Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

за 1 семестр

По дисциплине: «ООТПиСП»

Тема: «Наследование и виртуальные функции»

Выполнила:

Студентка 2 курса

Группы ПО-4(1)

Иваненко И. Л.

Проверил:

Миндер А. В.

2021

Лабораторная работа №3

Наследование и виртуальные функции

Цель работы: получить практические навыки создания иерархии классов и использования статических компонентов класса.

Вариант 10

Задание:

1. Определить иерархию классов (в соответствии с вариантом).

Перечень классов: организация, страховая компания, судостроительная компания, завод.

2. Определить в классе статическую компоненту – указатель на начало связанного списка объектов и статическую функцию для просмотра списка (инициализировать вне определения класса, в глобальной области).

3. Реализовать классы. Определить в классах все необходимые конструкторы и деструктор.

4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.

5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.

6. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе.

Задание по варианту: Квитанция, накладная, документ, чек;

Иерархия классов



Код программы:

Document.h:

#pragma once

#include<string>

class Document {

protected:

std::string date;

int number;

public:

Document(std::string date, int number);

Document();

virtual void print() = 0;

struct ListOfDocuments {

Document\* document;

ListOfDocuments\* nextItem;

};

static ListOfDocuments\* head;

static void printListOfDocuments();

void push();

void setDate(std::string date);

void setNumber(int number);

std::string getDate();

int getNumber();

virtual ~Document();

};

Document.cpp:

#include "Document.h"

Document::ListOfDocuments\* Document::head = NULL;

Document::Document(std::string date, int number) {

this->date = date;

this->number = number;

this->push();

}

Document::Document() {

this->date = "01.01.1970";

this->number = NULL;

this->push();

}

void Document::setDate(std::string date) {

this->date = date;

}

void Document::setNumber(int number) {

this->number = number;

}

std::string Document::getDate() {

return this->date;

}

int Document::getNumber() {

return this->number;

}

void Document::push() {

ListOfDocuments\* listItem = new ListOfDocuments;

listItem->document = this;

listItem->nextItem = head;

head = listItem;

}

void Document::printListOfDocuments() {

ListOfDocuments\* listItem = head;

while (listItem != NULL) {

listItem->document->print();

listItem = listItem->nextItem;

}

}

Document::~Document() {

delete this->head;

}

Check.h:

#pragma once

#include "Document.h"

class Check : public Document {

private:

float price;

public:

Check(std::string date, int number, float price);

Check();

void print() override;

void setPrice(float price);

float getPrice();

~Check() override;

};

Check.cpp:

#include "Check.h"

#include <iostream>

Check::Check(std::string date, int number, float price)

: Document(date, number) {

this->price = price;

}

Check::Check() : Document() {

this->price = NULL;

}

void Check::setPrice(float price) {

this->price = price;

}

float Check::getPrice() {

return this->price;

}

void Check::print() {

std::cout << "Check" << std::endl;

std::cout << "Number: " << this->number << std::endl;

std::cout << "Price: " << this->price << std::endl;

std::cout << "Date: " << this->date << std::endl << std::endl;

}

Check::~Check() {

}

Receipt.h:

#pragma once

#include "Check.h"

#include <string>

class Receipt : public Document {

private:

std::string bank;

float price;

public:

Receipt(std::string date, int number, float price, std::string bank);

Receipt();

void print() override;

void setPrice(float price);

float getPrice();

void setBank(std::string bank);

std::string getBank();

~Receipt() override;

};

Receipt.cpp:

#include <iostream>

#include "Receipt.h"

Receipt::Receipt(std::string date, int number, float price,

std::string bank) : Document(date, number) {

this->bank = bank;

this->price = price;

}

Receipt::Receipt() : Document() {

this->bank = "Tinkoff bank";

this->price = NULL;

}

void Receipt::setBank(std::string bank) {

this->bank = bank;

}

void Receipt::setPrice(float price) {

this->price = price;

}

std::string Receipt::getBank() {

return this->bank;

}

float Receipt::getPrice() {

return this->price;

}

void Receipt::print() {

std::cout << "Receipt" << std::endl;

std::cout << "Number: " << this->number << std::endl;

std::cout << "Bank: " << this->bank << std::endl;

std::cout << "Price: " << this->price << std::endl;

std::cout << "Date: " << this->date << std::endl << std::endl;

}

Receipt::~Receipt() {

}

Waybill.h:

#pragma once

#include "Document.h"

#include <string>

class Waybill : public Document {

private:

std::string sender;

std::string recipient;

std::string productName;

public:

Waybill(std::string date, int number, std::string sender,

std::string recipient, std::string productName);

Waybill();

void print() override;

void setSender(std::string sender);

void setRecipient(std::string recipient);

void setProductName(std::string productName);

std::string getSender();

std::string getRecipient();

std::string getProductName();

~Waybill() override;

};

Waybill.cpp:

#include "Waybill.h"

#include <iostream>

Waybill::Waybill(std::string date, int number, std::string sender,

std::string recipient, std::string productName) : Document(date, number){

this->sender = sender;

this->recipient = recipient;

this->productName = productName;

}

Waybill::Waybill() {

this->sender = "tuesdaydude";

this->recipient = "tuesdaydude";

this->productName = "coursework";

}

void Waybill::setSender(std::string sender) {

this->sender = sender;

}

void Waybill::setRecipient(std::string recipient) {

this->recipient = recipient;

}

void Waybill::setProductName(std::string productName) {

this->productName = productName;

}

std::string Waybill::getSender() {

return this->sender;

}

std::string Waybill::getRecipient() {

return this->recipient;

}

std::string Waybill::getProductName() {

return this->productName;

}

void Waybill::print() {

std::cout << "Waybill" << std::endl;

std::cout << "Number: " << this->number << std::endl;

std::cout << "Sender: " << this->sender << std::endl;

std::cout << "Recipient: " << this->recipient << std::endl;

std::cout << "Product Name: " << this->productName << std::endl;

std::cout << "Date: " << this->date << std::endl << std::endl;

}

Waybill::~Waybill() {

}

Source.cpp:

#include <iostream>

#include "Receipt.h"

#include "Waybill.h"

using namespace std;

int main() {

Document\* check = new Check("12.12.2012", 10, 10);

Document\* waybill = new Waybill("12.12.2012", 10, "tuesdaydude", "kreativnvp", "website");

Document\* receipt = new Receipt("10.10.2010", 11, 10.19, "tinkoff");

Document\* check2 = new Check();

Document\* waybill2 = new Waybill();

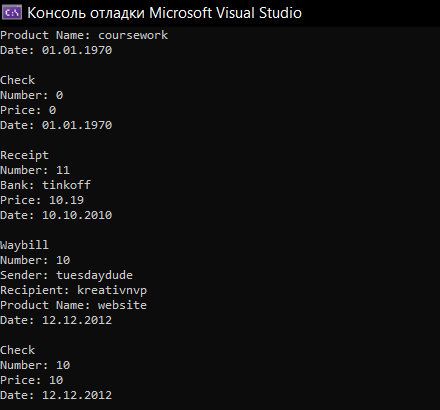
Document\* receipt2 = new Receipt();

Document::printListOfDocuments();

return 0;

}

Результат выполнения:



Вывод: получил практические навыки создания иерархии классов и использования статических компонентов класса.