**Статические переменные и методы**

[Java Syntax](https://javarush.ru/quests/QUEST_JAVA_SYNTAX)

[Уровень 6](https://javarush.ru/quests/lectures?quest=QUEST_JAVA_SYNTAX&level=6), Лекция 6

— Новая интересная тема. Хочу рассказать тебе о статических переменных и методах.

— О, я уже слышал про статические переменные. И про статические методы, кажется. Но хотелось бы больше подробностей.

— Когда мы описываем переменные в классе, мы указываем, будут ли эти переменные созданы всего один раз или же нужно создавать их копии для каждого объекта. По умолчанию создаётся новая копия переменной для каждого объекта. Вот как это выглядит:

Объявление класса

class Cat //класс

{

String name; //переменная

Cat(String name) //конструктор

{

this.name = name; //инициализация переменной

}

}

Код в методе main:

Cat cat1 = new Cat("Vaska"); //создали один объект, его name содержит строку «Vaska»

Cat cat2 = new Cat("Murka"); //создали один объект, его name содержит строку «Murka»

System.out.println(cat1.name);

System.out.println(cat2.name);

Вывод на экран будет таким:

Vaska

Murka

— Переменные cat1.name и cat2.name, хоть и описаны в одном классе – Cat, но хранят разные значения, т.к. привязаны к разным объектам.  
— Это понятно.  
— Статические же переменные – существуют в одном экземпляре, и обращаться к ним нужно по имени класса (внутри класса к статической переменной можно обращаться просто по имени):

Объявление класса

class Cat //класс

{

String name; //обычная переменная

static int catCount; //статическая переменная

Cat(String name)

{

this.name = name;

Cat.catCount++; //увеличиваем значение статический переменной на 1

}

}

Код в методе main:

System.out.println(Cat.catCount);

Cat cat1 = new Cat("Vaska");

System.out.println(Cat.catCount);

Cat cat2 = new Cat("Murka");

System.out.println(cat1.name);

System.out.println(cat2.name);

System.out.println(Cat.catCount);

Вывод на экран будет таким:

0

1

Vaska

Murka

2

— Это тоже понятно.

— Методы класса тоже делятся на две категории. **Обычные методы** вызываются у объекта и имеют доступ к данным этого объекта. **Статические методы** не имеют такого доступа – у них просто нет ссылки на объект, они способны обращаться либо к статическим переменным этого класса либо к другим статическим методам.

— Статические методы не могут обращаться к нестатическим методам или нестатическим переменным!

— А почему?

— Каждая обычная переменная класса находится внутри объекта. Обратиться к ней можно только имея ссылку на этот объект. В статический метод такая ссылка не передается.

— А в обычные методы передается?

— Да, в обычные методы передается, неявно. В каждый метод неявно передается ссылка на объект, у которого этот метод вызывают. Переменная, которая хранит эту ссылку, называется **this**. Таким образом, метод всегда может получить данные из своего объекта или вызвать другой нестатический метод этого же объекта.

— В статический метод вместо ссылки на объект передается null. Поэтому он не может обращаться к нестатическим переменным и методам – у него банально нет ссылки на объект, к которому они привязаны.

— Ясно.

— Так работают обычные нестатические методы:

|  |  |
| --- | --- |
| Как выглядит код  Cat cat = new Cat();  String name = cat.getName();  cat.setAge(17);  cat.setChildren(cat1, cat2, cat3); | Что происходит на самом деле  Cat cat = new Cat();  String name = Cat.getName(cat);  Cat.setAge(cat, 17);  Cat.setChildren(cat, cat1, cat2, cat3); |
| При вызове метода в виде «объект» точка «имя метода», на самом деле вызывается метод класса, в который первым аргументом передаётся тот самый объект. Внутри метода он получает имя this. Именно с ним и его данными происходят все действия. | |

— А вот как работают статические методы:

|  |  |
| --- | --- |
| Как выглядит код  Cat cat1 = new Cat();  Cat cat2 = new Cat();  int catCount = Cat**.getAllCatsCount**(); | Что происходит на самом деле  Cat cat1 = new Cat();  Cat cat2 = new Cat();  int catCount = Cat**.getAllCatsCount**(null); |
| При вызове статического метода, никакого объекта внутрь не передаётся. Т.е. this равен null, поэтому статический метод не имеет доступа к нестатическим переменным и методам (ему нечего неявно передать в обычные методы). | |

— Переменная или метод являются статическими, если перед ними стоит ключевое слово static.

— А зачем такие методы нужны, если они так сильно ограничены?

— У такого подхода тоже есть свои преимущества.

— Во-первых, для того, чтобы обратиться к статическим методам и переменным не надо передавать никакую ссылку на объект.

— Во-вторых, иногда бывает нужно, чтобы переменная была в единственном экземпляре. Как, например, переменная System.out (статическая переменная out класса System).

— И в третьих, иногда нужно вызвать метод, еще до того, как будет возможность создавать какие-то объекты

— Это когда же?

— А почему, по-твоему, метод main объявлен статическим? Чтобы его можно было вызвать сразу после загрузки класса в память. Еще до того, когда можно будет создавать какие-то объекты.