

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой _____

Преподаватель _____ (подпись)
«___» _____ 2022 г.

Отчет
по лабораторной работе № 4
«Делегирование и проху»
по дисциплине «Объектно-ориентированное программирование»

Студент гр. ПИ-02
Чередов Р.А.

Ассистент кафедры ПМ,
Рахманин Д.С.

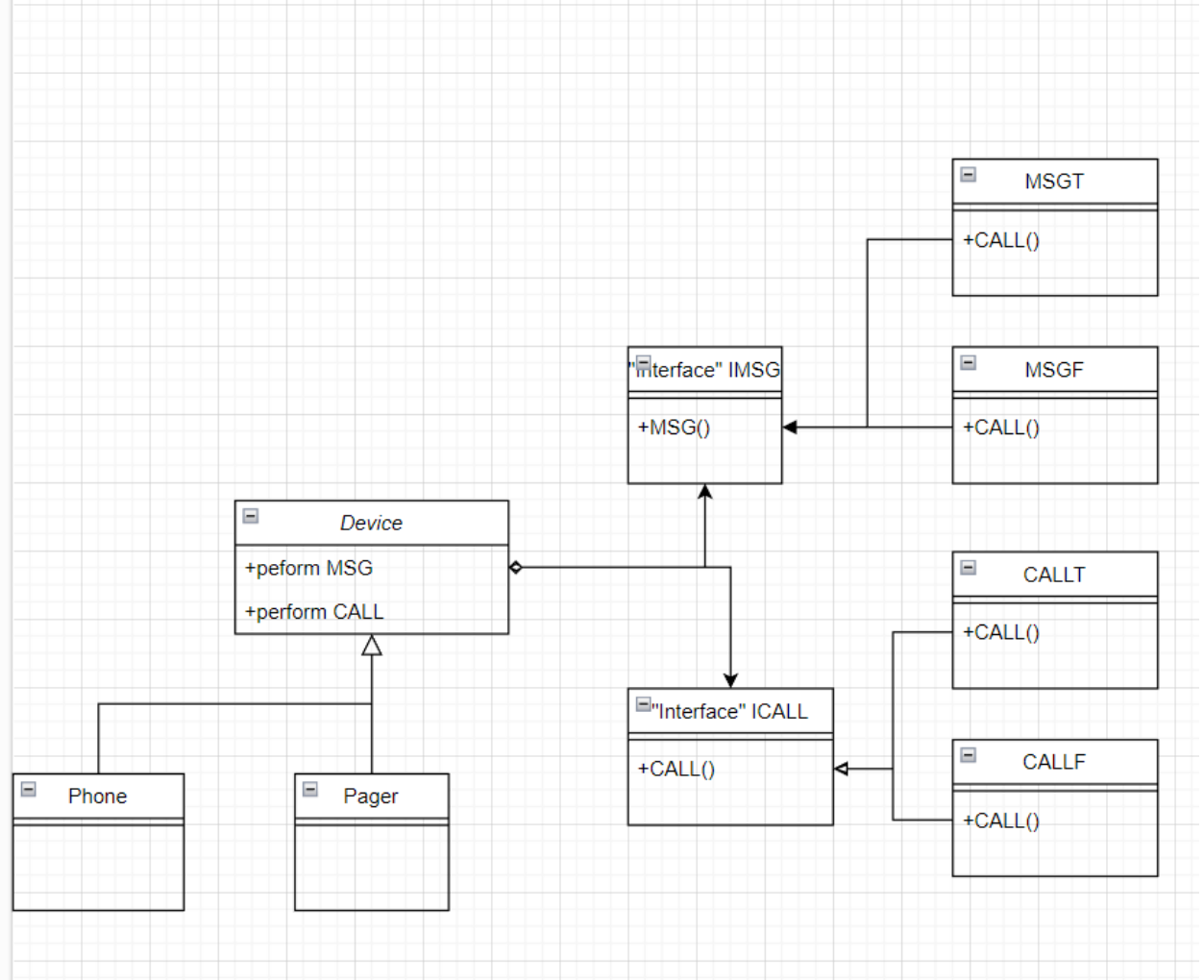
Барнаул 2023

125	Чередов Роман Алексеевич	Системы анализа больших данных	Проектирование системы обработки потоков данных о местонахождении абонентов сети сотовой связи
-----	-----------------------------	-----------------------------------	--

Задание:

Реализовать паттерны прокси и делегирования в своем проекте

1) Статическое делегирование



```

using namespace std;
class IMSG {
public:
    virtual void    msg() = 0; // интерфейс не имеет реализации
};
class ICALL {
public:
    virtual void    call() = 0; // интерфейс не имеет реализации
};

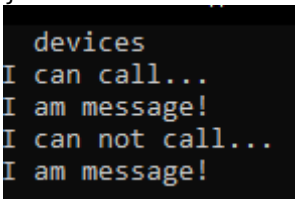
class MSGT : public IMSG {
    // класс поведения для устройств, которые умеют писать смс
public:
    void msg() {
        printf("I am message!\n");
    }
};
class MSGF : public IMSG {
    // класс поведения для устройств, которые не умеют писать смс
public:
    void msg() {

