11. **КЛАССЫ**

*Объект* - экземпляр класса.

Вы никогда не имеете прямого доступа к объекту, только через указатель (**объектную переменную**).

Класс - абстрактный тип данных, пользовательский тип, определяемый программистом исходя из модели абстракции.

Подразумевается, что на основе классов создаются *объекты*.

Абстракция - первый принцип ООП, набор значимых характеристик в контексте решаемой задачи.

В ООП всегда есть источник действия, и иногда правильнее, чтобы источником такого действия был сам *объект*:

marsel.go(30);

Структура класса в Java

class ИМЯ\_КЛАССА {

// поля (fields)

ТИП\_ДАННЫХ ИМЯ\_ПОЛЯ;

ТИП\_ДАННЫХ ИМЯ\_ПОЛЯ;

ТИП\_ДАННЫХ ИМЯ\_ПОЛЯ;

ТИП\_ДАННЫХ ИМЯ\_ПОЛЯ;

...

// конструкторы (constructors)

ИМЯ\_КЛАССА(ФОРМАЛЬНЫЕ\_ПАРАМЕТРЫ) {

...

}

ИМЯ\_КЛАССА(ФОРМАЛЬНЫЕ\_ПАРАМЕТРЫ) {

...

}

// методы (methods)

ТИП\_ДАННЫХ ИМЯ\_МЕТОДА(ФОРМАЛЬНЫЕ ПАРАМЕТРЫ) {

ТЕЛО\_МЕТОДА;

}

ТИП\_ДАННЫХ ИМЯ\_МЕТОДА(ФОРМАЛЬНЫЕ ПАРАМЕТРЫ) {

ТЕЛО\_МЕТОДА;

}

...

}

*Метод* - функция/процедура, объявленная внутри класса.

**Объект - экземпляр класса. Каждый объект обладает состоянием и поведением. Поведение - методы. Состояние - характеристики объекта, у каждого объекта они свои - поля. Конкретные значения полей каждого объекта определяют его состояние.**

Создание объектов

Создание объекта - это процесс выделения памяти для объекта, задание начальных характеристик этому объекту (инициализация) и присваивание указателя на этот объект объектной переменной.

НазваниеКласса имяПеременной = new НазваниеКласса()

- создается объектная переменная

- new - оператор выделения памяти, по факту создает объект

- НазваниеКласса() - **конструктор**.

= - присваивание.

*Конструктор* - набор инструкций (специальный метод - но это неверно). Выполняет инициализацию полей объекта (задает начальные значения полям). Если конструктор не был определен для класса, то его автоматически создает Java. При этом такой конструктор не содержит ничего, и называется "конструктором по умолчанию".

Вы можете описать свой *конструктор* - тогда, конструктор по умолчанию не будет создан.

Вы можете перегрузить конструктор - то есть объявить *конструкторы* с разной сигнатурой (разными формальными параметрами).

Можно выделить следующие типы *конструкторов* - по умолчанию, пустой, с параметрами, копирования.

*this* - объектная переменная, которая находится внутри объекта и содержит ссылку на этот самый объект.

Инкапсуляция

Инкапсуляция (в капсуле) - объект - это "капсула", внутри которой описывается как поведение, так и состояние. В отличие от структур в C/C++, где описывается ТОЛЬКО состояние.

Первое определение инкапсуляции - возможность объединения алгоритмов и структур данных в одной сущности - класса, и на основе класса создавать объект, обладающий состоянием и поведением.

Второе определение инкапсуляции - возможность закрытия внутренней структуры данных, с целью поддержания более "корректного" кода. То есть мы ограничиваем возможности по использованию наших классов.

Если не "защищать" внутреннюю структуру объекта, то возможны ситуации нарушения семантической целостности объекта. Следовательно, необходимо реализовать "защиту" внутренней структуры и разрешать производить только корректные изменения состояния объекта.

В Java это достигается за счет использования *модификаторов доступа*, обеспечивающих различный уровень доступа.

**private** - модификатор доступа, который закрывает возможность работы с членом класса вне этого класса. Данный модификатор обеспечивает уровень доступа "закрытый".

Но, что делать, если мы все-таки хотим работать с полем, но обеспечить целостность данных внутри поля? Используем - *методы доступа*.

Методы доступа - не являются синтаксической и обязательной единицей в Java. Они просто признак качественного ООП кода. Методы доступа позволяют обеспечить контролируемый доступ над полем.

Выделяют методы доступа - геттеры и сеттеры.

Геттеры имеют вид:

public ТИП\_ПОЛЯ getНазваниеПоля() {

return названиеПоля;

}

Геттеры позволяют прочитать значение.

Сеттеры имеют вид:

public void setИмяПоля(ТИП\_ПОЛЯ имяПоля) {

// тело сеттера

}

**public** - модификатор доступа, позволяющий работать с классом в любом участке кода, к которому подключен данный класс. Обеспечивает уровень доступа "Открытый"

Принято помечать:

- все поля private;

- все методы доступа public;

- все методы, предназначенные для внешнего использования – public;

- все методы, предназначенные для внутреннего использования - private.

Если модификатор не указан, то член класса или конструктор доступен внутри данного пакета. Уровень доступа - пакетный (package internal).

Статические члены класса

- члены класса (поля, методы, константы) помеченные модификатором **static**.

- инициализаторы (конструкторы), но объявленные иначе.

Статические поля - поля, глобальные для всех объектов данного класса. То есть если в одном объекте меняется значение этого поля, то оно меняется разом у всех объектов. (Правильнее - такое поле только одно, и оно доступно всем объектам).

Со статическими полями можно (нужно) работать только через класс.

В статическом инициализаторе можно работать ТОЛЬКО со СТАТИЧЕСКИМИ ПЕРЕМЕННЫМИ.

Когда нужны статические переменные? - Когда необходимо иметь какой-либо глобальный контекст для объектов, например - направление ветра, в программировании - "подключение к базе данных".

Статические константы - если объявляете константу, то всегда делайте ее статической.

Статические методы - методы, относящиеся к классу, их можно вызывать непосредственно из класса.

Удобно использовать тогда, когда поведение такого метода не завязано на конкретном объекте, а является поведением предметной области.

Статически метод может работать со статическими полями и методами, но не может работать с нестатическими полями и методами.

Отношения между классами

Агрегирование - связь между двумя объектами, построенная таким образом, что один объект содержит поле-ссылку на другой объект.