



ВВЕДЕНИЕ В
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ

- Свойства
- Поведение (функция)

Абстрагирование – это выделение каких-то существенных данных, необходимых для решения конкретной задачи и абстрагирование от остальных.

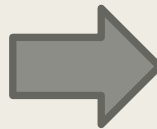
Домашнее задание

Придумать 3 примера объекта и каждый объект рассмотреть с не менее 3-х точек зрения (его свойства + поведение)

Кирилл Смирнов, студент 2 курса ф-та КНиИТ



- Свойства
- Поведение (функция)



В ООП:

- Поля
- Методы

Основные принципы ООП:

- ИНКАПСУЛЯЦИЯ
- НАСЛЕДОВАНИЕ
- ПОЛИМОРФИЗМ

ИНКАПСУЛЯЦИЯ

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.с

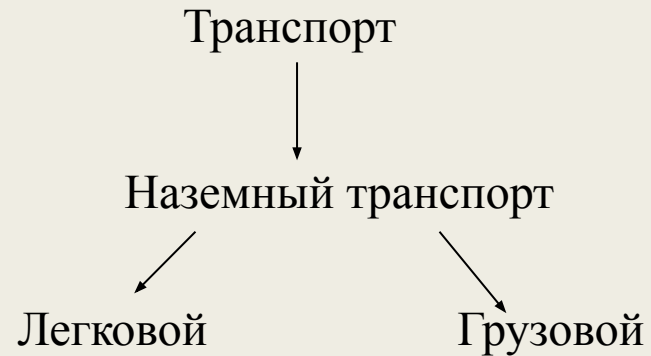


НАСЛЕДОВАНИЕ

Наследование – это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым или родительским. Новый класс – потомком, наследником или производным классом.



ИЕРАРХИЯ НАСЛЕДОВАНИЯ



Домашнее задание

Придумать 3 примера наследования и для каждого примера расписать не меньше 3 уровней

ПОЛИМОРФИЗМ

Полиморфизм – это возможность использовать в различных классах иерархии одного имени для обозначения сходных по смыслу действий и гибко выбирать требуемое действие во время выполнения программы.



Классы и объекты

```
class <имя класса> //имя класса должно быть осмысленным и прописано с заглавной буквы
{
<тело класса>
}; //в конце обязательна точка с запятой
```

```
class Students {
string secondname; //фамилия
string firstname; //имя студента
int grade; //курс
float average_ball; //итоговая оценка за семестр
};
```

```
4 #include "pch.h"
5 #include <iostream>
6 #include <string>
7
8
9 using namespace std;
10
11 class Student
12 {
13 public:
14     string secondname; //фамилия
15     string firstname; //имя студента
16     int grade; //курс
17     float grade_point_studies; //средний балл за учебу
18 };
19
20 int main()
21 {
22     setlocale(LC_ALL, "Russian");
23     Student first_student; //объект класса
24
25     //обращение к полям класса Student
26     // 1 способ присваивания значения полям
27     first_student.secondname = "Васильев";
28     first_student.firstname = "Андрей";
29     first_student.grade = 2;
30     first_student.grade_point_studies = 4.67;
31     cout << first_student.secondname << " " << first_student.firstname << " " << first_student.grade << " " << first_student.grade_point_studies << endl;
32     // 2 способ присваивания значения полям
33     Student second_student {"Ульянова", "Светлана", 3, 4.8};
34     cout << second_student.secondname << " " << second_student.firstname << " " << second_student.grade << " " << second_student.grade_point_studies << endl;
35
36 }
37
```

Вывод

Консоль отладки Microsoft Visual Studio

```
Васильев Андрей 2 4.67
Ульянова Светлана 3 4.8
```

Методы класса (функции)

```
#include "pch.h"  
#include <iostream>  
#include <string>
```

```
using namespace std;
```

```
class Student
```

```
{  
public:  
    string secondname; //фамилия  
    string firstname; //имя студента  
    int grade; //курс  
    float grade_point_studies; //средний балл за учебу
```

метод Show, в которой
пропишем, каким
образом класс Student
будет выводить
информацию

```
void Show()  
{  
    cout << "Фамилия: " << secondname << "\nИмя: " << firstname << "\nКурс: " << grade << "\nСредний балл обучающегося: " << grade_point_studies << endl;  
};
```

```
int main()  
{  
    setlocale(LC_ALL, "Russian");  
  
    Student first_student;  
    first_student.secondname = "Васильев";  
    first_student.firstname = "Андрей";  
    first_student.grade = 2;  
    first_student.grade_point_studies = 4.67;  
    first_student.Show();  
  
    Student second_student {"Ульянова", "Светлана", 3, 4.8};  
    second_student.Show();  
}
```

Вывод

Консоль отладки Microsoft Visual Studio

```
Фамилия: Васильев  
Имя: Андрей  
Курс: 2  
Средний балл обучающегося: 4.67  
Фамилия: Ульянова  
Имя: Светлана  
Курс: 3  
Средний балл обучающегося: 4.8
```

Модификаторы доступа классов

- Public
- Private
- Protected

Поля помечены как private

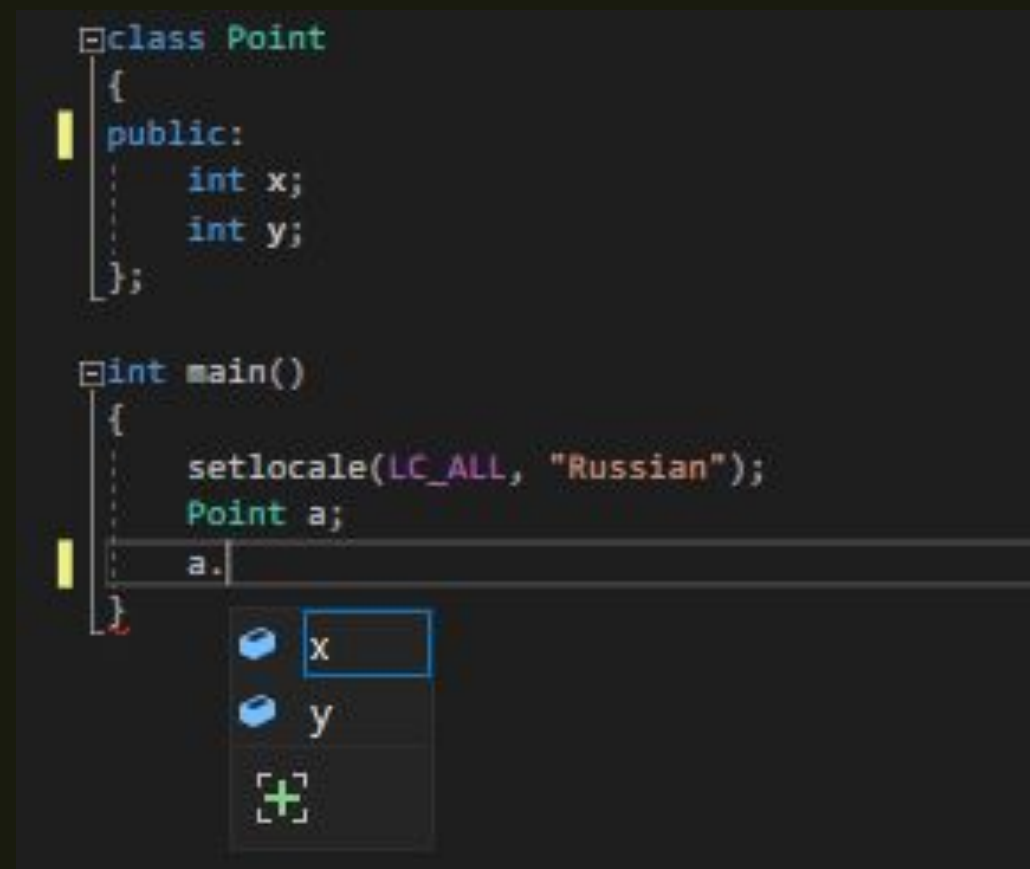
```
class Point
{
    //public:
    int x;
    int y;
};

int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.
```

Поля помечены как public

```
class Point
{
    public:
        int x;
        int y;
};

int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.
```



```
class Point
{
public:
    int x;
private:
    int y;
    int z;
};
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.x = 5;
    a.
```

x public : int Point::x
Файл: ConsoleApplication25.cpp

```
class Point
{
public:
    int x;
protected:
    int y;
    int z;
};
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.x = 5;
    a.
```

x public : int Point::x
Файл: ConsoleApplication25.cpp

```

class Point
{
public:
    int x;

    void Show()
    {
        cout << "Координата x: " << x << "\nКоордината y: " << y << "\nКоордината z: " << z;
    }
private:
    int y;
    int z;
};

int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.Show();
}

```

```

class Point
{
public:
    int x;

private:
    int y;
    int z;

    void Show()
    {
        cout << "Координата x: " << x << "\nКоордината y: " << y << "\nКоордината z: " << z;
    }
};

int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.Show();
}

```

void Point::Show()

функцию "Point::Show" (объявлено в строке 53) недоступно

```
class Point
{
public:
    int x;

    void Show()
    {
        cout << "Координата x: " << x << "\nКоордината y: " << y << endl;

        ShowZ();
    }

private:
    int y;
    int z;

    void ShowZ()
    {
        cout << "Координата z: " << z;
    }
};

int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.Show();
}
```


Геттеры и сеттеры

```
class Point
{
public:
    int x;
    int y;

public:
    int GetX()
    {
        return x;
    }

    void SetX(int valueX)
    {
        x = valueX;
    }

    int GetY()
    {
        return y;
    }

    void SetY(int valueY)
    {
        y = valueY;
    }
};
```

Консоль отладк

```
X = 5
Y = 10
```

```
class Point
{
public:
    int x;
    int y;

public:
    int GetX()
    {
        return x;
    }

    void SetX(int valueX)
    {
        x = valueX;
    }

    int GetY()
    {
        return y;
    }

    void SetY(int valueY)
    {
        y = valueY;
    }

    void Show()
    {
        cout << "X= " << x << "Y= " << y;
    }
};
```

Консоль отладк

```
X= 5
Y= 10
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.SetX(5);
    a.SetY(10);
    int resultX = a.GetX();
    int resultY = a.GetY();
    cout << "X = " << resultX << "\nY = " << resultY;
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
    a.SetX(5);
    a.SetY(10);

    a.Show();
}
```

КОНСТРУКТОР (ПРИСВОЕНИЕ В ТЕЛЕ КОНСТРУКТОРА)

```
class Point
{
private:
    int x;
    int y;

public:
    //конструктор по умолчанию
    Point()
    {
    }
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
}
```

Вывод

```
Консоль отладки Micro:
X= -858993460
Y= -858993460
```

```
class Point
{
private:
    int x;
    int y;

public:
    //конструктор
    Point(int valueX, int valueY)
    {
        x = valueX;
        y = valueY;
    }
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a;
}
```

для класса "Point" не существует конструктор по умолчанию

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a (5, 10);

    a.Show();
}
```

Вывод

```
Консоль отладки Micro:
X= 5
Y= 10
```

КОНСТРУКТОР (ИНИЦИАЛИЗАЦИЯ)

```
class Point
{
private:
    int x;
    int y;

public:
    Point(int valueX, int valueY):x(valueX), y(valueY)
    {
    }

    ///конструктор
    //Point(int valueX, int valueY)
    //{
    //    x = valueX;
    //    y = valueY;
    //}
}
```

//без параметров

```
Point ():x(0), y(0)
{
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Point a (5, 10);

    a.Show();
}
```

Вывод

```
Консоль от
X= 5
Y= 10
```

Домашнее задание

Методичка с 23-27, практикум №1, создать класс **без** статического поля для подсчета количества экземпляров класса и **без** перегрузок.

```
class Fraction
{
private:
    int num;
    int den;

public:
    Fraction(int valueX, int valueY):num(valueX), den(valueY)
    {
        ...
    }

    bool operator ==(Fraction a)
    {
        return (num * a.den == a.num * den);
    }
}
```

```
int main()
{
    setlocale(LC_ALL, "Russian");
    Fraction a (1, 4);
    Fraction b(8, 32);

    bool result = a == b;
}
```

Если в задании необходимо
сравнить что-то!

Спасибо за внимание!