```

```

        printf("I am dont message!\n");
    }
};
class CALLT : public ICALL {
    // класс поведения для устройств, которые НЕ умеют call
public:
    void call() {
        printf("I can call...\n");
    }
};
class CALLF : public ICALL {
    // класс поведения для устройств, которые НЕ умеют call
public:
    void call() {
        printf("I can not call...\n");
    }
};
class Device {           // абстрактный класс устройства
public:
    MSG* msgaction;
    ICALL* callaction;
    Device() {}
    ~Device();
    // делегируем выполнение операции классам поведения :
    void performcall() { callaction->call(); }
    void performmsg() { msgaction->msg(); }
};
class Phone : public Device {
public:
    Phone() {
        callaction = new CALLT();
        msgaction = new MSGT();
    }
};
class Pager : public Device {
public:
    Pager() {
        callaction = new CALLF();
        msgaction = new MSGT();
    }
};
int main() {           // создаем объекты устройств
    printf(" devices\n");
    Phone Phone1;
    Pager Pager2;
    Phone1.performcall();
    Phone1.performmsg();
    Pager2.performcall();
    Pager2.performmsg();
}

```

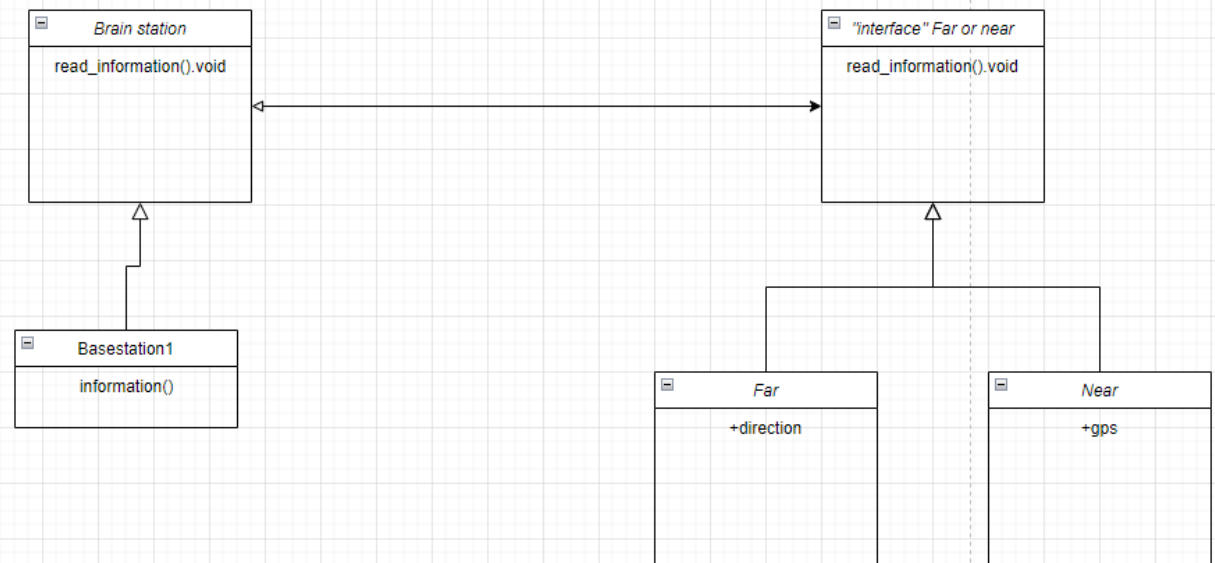


```

devices
I can call...
I am message!
I can not call...
I am message!

