

ЧЕТВЕРТОЕ ЗАНЯТИЕ

ООП

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

ПРИНЦИПЫ ООП

- Инкапсуляция
 - Как данный язык скрывает детали внутренней реализации объектов и предохраняет целостность данных?
- Наследование
 - Как данный язык стимулирует многократное использование кода?
- Полиморфизм
 - Как данный язык позволяет трактовать связанные объекты сходным образом?
- Абстракция
 - Как данный язык позволяет отделять способы использования объектов от их конкретной реализации?

КЛАССЫ И ОБЪЕКТЫ

КЛАСС

КЛАСС ПРЕДСТАВЛЯЕТ СОБОЙ ШАБЛОН, ПО КОТОРОМУ ОПРЕДЕЛЯЕТСЯ ФОРМА ОБЪЕКТА. В НЕМ УКАЗЫВАЮТСЯ ДАННЫЕ И КОД, КОТОРЫЙ БУДЕТ ОПЕРИРОВАТЬ ЭТИМИ ДАННЫМИ. В С# ИСПОЛЬЗУЕТСЯ СПЕЦИФИКАЦИЯ КЛАССА ДЛЯ ПОСТРОЕНИЯ ОБЪЕКТОВ, КОТОРЫЕ ЯВЛЯЮТСЯ ЭКЗЕМПЛЯРАМИ КЛАССА.



ПРОСТЕЙШИЙ КЛАСС

CLASS SAMPLE

{

}

СОЗДАНИЕ ЭКЗЕМПЛЯРА
(ОБЪЕКТА)

```
SAMPLE SAMPLEVAR = NEW SAMPLE();
```

CLASS BOOK

{

ПОЛЯ КЛАССА

PUBLIC STRING NAME;

PUBLIC STRING AUTHOR;

PUBLIC INT YEAR;

}


```
CLASS BOOK
```

```
{
```

```
...
```

МЕТОД ДЛЯ РАБОТЫ
С ДАННЫМИ

```
PUBLIC VOID INFO()
```

```
{
```

```
    CONSOLE.WRITELINE(
```

\$ ИСПОЛЬЗУЕТСЯ ДЛЯ ПОДСТАНОВКИ
ЗНАЧЕНИЙ

```
        $"BOOK {NAME} AUTHOR {AUTHOR} YEAR -  
        {YEAR}");
```

```
    }
```

```
}
```

КОНСТРУКТОР.
ПРЕДНАЗНАЧЕН
ДЛЯ ИНИЦИАЛИЗАЦИИ
ПЕРЕМЕННЫХ

ВАЖНО!

Конструктор не имеет
возвращаемого типа!

Даже не void

```
CLASS Book
{
    ...
    PUBLIC Book()
    {
    }
    PUBLIC Book(STRING NAME, STRING AUTHOR, INT YEAR)
    {
        NAME = NAME;
        AUTHOR = AUTHOR;
        YEAR = YEAR;
    }
}
```

СТАТИЧНЫЕ ПОЛЯ

Если нам необходимо определить поведение не для конкретного объекта книги, а для всего класса "Книга" мы можем использовать ключевое слово `static`. Если мы определяем поля, методы или свойства как статические, то они существуют не на уровне конкретного объекта, а на уровне всего класса.

Таким образом мы можем обращаться к ним так:

`<Название класса>.<Статическое поле/метод/свойство>`

МОДИФИКАТОРЫ ДОСТУПА

- PUBLIC: ПУБЛИЧНЫЙ, ОБЩЕДОСТУПНЫЙ КЛАСС ИЛИ ЧЛЕН КЛАССА. ТАКОЙ ЧЛЕН КЛАССА ДОСТУПЕН ИЗ ЛЮБОГО МЕСТА В КОДЕ, А ТАКЖЕ ИЗ ДРУГИХ ПРОГРАММ И СБОРОК.
- PRIVATE: ЗАКРЫТЫЙ КЛАСС ИЛИ ЧЛЕН КЛАССА. ПРЕДСТАВЛЯЕТ ПОЛНУЮ ПРОТИВОПОЛОЖНОСТЬ МОДИФИКАТОРУ PUBLIC. ТАКОЙ ЗАКРЫТЫЙ КЛАСС ИЛИ ЧЛЕН КЛАССА ДОСТУПЕН ТОЛЬКО ИЗ КОДА В ТОМ ЖЕ КЛАССЕ ИЛИ КОНТЕКСТЕ.
- PROTECTED: ТАКОЙ ЧЛЕН КЛАССА ДОСТУПЕН ИЗ ЛЮБОГО МЕСТА В ТЕКУЩЕМ КЛАССЕ ИЛИ В ПРОИЗВОДНЫХ КЛАССАХ.
- INTERNAL: КЛАСС И ЧЛЕНЫ КЛАССА С ПОДОБНЫМ МОДИФИКАТОРОМ ДОСТУПНЫ ИЗ ЛЮБОГО МЕСТА КОДА В ТОЙ ЖЕ СБОРКЕ, ОДНАКО ОН НЕДОСТУПЕН ДЛЯ ДРУГИХ ПРОГРАММ И СБОРОК (КАК В СЛУЧАЕ С МОДИФИКАТОРОМ PUBLIC).
- PROTECTED INTERNAL: СОВМЕЩАЕТ ФУНКЦИОНАЛ ДВУХ МОДИФИКАТОРОВ. КЛАССЫ И ЧЛЕНЫ КЛАССА С ТАКИМ МОДИФИКАТОРОМ ДОСТУПНЫ ИЗ ТЕКУЩЕЙ СБОРКИ И ИЗ ПРОИЗВОДНЫХ КЛАССОВ.

СВОЙСТВА

В то время, как класс хранит в своих полях необходимые ему данные, он не должен предоставлять внешнему коду прямой доступ к ним. Он должен как-то ограждать свои данные либо методами либо **свойствами**

ОПРЕДЕЛЕНИЕ

ЕСТЬ ПРИВАТНОЕ ПОЛЕ

```
PRIVATE INT FIELD;
```

ТОЛЬКО КЛАСС, В КОТОРОМ ЭТО ПОЛЕ ОБЪЯВЛЕНО, ИМЕЕТ ДОСТУП К ДАННОМУ ПОЛЮ

ЧТОБЫ ПРЕДОСТАВИТЬ ДОСТУП К ПОЛЮ, МЫ МОЖЕМ ОБЕРНУТЬ ДАННОЕ ПОЛЕ В СВОЙСТВО

```
PUBLIC INT FIELD { GET{ RETURN FIELD;} SET{ FIELD = VALUE;} }
```

ВАЖНО ПОНИМАТЬ, ЧТО СВОЙСТВО – ЭТО ПРОСТО ПАРА ОБЫЧНЫХ МЕТОДОВ, ОФОРМЛЕННЫХ БОЛЕЕ ЭЛЕГАНТНО. НЕ БОЛЕЕ. И КРОМЕ СТРОКИ ДЛЯ ВОЗВРАЩЕНИЯ ЗНАЧЕНИЯ И ПРИСВАИВАНИЯ МЫ МОЖЕМ ПИСАТЬ ЧТО УГОДНО

АВТО СВОЙСТВА

Для удобства, если нам не надо совершать какие либо действия кроме изменения значения самого поля мы можем использовать автоматические свойства.

```
PUBLIC INT FIELD {GET; SET;}
```

Такой код сам сгенерирует приватное поле для хранения значения и методы, которые этим полем манипулируют.

Позже, если понадобится, мы сможем изменить логику поведения данного свойства на необходимую нам. При этом мы не затронем внешний код, который пользуется нашим классом.