МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Реализация простейшей иерархии классов

Лабораторная работа №2

по дисциплине «Объекно-ориентированное программирование»

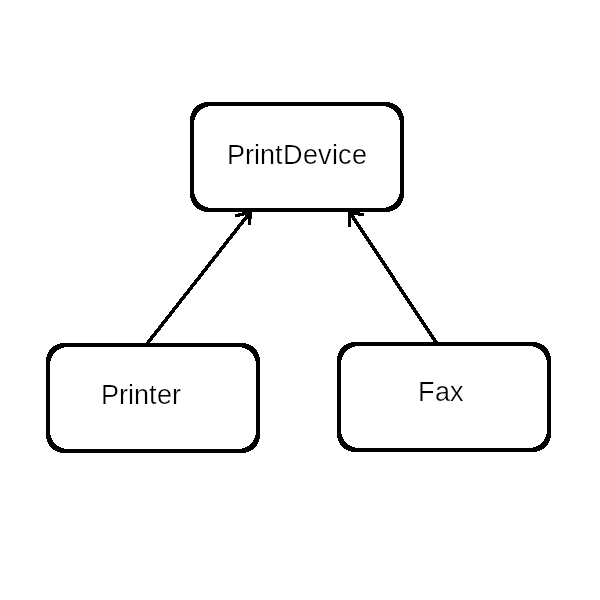
Выполнил студент Чекулаев В. Ю.

Факультет, группа ФКФН, ПО(аб)-81

Проверил Федосеев А. А.

Хабаровск – 2020г.

1. **Описание иерархии классов**



1. **Описание классов**

Имя:

*PrintDevice;*

Методы:

*PrintDevice()* - конструктор;

*virtual ~PrintDevice()* - деструктор;

*virtual void print() const* — чистый виртуальный метод;

*virtual std::string get\_info() const* — чистый виртуальный метод;

Имя:

*Printer (наследник PrintDevice)*

Свойства:

*std::string m\_name;*

Методы:

*Printer(std::string name = "Printer")* - конструктор;

*~Printer()* - деструктор;

*virtual void print() const override* - метод состояния печати;

*virtual std::string get\_info() const override* - геттер метода m\_name;

Имя:

*Fax (наследник PrintDevice)*

Свойства:

*std::string m\_name;*

Методы:

*Fax (std::string name = "Fax ")* - конструктор;

*~Fax ()* - деструктор;

*virtual void print() const override* - метод состояния печати;

*virtual std::string get\_info() const override* - геттер метода m\_name;

virtual std::string send\_data() - метод состояния отправки информации

virtual void get\_data() - метод состояния принятия информации;

1. **Содержимое заголовочного файла devices.h**

#ifndef DEVICES.H

#define DEVICES.H

#include <iostream>

#include <string>

class PrintDevise{

public:

PrintDevise(){}

virtual ~PrintDevise(){}

virtual void print() const = 0;

virtual std::string get\_info() const = 0;

};

class Printer : public PrintDevise{

public:

Printer(std::string name = "Printer") : PrintDevise(), m\_name(name){}

~Printer(){}

void print() const override { std::cout << m\_name << " is printing\n"; }

std::string get\_info() const override { return m\_name; }

private:

std::string m\_name;

};

class Fax : public PrintDevise{

public:

Fax(std::string name = "Fax") : PrintDevise(), m\_name(name){}

~Fax(){}

void print() const override { std::cout << m\_name << " is printing\n"; }

std::string get\_info() const override { return m\_name; }

virtual std::string send\_data(){ std::cout << m\_name << " is sending data\n"; }

virtual void get\_data(){ std::cout << m\_name << " is getting data\n"; }

private:

std::string m\_name;

};

#endif

1. **Пример работы тестового приложения**

Содержание файла main.cpp:

#include <iostream>

#include «devices.h»

int main(){

PrintDevise\* parent = new Printer("Printer 1");

print\_info(parent);

parent→print();

Fax fax(std::string("Fax 1"));

delete parent;

parent = &fax;

std::cout << "\n";

print\_info(parent);

parent->print();

fax.get\_data();

fax.send\_data();

std::cout << "\n";

}

Вывод программы:

