# Лекция 4. Классы

# Понятие класса

**Класс** – это тип данных, являющийся отображением сущностей, явлений или процессов предметной области, который может содержать свойства этого явления и характерное поведение.

**Объект** (instance, экземпляр класса) – конкретное отображение класса в памяти, обладающее определённым состоянием (заданными значениями свойств) и поведением.

*(!) Понятие класса неразрывно связано с объектно-ориентированной парадигмой программирования, согласно которой программа – это совокупность взаимодействующих объектов, а объекты в свою очередь – экземпляры некоторых классов.*

Иначе говоря, класс – это «шаблон» для создания объекта. Класс определяет **структуру (свойства, характеристики)** объекта и **действия (поведение)**, которые можно произвести с объектом.

# Структура класса

Как сказано выше, класс – это шаблон, описывающий любое явление, процесс или сущность некоторой предметной области. Описываемое явление имеет свои характеристики и поведение. В объектно-ориентированном программировании характеристики явления – это **поля** класса, а поведение – **методы**.

Создание полей класса – это процесс описания всех свойств будущих объектов. Проектирование полей классов требует анализа предметной области с целью разделения (декомпозиции) сущностей. Полей в классе должно быть достаточно, чтобы получить всю необходимую информацию об объекте, который будет создан на основе данного класса. Совокупность конкретных значений полей объекта в момент времени называется **состоянием**.

*(!) Стоит заметить, что существуют классы, у которых не определено ни одного поля (например, классы-конверторы, которые на вход метода преобразования получают объект для преобразования и возвращают преобразованный объект – таким классам совсем не обязательно иметь характеристики), однако, в момент исполнения программы объекты таких классов тоже находятся в определенном состоянии.*

Описание методов класса – это процесс проектирования действий, которые можно будет произвести над объектом данного класса в момент исполнения программы. Методов, как и полей, может быть много, а может не быть вообще. Таким образом, методы изменяют состояние объекта в момент исполнения программы.

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Класс** | | ***Поля*** | | ***Методы*** | |
| Рис.1. Структура класса |

Рассмотрим в качестве сущности для проектирования класса собак. У собаки можно выделить следующие свойства:

* имя;
* возраст;
* название породы.

Какие действия можно произвести с собакой? Например, «назвать» ее, «узнать» имя; собаки также могут гавкать и пр.

Таким образом, можно спроектировать класс следующего вида (см. рис. 2).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Собака** | | ***Поле:*** имя | | ***Поле:*** возраст | | ***Поле:*** название породы | | ***Метод:*** задать имя | | ***Метод:*** получить имя | | ***Метод:*** подать голос | |
| Рис. 2. Структура класса «Собака» |

# Объявление классов в Java

Для объявления класса в Java используется ключевое слово ***class***.

*(!) В файле исходного кода Java может быть объявлена один и более классов (но только один класс с модификатором доступа public), а может быть не объявлено ни одного.*

Объявление класса в Java выглядит следующим образом:

|  |
| --- |
| [модификатор\_доступа] **class** Имя\_класса{ /\* тело класса \*/} |
| Формула 1. Объявление класса в Java |

Например, объявление класса Собака в Java будет выглядеть следующим образом:

|  |
| --- |
| **public** **class** Dog  {  } |

где, согласно формуле 1, «***public***» - модификатор доступа, «***class***» - ключевое слово для объявления класса, «***Dog***» - имя шаблона (класса).

Существует несколько рекомендаций по именованию классов:

1. Имя класса – существительное.
2. Имя класса пишется с заглавной буквы (Address).
3. Если в имени класса несколько слов, то каждое следующее пишется с заглавной буквы (например, PhysicalAddress.).
4. В имени класса не следует использовать заглавный регистр, или символ нижнего подчеркивания.

**Нежелательно**: Postal\_Address, HTTPConvertor.

**Желательно**: PostalAddress, HttpConvertor.

1. Имя класса должно быть информативным и интуитивно понятным. Например, если класс называется Address, ожидается, что он содержит данные физического адреса, а не интернет-адрес. В таком случае, интернет-адрес лучше назвать InetAddress.

В теле класса последовательно перечисляются поля класса и методы.

Объявление полей класса происходит следующим образом.

|  |
| --- |
| [модификатор\_доступа] **тип\_данных** имя\_поля; |
| Формула 2. Объявление поля класса в Java |

Например, объявление полей класса Dog будет выглядеть следующим образом:

|  |
| --- |
| public class Dog {    String name; // имя  int age; // возраст  String breed;// порода  } |

где, согласно формуле 2, объявлено 3 поля – name типа *String*, age типа *int* и breed типа *String*.

Для того, чтобы обратиться к полю класса (например, чтобы задать значение этого поля, или наоборот использовать его значение в программе) необходимо указать имя переменной заданного типа и через точку имя поля класса (см. формулу 3).

|  |
| --- |
| имя\_переменной**.**имя\_поля; |
| Формула 3. Обращение к полю класса в Java |

В примерах ниже приведено две ситуации: в первой полю name задается значение (происходит инициализация этого поля), а во втором случае это значение используется:

|  |
| --- |
| **public** **class** Executor  {  **public** **static** **void** main(String[] args)  {  Dog dog=**new** Dog();  dog.name="Businka"; // задание значения полю name перменной dog типа Dog  String dogsName=dog.name; // задание значения поля name переменной dogsName типа String  } |
|  |

Существует несколько рекомендаций по именованию полей классов:

1. Имена полей классов – существительные, так как поле – это характеристика явления.
2. Имена полей классов начинаются с прописной буквы и если состоят из нескольких слов, то каждое следующее – с заглавной (использование нижнего подчеркивания также нежелательно).

