Технологии конструирования программного обеспечения

Отчет по лабораторной работе № 5

Группа: 221-3210 **Студент:** Обухов Алексей Сергеевич

Задание на лабораторную работу

Разработать диаграмму конечных автоматов (состояний) для заданного класса. Описать в форме таблицы варианты реакции экземпляра класса на операции, вызываемые в указанных состояниях.

Класс Телефон:

Атрибуты:

- Номер;
- Баланс;
- Вероятность поступления звонка.

Операции:

- Позвонить;
- Ответить на звонок;
- Завершить разговор;
- Пополнить баланс.

Состояния:

- Ожидание;
- Звонок;
- Разговор;
- Заблокирован (отрицательный баланс)

Диаграмма классов

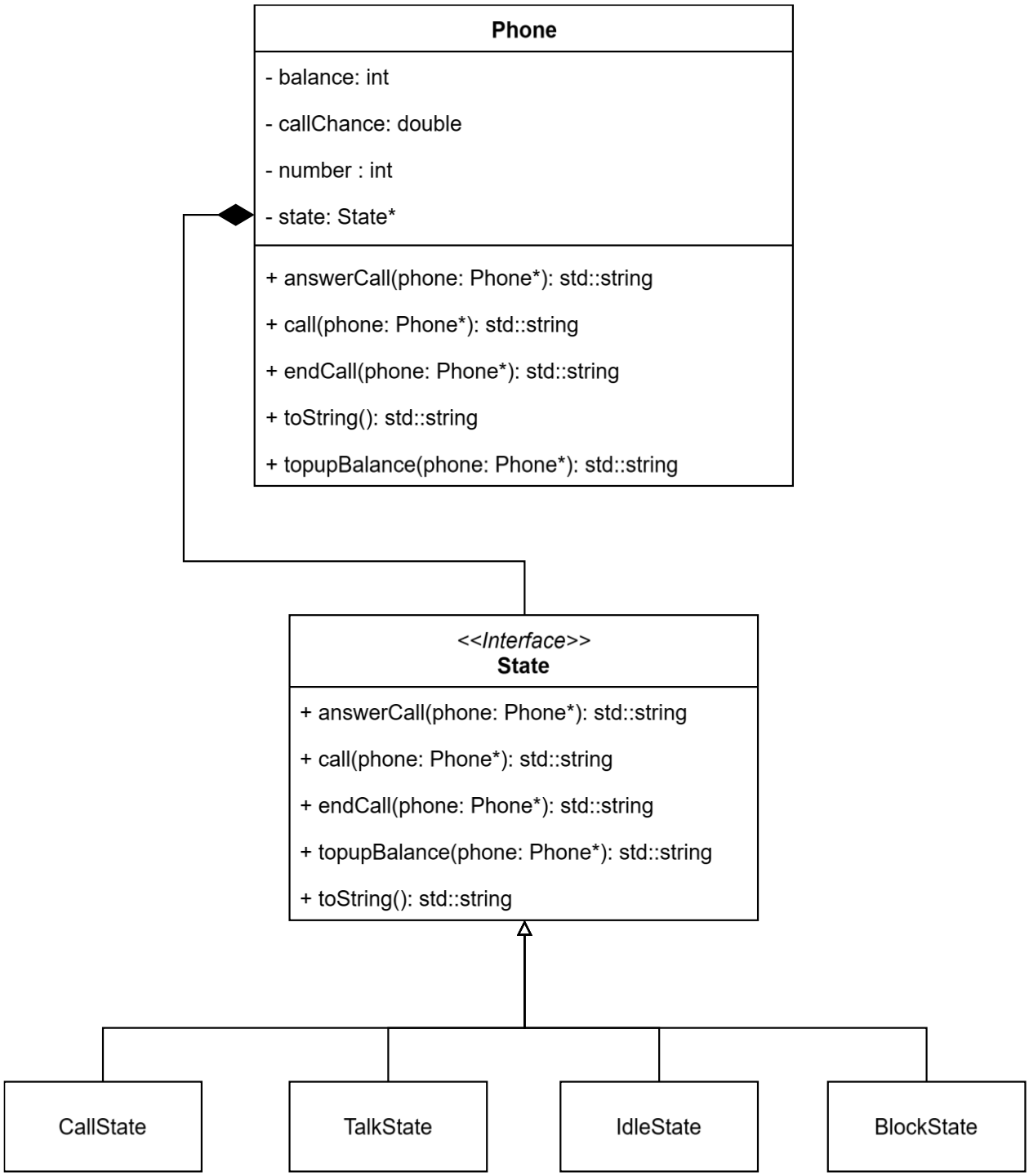


Диаграмма последовательности

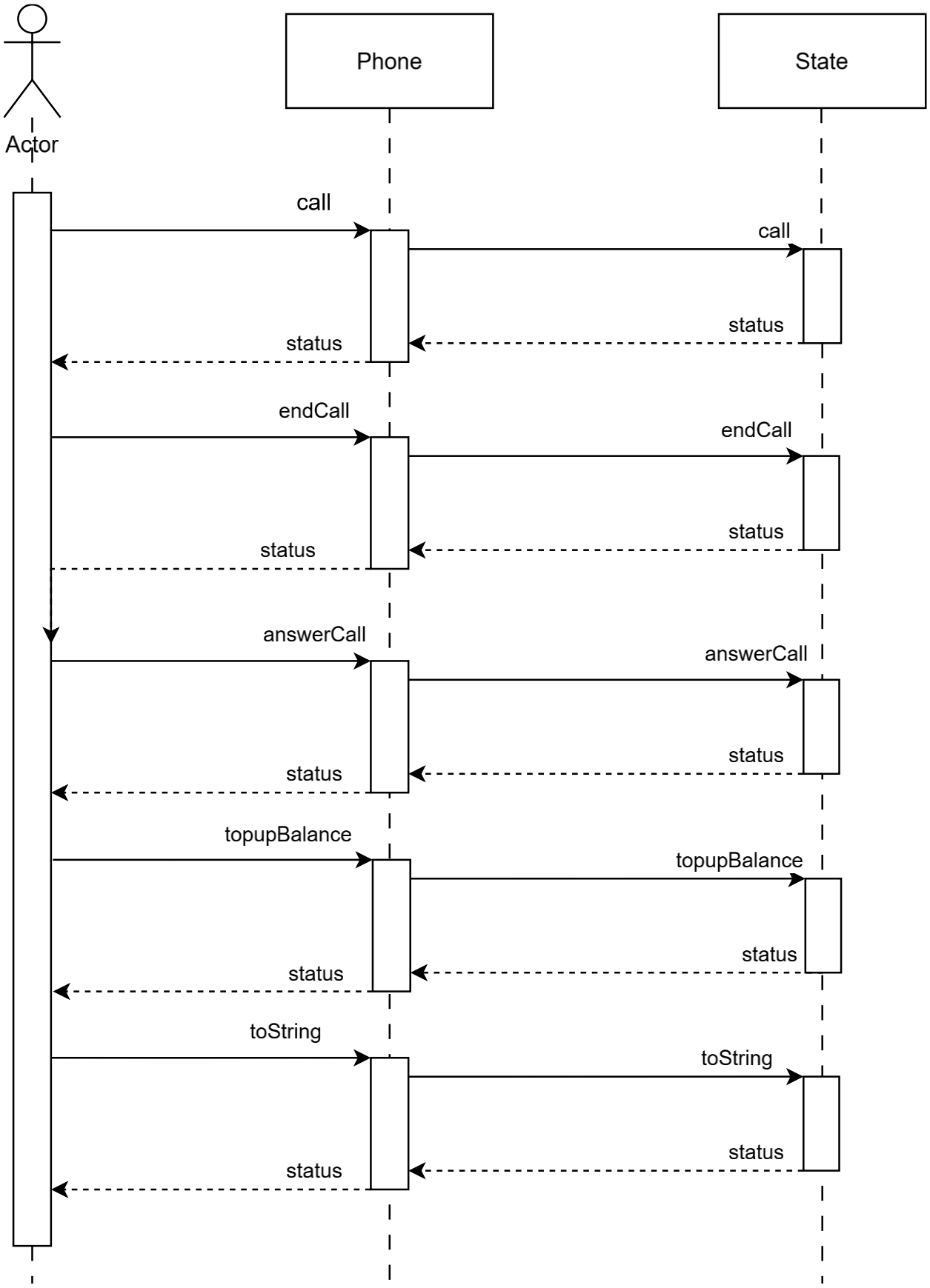
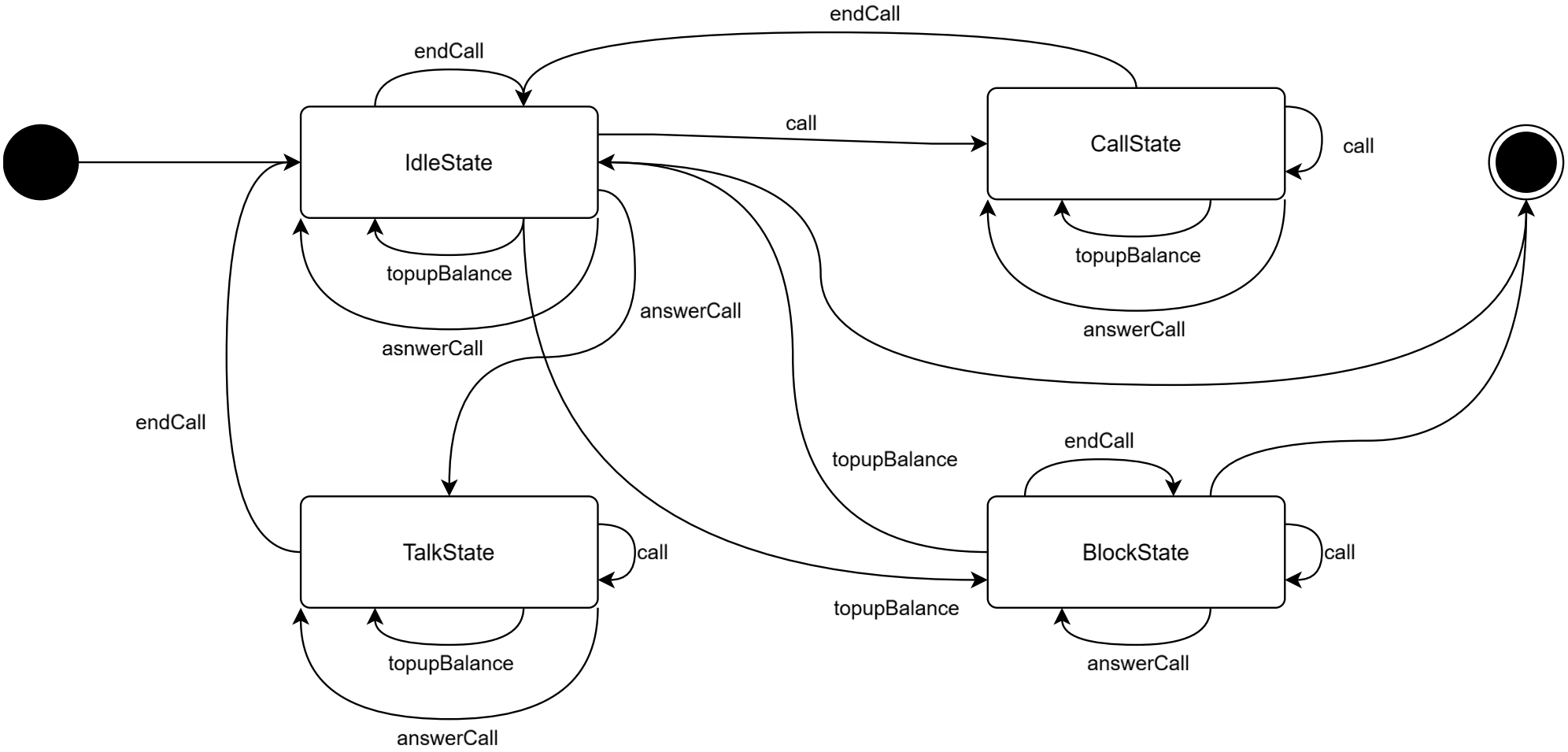


