

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)
«___» _____ 2023 г.

Отчет
по лабораторной работе № 8
«Паттерны поведения 1 - реализация
(State, Observer)»
по дисциплине «Объектно-ориентированное программирование»

Студент гр. ПИ-02
Чередов Р.А.

Ассистент кафедры ПМ,
Рахманин Д. С.

Барнаул 2023

Тема: Паттерны поведения 1 - реализация
(State , Observer)

Отчет:

Логи работы программы

Архив с кодом программы

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

State

```
#include <iostream>
#include <typeinfo>
#include <cstdlib>
#include <ctime>
using namespace std;

class Packetinfo;

//Состояние
class State {
public:
    virtual void Display(Packetinfo* inCon) = 0; //Вывод информации
};

class Packetinfo {
private:
    State* currentState;

public:
    Packetinfo(State* inSt) {
        currentState = inSt;
    }
    //Установить состояние
    void setState(State* inSt) {
        if (currentState)
            delete currentState;
        currentState = inSt;
    }
    //Вывод состояния
    void Display() {
        if (currentState)
            currentState->Display(this);
    }
    ~Packetinfo() {
        delete currentState;
    }
};

class near : public State {
public:
    void Display(Packetinfo* inCon);
};

class direction : public State {
private:
    int count; //Счетчик
    int min = 0;
    int max = 5;
public:
    direction() {
```

```

    }
    virtual void Display(Packetinfo* inCon) {

        count = min + rand() % ((max + 1) - min);

        if (count == 1) {
            cout << "south" << endl;
        };
        if (count == 2) {
            cout << "west" << endl;
        };
        if (count == 3) {
            cout << "east" << endl;
        };
        if (count == 4) {
            cout << "north" << endl;
        };
        if (count == 5) {
            inCon->setState(new near());
        };

    }
};

void near::Display(Packetinfo* inCon) {
    cout << rand() << "." << rand() << "." << rand() << endl;
    inCon->setState(new direction());
}

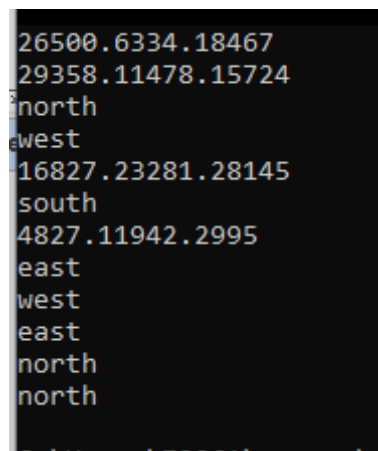
int main() {
    State* A = new near();
    State* B = new direction();

    Packetinfo* c = new Packetinfo(B);

    for (int i = 0; i < 18; i++) {
        c->Display();
    }

    delete A, B, c;
    return 0;
}

```



```

26500.6334.18467
29358.11478.15724
north
west
16827.23281.28145
south
4827.11942.2995
east
west
east
north
north

```

Рис. 1. Лог работы программы (State)

Observer

```
#include <iostream>
using namespace std;

class Observer {
public:
    virtual void update(int v) = 0; // данные через v в update(v)
};

// конкретный наблюдатель - Scanner
class Scanner : public Observer {
private:
    int oldTime, curTime;
    //Анализ данных
    void check() {
        cout << "Time " << curTime;
        if (oldTime == curTime)
            cout << " hasn't changed " << endl;
        else {
            if (oldTime < curTime) // обработка 1
                cout << " increased by " << curTime - oldTime << "
milliseconds " << endl;
            else // обработка 2
                cout << " decreased by " << oldTime - curTime << "
milliseconds " << endl;
        }
    }
public:
    //Обновление данных
    void update(int v) {
        oldTime = curTime;
        curTime = v;
        if (oldTime >= 0) check();
    }
    Scanner() { curTime = -1; }
};

class Subject {
protected:
    Observer* myObserver;
public:
    void attach(Observer* a) { myObserver = a; }
    void detach(Observer* a) {
        if (a == myObserver) myObserver = nullptr;
    }
    virtual void notify() = 0;
    Subject() {}
    ~Subject() {}
};

// LineUp - конкретный наблюдаемый объект
class LineUp : public Subject {
private:
    int t;
public:
    LineUp(int start) { t = start; }
    ~LineUp() {}
    void notify() { myObserver->update(t); }
    int getState() { return t; }
    void setState(int newX) {
        t = newX;
        notify();
    }
};
```

```

    }
};

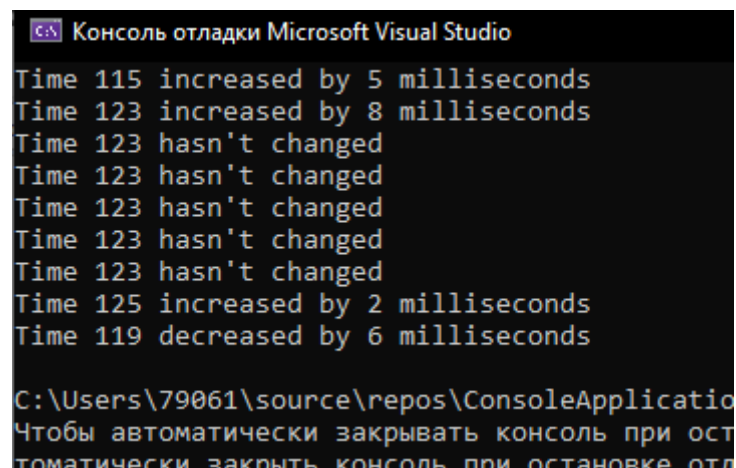
int main() {
    srand(time(NULL));
    Observer* obs = new Scanner();
    LineUp* LineU = new LineUp(110);

    LineU->attach(obs);

    //Генерация времени
    for (int i = 0; i < 10; i++) {
        if (i % 3)
            LineU->setState(LineU->getState() + i * rand() % 10);
        else
            LineU->setState(LineU->getState() - i * rand() % 10);
    }

    delete obs;
    delete LineU;
    return 0;
}

```



Консоль отладки Microsoft Visual Studio

```

Time 115 increased by 5 milliseconds
Time 123 increased by 8 milliseconds
Time 123 hasn't changed
Time 123 hasn't changed
Time 123 hasn't changed
Time 123 hasn't changed
Time 123 hasn't changed
Time 125 increased by 2 milliseconds
Time 119 decreased by 6 milliseconds

C:\Users\79061\source\repos\ConsoleApplicatio
Чтобы автоматически закрывать консоль при ост
томатически закрыть консоль при остановке отл

```

Рис. 2. Лог работы программы (Observer)