Например: name, dogName, ownerName.

1. Если объявляется константа класса, то ее имя пишется только заглавными буквами, а в качестве разделителя используется нижнее подчеркивание: SIMPLE\_DOG, HUSKIES.
2. Если поле типа boolean, оно должно быть либо деепричастием, либо начинаться с «is» или «has»: initialized, isBroken, hasOwner.

Объявление методов класса показано в формуле 3:

|  |
| --- |
| [модификатор\_доступа] тип\_вовращаемого\_значения имя\_метода ([список\_параметров]) { /\* тело метода \*/} |
| Формула 4. Объявление метода класса в Java |

Для того, чтобы вызвать метод на объекте класса (тем самым изменить состояние этого объекта в памяти) необходимо узакать имя пременной заданного типа и через точку имя метода, который был объявлен в этом классе и передать в него все необходимые параметры, если такие требуются в методе (см. формулу 5).

|  |
| --- |
| имя\_переменной**.**имя\_метода([список параметров]); |
| Формула 5. Обращение к методу класса в Java |

В примере ниже продемострированны вызовы двух методов, один из которых в качестве параметров принимает переменную типа String (имя, которое будет задано собаке), а второй метод на вход не принимает никакие параметры, но возвращает результат работы метода – имя собаки.

|  |
| --- |
| **public** **class** Executor  {  **public** **static** **void** main(String[] args)  {  Dog dog=**new** Dog();  String dogsName="Businka";  dog.setName(dogsName); // задание имени собаки с помощью метода  String returnedDogsName=dog.getName(); // «взятие» имени собаки с помощью метода  }  } |

Существует несколько рекомендаций по именованию методов классов:

1. Имена методов начинаются глаголов, так как метод – это действие.
2. Имена методов классов начинаются с прописной буквы и если состоят из нескольких слов, то каждое следующее – с заглавной (использование нижнего подчеркивания также нежелательно).

Например: getName(), setValue, convert().

Добавим метод задания имени в классе Dog:

|  |
| --- |
| public class Dog {    String name; // имя  int age; // возраст  String breed;// порода  **// метод задания имени**  **public** **void** setName(String setName) {  name = setName;  }  } |

где, согласно формуле 3, метод с именем setName ничего не возвращает, а в качестве параметра получает то имя, которое необходимо задать собаке.

Добавим метод для получения текущего имени собаки:

|  |
| --- |
| public class Dog {    String name; // имя  int age; // возраст  String breed;// порода  // метод задания имени  **public** **void** setName(String setName) {  name = setName;  }  **// метод получения имени**  **public** String getName() {  **return** name;  }  } |

где, в отличие от метода setName(), метод getName не имеет никаких параметров и возвращает имя собаки.

*(!) Для возвращения из методов используется оператор return [параметр]. Оператор возврата используется для досрочного выхода из метода без возвращаемого значения (в этом случае не передается возвращаемое значение после ключевого слова return), и для возвращения значении из метода (return возвращаемое\_значение;).*

*(!) Обратите внимание, что тип возвращаемого значения оператора return должен быть таким же, как тип возвращаемого значения метода!*

Классы могут взаимодействовать друг с другом. Например, правильным решением, с точки зрения дизайна программы, было бы создание отдельного класса для запуска приложения, в котором будет создан объект типа Dog и в консоли будет выведено имя собаки, возраст и порода.

*(!) Класс является ссылочным типом данных в Java, следовательно, создание новых экземпляров возможно только с операцией new.*

|  |
| --- |
| **public** **class** Engine {  **public** **static** **void** main(String[] args) {  Dog businka = **new** Dog();    System.***out***.println("Dog state\_0: name=" + businka.getName() + "; age="+ businka.getAge() + "; breed=" + businka.getBreed());    businka.setName("Businka");  businka.setAge(**5**);  businka.setBreed("UNKNOWN");  System.***out***.println("Dog state\_1: name=" + businka.getName() + "; age="+ businka.getAge() + "; breed=" + businka.getBreed());  }  } |

Класс Engine – точка входа в приложение. В данном программном модуле происходит инициализация переменной businka типа Dog, выводится состояние объекта после инициализации. После состояние объекта меняется и снова выводится на экран.

В ходе инициализации с помощью оператора new в памяти был создан типа Dog, но ни одного значения поля задано не было. Поэтому, при выводе в консоль данных о businka в состоянии 0, все значения будут пустыми.

Далее последовательно объекту задаются значения имени, возраста и названия породы, после чего его состояние снова выводится на экран.

|  |
| --- |
|  |
| Рис. 3. Результат выполнения программы |

**Самостоятельно:**

Расширить программу, приведенную в качестве примера в лекции:

1. Добавить в Dog информацию о владельце (у собаки может быть только один хозяин).
2. «Научить» собаку лаять.
3. Создать сущность Владелец, описать его свойства и методы. У Владельца может быть до 5 собак.
4. Необходимо иметь возможность добавлять Владельцу собак, но только тех, у которых еще нет хозяина. Если добавление возможно, то после добавления собаки в список владельца, необходимо собаке выставить информацию о ее владельце.