Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа 6

"Наследование и виртуальные функции"

Выполнил:

студент 2 курса группы АС-53

Зайчук Д.Р.

Проверила:

Давидюк Ю.И.

Брест, 2020

Вариант 10

Иерархия классов:

Описание полей и пояснения к иерархии классов находятся в Приложении 1.

Листинг программы с определением классов:

```
#include <iostream>
#include <string>

using namespace std;

class Document {
public:
	static Document* start;
	Document* next = NULL;
	static void ShowList() {
		Document* p = start;
		while (p) {
			p->show();
			p = p->next;
		}
	}
	Document() {
		cout << "defaul constructor" << endl;
	}
	Document(string newDate, double newSum) {
		date = newDate;
		sum = newSum;
	}
	virtual ~Document() {
		cout << "default destructor" << endl;
	}
	virtual void show() = 0;
	virtual void input() = 0;
	virtual void addToList() = 0;
protected:
	string date;
	double sum;
};

class Bill :public Document {
public:
	Bill() : Document() {};
	Bill(int amountN, string sellerN, string nameOfGoodN,
		 string dateN, double sumN) {
		amount = amountN;
		seller = sellerN;
		nameOfGood = nameOfGoodN;
```

```
		date = dateN;
		sum = sumN;
	}
	void show() {
		cout << "Bill" << endl;
		cout << "Date: " << date << endl;
		cout << "Sum: " << sum << endl;
		cout << "Name of good: " << nameOfGood << endl;
		cout << "Amount: " << amount << endl;
		cout << "Seller: " << seller << endl;
		cout << endl;
	}
	void input() {
		cout << "Bill" << endl;
		cout << "Enter date: ";		cin >> date;
		cout << "Enter sum: ";		cin >> sum;
		cout << "Enter name of good: "; cin >> nameOfGood;
		cout << "Enter amount: ";	cin >> amount;
		cout << "Enter seller: ";	cin >> seller;
		cout << endl;
	}
	void addToList() {
		Document* p = start;
		while (p->next) {
			p = p->next;
		}
		p->next = this;
	}
protected:
	int amount;
	string seller;
	string nameOfGood;
};

class Invoice :public Bill {
public:
	Invoice() : Bill() {};
	Invoice(int amountN, string sellerN, string nameOfGoodN,
		string dateN, double sumN, string nameOfOrganizationN,
		int numberOfInvoiceN, string buyerN, string gradeN) {
		amount = amountN;
		seller = sellerN;
		nameOfGood = nameOfGoodN;
		date = dateN;
		sum = sumN;
		nameOfOrganization = nameOfOrganizationN;
		numberOfInvoice = numberOfInvoiceN;
		buyer = buyerN;
		grade = gradeN;
	}
	void show() {
		cout << "Invoice" << endl;
		cout << "Name of organization: " << nameOfOrganization << endl;
		cout << "Number of invoice: " << numberOfInvoice << endl;
		cout << "Date: " << date << endl;
		cout << "Sum: " << sum << endl;
		cout << "Name of good: " << nameOfGood << endl;
		cout << "Amount: " << amount << endl;
		cout << "Grade: " << grade << endl;
		cout << "Seller: " << seller << endl;
		cout << "Buyer: " << buyer << endl;
```

```
		cout << endl;
	}
	void input() {
		cout << "Invoice" << endl;
		cout << "Enter name of organization: ";   cin >> nameOfOrganization;
		cout << "Enter number of invoice: ";     cin >> numberOfInvoice;
		cout << "Enter date: ";                          cin >> date;
		cout << "Enter sum: ";                           cin >> sum;
		cout << "Enter name of good: ";                cin >> nameOfGood;
		cout << "Enter amount: ";                      cin >> amount;
		cout << "Enter grade: ";                       cin >> grade;
		cout << "Enter seller: ";                      cin >> seller;
		cout << "Enter buyer: ";                       cin >> buyer;
		cout << endl;
	}
	void addToList() {
		Document* p = start;
		while (p->next) {
			p = p->next;
		}
		p->next = this;
	}
private:
	string nameOfOrganization;
