bool CFileStorage::Load() {  
 ifstream fileStream;  
 fileStream.open(iFileName, ios\_base::in | ios\_base::binary);  
 if (!fileStream) {  
 iLastError = *EFileOpen*;  
 return false;  
 }  
  
 iInterface.OnLoad(fileStream);  
 fileStream.close();  
 iLastError = *ENoError*;  
 return true;  
}  
  
// -----------------------------------------------------------------------------  
// int CFileStorage::GetLastError() const  
// Retrieves result of the last operation  
// -----------------------------------------------------------------------------  
//  
int CFileStorage::GetLastError() const {  
 return iLastError;  
}

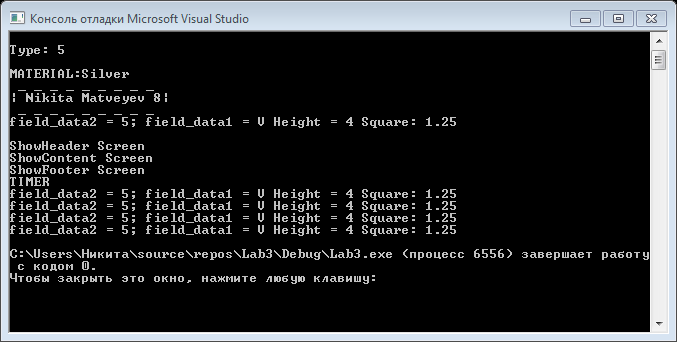
2.3.12 Файл*FileStorage.h*

#define \_CRT\_SECURE\_NO\_WARNINGS  
  
/\*  
 ============================================================================  
 Name : FileStore.h  
 Author : Rostyslav Gryb  
 Version : 1.00.001  
 Copyright : Mobile Effort  
 Description : Declaration of CFileStore  
 ============================================================================  
 \*/  
  
#ifndef FILE\_STORE\_H  
#define FILE\_STORE\_H  
  
class MStorageInterface;  
  
class CFileStorage {  
public:  
 /\*\*  
 \* Creates instance of class  
 \* @param aInterface Interface owner  
 \* @param aFileName File name for operationss  
 \* @return A pointer to constructed object  
 \*/  
 static CFileStorage\* Create(MStorageInterface& aInterface,  
 const char\* aFileName);  
  
 /\*\*  
 \* Destructor  
 \*/  
 ~CFileStorage();  
  
 /\*\*  
 \* Stores data by calling MStorageInterface::OnStore() callback  
 \* @return true if operation was successful, otherwise false  
 \* @note Use GetLastError() to receive type of error  
 \*/  
 bool Store();  
  
 /\*\*  
 \* Loads data by calling MStorageInterface::OnLoad() callback  
 \* @return true if operation was successful, otherwise false  
 \* @note Use GetLastError() to receive type of error  
 \*/  
 bool Load();  
  
 /\*\*  
 \* Retrieves result of the last operation  
 \* @return Returns last error code according to the last operation  
 \*/  
 int GetLastError() const;  
  
public:  
 enum TErrors {  
 *ENoError* = 0, *EFileOpen* = 1,  
  
 *ELastError* };  
  
protected:  
 /\*\*  
 \* First stage constructor  
 \* @param aInterface Interface owner  
 \*/  
 CFileStorage(MStorageInterface& aInterface);  
  
 /\*\*  
 \* Second stage constructor  
 \* @param aFileName Name of file for operations  
 \*/  
 void Construct(const char\* aFileName);  
  
private:  
 MStorageInterface& iInterface;  
 char\* iFileName;  
 int iLastError;  
};  
  
#endif // FILE\_STORE\_H

2.3.12 Файл*StorageInterface.h*

#ifndef STORAGE\_INTERFACE\_H  
#define STORAGE\_INTERFACE\_H  
  
#include <iostream>  
  
class MStorageInterface {  
public:  
 virtual ~MStorageInterface() {};  
  
 /\*\*  
 \* Зберігає дані в потік  
 \* @param aStream Відкритий потік для збереження даних  
 \*/  
 virtual void OnStore(std::ofstream& aStream) = 0;  
  
 /\*\*  
 \* Завантажує дані із потоку  
 \* @param aStream Відкритий потік для завантаження даних  
 \*/  
 virtual void OnLoad(std::ifstream& aStream) = 0;  
};  
  
#endif // STORAGE\_INTERFACE\_H

**3 РЕЗУЛЬТАТИ**

****

**ВИСНОВКИ**

Навчились застосовувати інтерфейси для роботи класів на прикладі задач серіалізації.