Диаграмма конечных автомат



Исходный код программы без графического пользовательского интерфейса

Содержимое «Phone.h»

```
#include <string>

#pragma once
#define CALL_PRICE 10

class State;

// Телефон
class Phone {
	friend class State;
	friend class IdleState;
	friend class CallState;
	friend class TalkState;
	friend class BlockState;

private:
	int number;		// Номер телефона
	int balance;		// Баланс
	int callChance;	// Вероятность поступления звонка в процентах
	State* state;		// Состояние телефона

public:
	std::string call();
	std::string answerCall();
	std::string endCall();
	std::string topupBalance(int amount);
	std::string toString();

	Phone();
	Phone(int number, int balance, double callChance);
};



// Базовый класс состояния
class State {
public:
	virtual std::string call(Phone* phone) = 0;
	virtual std::string answerCall(Phone* phone) = 0;
	virtual std::string endCall(Phone* phone) = 0;
	virtual std::string toString() = 0;
	virtual std::string topupBalance(Phone* phone, int amount);
};

// Состояние ожидания
class IdleState : public State {
	std::string call(Phone* phone) override;

	std::string answerCall(Phone* phone) override;

	std::string endCall(Phone* phone) override;

	std::string toString() override;
};

// Состояние звонка
class CallState : public State {
	std::string call(Phone* phone) override;
```

```
        std::string answerCall(Phone* phone) override;

        std::string endCall(Phone* phone) override;

        std::string toString() override;
};

// Состояние разговора
class TalkState : public State {
        std::string call(Phone* phone) override;

        std::string answerCall(Phone* phone) override;

        std::string endCall(Phone* phone) override;

        std::string toString() override;
};


// Состояние блокировки
class BlockState : public State {
        std::string call(Phone* phone) override;

        std::string answerCall(Phone* phone) override;

        std::string endCall(Phone* phone) override;

        std::string toString() override;

        std::string topupBalance(Phone* phone, int amount) override;
};
```

