

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”  
**ИНТЕЛЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

**ОТЧЁТ**  
по лабораторной работе №4  
**“Наследование.”**

Выполнил:  
студент группы ПО-9  
Тусюк Т.В.

Проверил:  
Козик И.Д.

Брест 2023

## Лабораторная работа №5

### Вариант 22

**Цель работы:** научиться создавать простейшие классы-наследники.

#### Основное содержание работы.

Для выполнения лабораторной работы 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.

Затем необходимо создать класс-наследник класса Window по варианту, реализующий имитацию заданной функциональности.

Работу выполнять на языке C++.

**Задание №2:** Создать класс GameWindow. В данном классе добавляется поле settings. Settings реализовать отдельным классом с минимум 5 полями настроек. В классе должен быть абстрактный метод doAction. Реализовать 3 различных класса-наследника от класса GameWindow (например, RockPaperScissorsGameWindow) и столько же классов-наследников Settings (свой для каждого класса, который передаётся ему в конструкторе). Переопределить в каждом классе-наследнике doAction с вызовом doWork (высчитывать необходимый memoryNeeded исходя из настроек и самого doAction). В основной программе реализовать функцию запуска всех 3 наследников GameWindow.

#### Текст программы:

```
#include <iostream>

class Window
{
public:
    int id;
    int height;
    int width;
    int memoryNeeded;
    bool areAdministatorRightsGranted;
    bool isShown;

public:
    Window(int id, int height, int width, bool areAdministatorRightsGranted)
        : id(id), height(height), width(width), memoryNeeded(0),
        areAdministatorRightsGranted(areAdministatorRightsGranted), isShown(false) {}

    void doWork(int calculatedMemory)
    {
        memoryNeeded += calculatedMemory;
    }
}
```

```

        void showOrHide()
        {
            isShown = !isShown;
        }
};

class Settings
{
public:
    int setting1;
    int setting2;
    int setting3;
    int setting4;
    int setting5;

public:
    Settings(int setting1, int setting2, int setting3, int setting4, int setting5)
        : setting1(setting1), setting2(setting2), setting3(setting3),
        setting4(setting4), setting5(setting5) {}

};

class GameWindow
{
public:
    // Конструктор
    GameWindow(int id, int height, int width, bool areAdministratorRightsGranted, const
    Settings& settings)
        : window(id, height, width, areAdministratorRightsGranted), settings(settings) {}

    virtual void doAction() = 0;

protected:
    Window window;
    Settings settings;
};

class RockPaperScissorsSettings : public Settings
{
public:
    RockPaperScissorsSettings(int _setting1, int _setting2, int _setting3, int
    _setting4, int _setting5) : Settings(_setting1, _setting2, _setting3, _setting4,
    _setting5) {}
};

class RockPaperScissorsGameWindow : public GameWindow
{
public:
    RockPaperScissorsGameWindow(int id, int height, int width, bool
    areAdministratorRightsGranted, const Settings& settings)
        : GameWindow(id, height, width, areAdministratorRightsGranted, settings) {}

    void doAction() override
    {
        int calculatedMemory = settings.setting1 + settings.setting2 + settings.setting3
+ settings.setting4 + settings.setting5 + window.height * window.width;
        window.doWork(calculatedMemory);
        std::cout << "Memory needed for RockPaperScissorsGameWindow: " <<
window.memoryNeeded << std::endl;
    }
};

class TicTacToeSettings : public Settings
{
public:
    TicTacToeSettings(int _setting1, int _setting2, int _setting3, int _setting4, int
    _setting5) : Settings(_setting1, _setting2, _setting3, _setting4, _setting5) {}
};

```

```

class TicTacToeGameWindow : public GameWindow
{
public:
    TicTacToeGameWindow(int id, int height, int width, bool
areAdministratorRightsGranted, const Settings& settings)
        : GameWindow(id, height, width, areAdministratorRightsGranted, settings) {}

    void doAction() override
    {
        int calculatedMemory = settings.setting1 + settings.setting2 + settings.setting3
+ settings.setting4 + settings.setting5 + window.height * window.width;
        window.doWork(calculatedMemory);
        std::cout << "Memory needed for TicTacToeGameWindow: " << window.memoryNeeded <<
std::endl;
    }
};

class CountingTableSettings : public Settings
{
public:
    CountingTableSettings(int _setting1, int _setting2, int _setting3, int _setting4,
int _setting5) : Settings(_setting1, _setting2, _setting3, _setting4, _setting5) {}
};

class CountingTableGameWindow : public GameWindow
{
public:
    CountingTableGameWindow(int id, int height, int width, bool
areAdministratorRightsGranted, const Settings& settings)
        : GameWindow(id, height, width, areAdministratorRightsGranted, settings) {}
    void doAction() override
    {
        int calculatedMemory = settings.setting1 + settings.setting2 + settings.setting3
+ settings.setting4 + settings.setting5 + window.height * window.width;
        window.doWork(calculatedMemory);
        std::cout << "Memory needed for CountingTableGameWindow: " <<
window.memoryNeeded << std::endl;
    }
};

void startAllWindows(RockPaperScissorsGameWindow& rps, TicTacToeGameWindow& ttt,
CountingTableGameWindow& ct) {
    rps.doAction();
    ttt.doAction();
    ct.doAction();
}

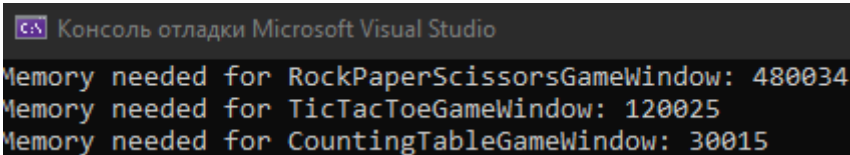
int main()
{
    Settings RockPaperScissorsSettings(10, 9, 5, 3, 7);
    Settings TicTacToeSettings(8, 4, 9, 3, 1);
    Settings CountingTableSettings(1, 2, 3, 4, 5);

    RockPaperScissorsGameWindow RockPaperScissors(1, 800, 600, true,
RockPaperScissorsSettings);
    TicTacToeGameWindow TicTacToe(1, 400, 300, true, TicTacToeSettings);
    CountingTableGameWindow CountingTable(1, 200, 150, true, CountingTableSettings);

    startAllWindows(RockPaperScissors, TicTacToe, CountingTable);
    return 0;
}

```

## Результат программы:



```

Консоль отладки Microsoft Visual Studio
Memory needed for RockPaperScissorsGameWindow: 480034
Memory needed for TicTacToeGameWindow: 120025
Memory needed for CountingTableGameWindow: 30015

```

**Вывод:** я научился создавать простейшие классы-наследники.