МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №4

**«Наследование»**

Выполнил:

студент 3 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверил:

Козик И. Д.

**Цель работы**: изучить, научиться и разобраться, как работать с наследованием в C++, а также как создавать простейшие классы-наследники.

**Вариант №5**

Необходимо создать класс Window. В классе должны быть следующие поля: id (int), height (int), width (int), memoryNeeded (int), areAdministatorRightsGranted (boolean) и isShown (boolean). Требуется реализовать конструктор, задающий id и принимающий параметры height, width и areAdministatorRightsGranted, метод doWork, в котором будет изменяться значение memoryNeeded и метод showOrHide, меняющий значение переменной isShown.

Создать класс ConsoleWindow. В данном классе добавляются поля commandsList и color (int). Реализовать класс Command с абстрактным методом doAction. Реализовать минимум 5 различных наследников от него для разных команд (копирование, удаление, смена активной папки и т.д.) Список доступных команд подгружается при создании. Сделать несколько различных объектов одной команды с разными параметрами. Реализовать вызов команды по её имени. MemoryNeeded должно передаваться из вызываемой команды. В основной программе необходимо реализовать логику использования всех созданных команд.

**Код программы:**

**Window.h**

#pragma once

class Window

{

protected:

int id;

int height;

int width;

int memoryNeeded;

bool areAdministratorRightsGranted;

bool isShown;

public:

Window(int id, int height, int width, bool areAdminRightsGranted)

: id(id), height(height), width(width), memoryNeeded(0),

areAdministratorRightsGranted(areAdminRightsGranted), isShown(false) {}

void doWork(int memoryNeeded) {

this->memoryNeeded += memoryNeeded;

}

void showOrHide() {

isShown = !isShown;

}

};

**ConsoleWindow.h**

#pragma once

#include <string>

#include <vector>

#include "Window.h"

#include "Command.h"

using namespace std;

class ConsoleWindow : public Window

{

protected:

vector<Command\*> commandsList = { new CopyCommand(), new CopyCommand("xcopy", 5), new DeleteCommand(), new MoveFileCommand(), new RenameFileCommand(), new CreateFileCommand() };

int color;

public:

ConsoleWindow(int id, int height, int width, bool areAdminRightsGranted, int color)

: Window(id, height, width, areAdminRightsGranted), color(color) {}

~ConsoleWindow() {

for (auto command : commandsList) {

delete command;

}

}

void executeCommand(const string commandName) {

for (auto command : commandsList) {

if (command->getName() == commandName) {

this->doWork(command->doAction());

return;

}

}

cout << "Command '" << commandName << "' is unknown.\n";

}

void printAvailableCommands() {

cout << "Available Commands:\n";

for (auto command : commandsList) {

cout << "\t" << command->getName() << "\n";

}

}

void printMemoryUsed() {

cout << "Memory used: " << memoryNeeded << "\n";

}

};

**Command.h**

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Command

{

protected:

string name;

int memoryNeeded;

public:

Command(const string name, int memoryNeeded)

: name(name), memoryNeeded(memoryNeeded) {}

// this virtual method ( = 0 ) makes this class abstruct and creates a need to override this method

virtual int doAction() = 0;

string getName() {

return name;

}

};

class CopyCommand : public Command

{

public:

CopyCommand(const string name, int memoryNeeded)

: Command(name, memoryNeeded) {}

CopyCommand()

: Command("copy", 10) {}

int doAction() override {

cout << "Copying data. Memory needed: " << memoryNeeded << "\n";

return memoryNeeded;

}

};

class DeleteCommand : public Command

{

public:

DeleteCommand(const string name, int memoryNeeded)

: Command(name, memoryNeeded) {}

DeleteCommand()

: Command("del", 0) {}

int doAction() override {

cout << "Deleting data. Memory needed: " << memoryNeeded << "\n";

return memoryNeeded;

}

};

class MoveFileCommand : public Command

{

public:

MoveFileCommand(const string name, int memoryNeeded)

: Command(name, memoryNeeded) {}

MoveFileCommand()

: Command("move", 30) {}

int doAction() override {

cout << "Moving file. Memory needed: " << memoryNeeded << "\n";

return memoryNeeded;

}

};

class RenameFileCommand : public Command

{

public:

RenameFileCommand(const string name, int memoryNeeded)

: Command(name, memoryNeeded) {}

RenameFileCommand()

: Command("rename", 12) {}

int doAction() override {

cout << "Renaming file. Memory needed: " << memoryNeeded << "\n";

return memoryNeeded;

}

};

class CreateFileCommand : public Command

{

public:

CreateFileCommand(const string name, int memoryNeeded)

: Command(name, memoryNeeded) {}

CreateFileCommand()

: Command("create", 23) {}

int doAction() override {

cout << "Creating file. Memory needed: " << memoryNeeded << "\n";

return memoryNeeded;

}

};

**main.cpp**

/\*

\* Variant #5

\*/

#include <iostream>

#include "ConsoleWindow.h"

int main()

{

ConsoleWindow w(1, 250, 400, false, 0x000000);

w.printAvailableCommands();

cout << "\n";

w.executeCommand("xcopy");

w.executeCommand("copy");

w.executeCommand("move");

w.executeCommand("delete");

w.executeCommand("del");

w.executeCommand("rename");

w.executeCommand("create");

cout << "\n";

w.printMemoryUsed();

return 0;

}

**Пример работы программы:**

**Available Commands:**

**copy**

**xcopy**

**del**

**move**

**rename**

**create**

**Copying data. Memory needed: 5**

**Copying data. Memory needed: 10**

**Moving file. Memory needed: 30**

**Command 'delete' is unknown.**

**Deleting data. Memory needed: 0**

**Renaming file. Memory needed: 12**

**Creating file. Memory needed: 23**

**Memory used: 80**