Содержимое «Phone.cpp»

```
#include "Phone.h"

std::string IdleState::call(Phone* phone)
{
	if (phone->balance > CALL_PRICE) {
		phone->state = new CallState;
		phone->balance -= CALL_PRICE;
		return "Выполняем звонок";
	}
	return "Недостаточно средств для звонка";
}

std::string IdleState::answerCall(Phone* phone)
{
	if (rand() % 100 < phone->callChance) {
		phone->state = new TalkState;
		return "Отвечаем на звонок";
	}
	else
		return "Нет входящего звонка";}

std::string IdleState::endCall(Phone* phone)
{
	return "В текущий момент нет звонка, который можно было бы заврешить";
}

std::string IdleState::toString()
{
	return "Ожидание";
}

std::string CallState::call(Phone* phone)
{
	return "В настоящий момент звонок уже выполняется";
}

std::string CallState::answerCall(Phone* phone)
{
	return "Нельзя ответить на звонок, во время выполнения звонка";
}

std::string CallState::endCall(Phone* phone)
{
	phone->state = new IdleState;
	return "Звонок завершен";
}

std::string CallState::toString()
{
	return "Выполняется звонок";
}

std::string TalkState::call(Phone* phone)
{
	return "Нельзя начать новый звонок, поскольку Вы уже разговариваете";
}

std::string TalkState::answerCall(Phone* phone)
{
	return "Нельзя ответить на звонок, посколько Вы уже разговариваете";
}

std::string TalkState::endCall(Phone* phone)
```

```
{
	phone->state = new IdleState;
	return "Разговор был завершен";
}

std::string TalkState::toString()
{
	return "Идет разговор";
}

std::string BlockState::call(Phone* phone)
{
	return "Невозможно совершить звонок с отрицательным балансом";
}

std::string BlockState::answerCall(Phone* phone)
{
	return "Невозможно ответить на звонок с отрицательным балансом";
}

std::string BlockState::endCall(Phone* phone)
{
	return "В текущий момент нет звонка, который можно было бы завершить";
}

std::string BlockState::toString()
{
	return "Блокировка";
}

std::string BlockState::topupBalance(Phone* phone, int amount)
{
	phone->balance += amount;
	std::string res;
	res += "Баланс был пополнен на " + std::to_string(amount) + "\n";
	res += "Новый баланс: " + std::to_string(phone->balance);

	if (phone->balance >= 0) {
		phone->state = new IdleState;
		res += "\nБлокировка снята";
	}

	return res;
}

std::string Phone::call()
{
	return state->call(this);
}

std::string Phone::answerCall()
{
	return state->answerCall(this);
}

std::string Phone::endCall()
{
	return state->endCall(this);
}

std::string Phone::topupBalance(int amount)
{
	return state->topupBalance(this, amount);
}
```

```
std::string Phone::toString()
{
	std::string data;
	data += "Номер:  " + std::to_string(number) + "\n";
	data += "Баланс: " + std::to_string(balance) + "\n";
	data += "Вероятность звонка: " + std::to_string(callChance) + "\n";
	data += "Состояние: " + state->toString() + "\n";

	return data;
}

Phone::Phone()
{
	number = rand() % 1'000;
	balance = 100;
	callChance = 50;
	state = new IdleState();
}

Phone::Phone(int number, int balance, int callChance)
{
	this->number = number;
	this->balance = balance;
	this->callChance = callChance;
	if (balance < 0)
		state = new BlockState();
	else
		state = new IdleState();
}

std::string State::topupBalance(Phone* phone, int amount)
{
	phone->balance += amount;
	std::string res;
	res += "Баланс был пополнен на " + std::to_string(amount) + "\n";
	res += "Новый баланс: " + std::to_string(phone->balance);
	if (phone->balance < 0)
		phone->state = new BlockState;
	return res;
}
```

Содержимое «main.cpp»

```
#include <iostream>
#include <string>
#include "Phone.h"


int main() {
	setlocale(LC_ALL, "Russian");

	Phone phone(123, 30, 50);
	std::cout << phone.toString() << "\n";
	std::cout << phone.call() << "\n";
	std::cout << phone.endCall() << "\n";
	std::cout << phone.answerCall() << "\n";
	std::cout << phone.answerCall() << "\n\n";
	std::cout << phone.toString() << "\n";
	std::cout << phone.endCall() << "\n";
	std::cout << phone.toString();

	return 0;
}
```

Исходный код программы с графическим пользовательским интерфейсом

Исходный код программы с графическим пользовательским интерфейсом расположен в репозитории на github, доступном по следующей ссылке:

https://github.com/AlexOS12/tkpo_2024/tree/main/lab5/PhoneForm