```

2) Динамическое делегирование



```

#include <iostream>
#include <string>
using namespace std;
class farornear
{
public:
    virtual string readinformation() = 0;
};
class BrainStation
{
public:
    void ChekInfo(farornear* rc)
    {
        info = rc;
    }
    string readinformation()
    {
        if (info)
        {
            return info->readinformation();
        }
        return "Не установлен ";
    }
private:
    farornear* info = nullptr;;
};

class Far : public farornear
{
public:
    string readinformation() override
    {
        return "Направление";
    }
};

class Near : public farornear
{
public:
    string readinformation() override
    {

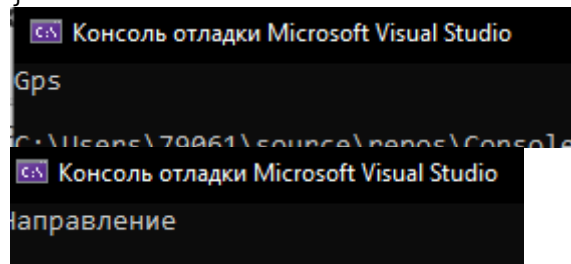
```

```

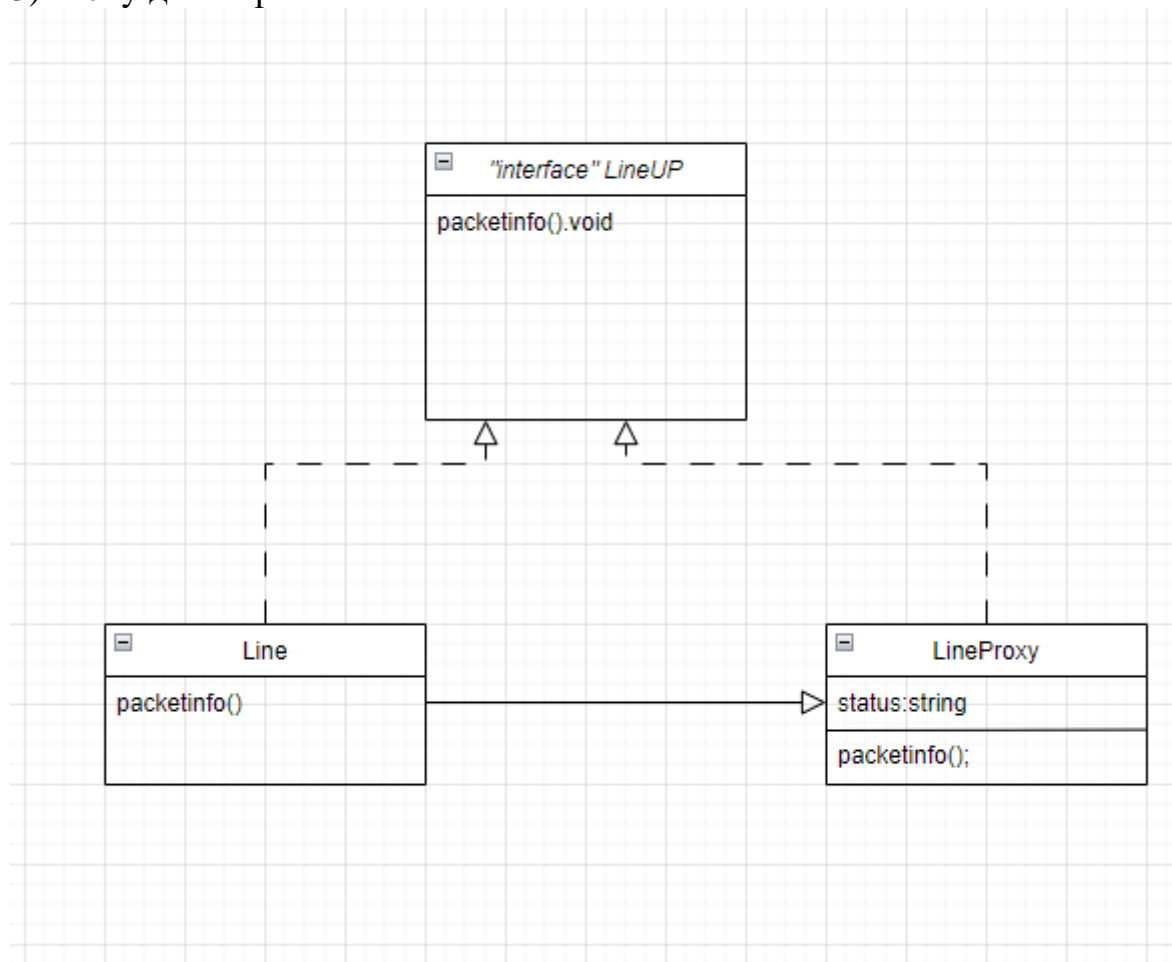
        // create pdf report logic
        return "Gps";
    }
};
int main()
{
    setlocale(LC_ALL, "Russian");
    Far Far;
    Near Near;
    BrainStation BrainStation;
    BrainStation.ChekInfo(&Far);
    BrainStation.ChekInfo(&Near);
    cout << BrainStation.readinformation() << endl;

    return 0;
}

```



3) Проxy делегирование



```

class LineUP {
    // класс, для которого создадим Proxy

```

```

public:
    virtual void packetinfo() = 0;
};
class LineUP : public LineUP {
    // настоящий класс для обработки данных
public:
    int a;

    virtual void packetinfo() {
        if (a == 1) {
            cout << "i dont have package\n";
        }
        else {
            cout << "i have package: " << endl;
        }
    }
    Line(int inA) { a = inA;}
};
class LineProxy : public LineUP {
private:
    Line* prox;
    void status() { cout << "im ready" << endl; }

public:

    virtual void packetinfo() { cout << "No " << endl; }

    LineProxy(int inA ) {
        prox = new Line(inA);
        // здесь Proxy создает реальный объект M1
    }
    ~LineProxy() { delete prox; }
};
int main() {
    LineUP* t = new Line(0);
    LineUP* p = new LineProxy(1);
    cout << "Line\n";
    t->packetinfo();
    cout << "\LineProxy\n";
    p->packetinfo();
    delete p;
    delete t;
    return 0;
}

```

```

Line
i have package:
LineProxy
No

```