	int numberOfInvoice;
	string buyer;
	string grade;
};

class Receipt :public Document {
public:
	Receipt() : Document() {};
	Receipt(string dateN, double sumN, int numberOfReceiptN,
		string nameOfOrganizationN,     int numberOfContractN,
		string dateOfContractN,   string typeN, string clientN,
		string periodN) {
		date = dateN;
		sum = sumN;
		numberOfReceipt = numberOfReceiptN;
		nameOfOrganization = nameOfOrganizationN;
		numberOfContract = numberOfContractN;
		dateOfContract = dateOfContractN;
		type = typeN;
		client = clientN;
		period = periodN;
	}
	void show() {
		cout << "Receipt" << endl;
		cout << "Name of organization: " << nameOfOrganization << endl;
		cout << "Number of receipt: " << numberOfReceipt << endl;
		cout << "Number of contract: " << numberOfContract << endl;
		cout << "Date of contract: " << dateOfContract << endl;
		cout << "Type of insurance: " << type << endl;
		cout << "Client: " << client << endl;
		cout << "Period: " << period << endl;
		cout << "Sum: " << sum << endl;
		cout << "Date: " << date << endl;
		cout << endl;
	}
	void input() {
		cout << "Receipt" << endl;
```

```
            cout << "Enter name of organization: "; cin >> nameOfOrganization;
            cout << "Enter number of receipt: ";   cin >> numberOfReceipt;
            cout << "Enter number of contract: ";  cin >> numberOfContract;
            cout << "Enter date of contract: ";          cin >> dateOfContract;
            cout << "Enter type of insurance: ";   cin >> type;
            cout << "Enter client: ";                    cin >> client;
            cout << "Enter period: ";                    cin >> period;
            cout << "Enter sum: ";                             cin >> sum;
            cout << "Enter date: ";                            cin >> date;
            cout << endl;
      }
      void addToList() {
            Document* p = start;
            while (p->next) {
                  p = p->next;
            }
            p->next = this;
      }
protected:
      int numberOfReceipt;
      string nameOfOrganization;
      int numberOfContract;
      string dateOfContract;
      string type;
      string client;
      string period;
};

Document* Document::start = NULL;

int main() {
      Bill* bill;
      Receipt* receipt;
      Invoice* invoice;
      bill = new Bill();
      receipt = new Receipt();
      invoice = new Invoice();
      bill->input();
      receipt->input();
      invoice->input();
      Document::start = bill;
      receipt->addToList();
      invoice->addToList();
      Document::ShowList();
}
```

Результат выполнения


```
defaul constructor
defaul constructor
defaul constructor
Bill
Enter date: 12/12/12
Enter sum: 12
Enter name of good: 12
Enter amount: 12
Enter seller: 21

Receipt
Enter name of organization: 13
Enter number of receipt: 13
Enter number of contract: 13
Enter date of contract: 13/13/13
Enter type of insurance: 13
Enter client: 13
Enter period: 13
Enter sum: 13
Enter date: 13/13/13

Invoice
Enter name of organization: 14
Enter number of invoice: 14
Enter date: 14/14/14
Enter sum: 14
Enter name of good: 14
Enter amount: 14
Enter grade: 14
Enter seller: 14
Enter buyer: 14

Bill
Date: 12/12/12
Sum: 12
Name of good: 12
Amount: 12
Seller: 21

Receipt
Name of organization: 13
Number of receipt: 13
Number of contract: 13
Date of contract: 13/13/13
Type of insurance: 13
Client: 13
Period: 13
Sum: 13
Date: 13/13/13

Invoice
Name of organization: 14
Number of invoice: 14
Date: 14/14/14
Sum: 14
Name of good: 14
Amount: 14
Grade: 14
Seller: 14
Buyer: 14
```