

---

QA  
מע

אלכס גורבצ'וב  
АЛЕКСЕЙ ГОРБАЧЁВ



---

JAVA  
JAVA



## Что такое ООП

Аббревиатура ООП расшифровывается как "Объектно-ориентированное программирование".

Всё вокруг нас является  
**объектом.**



У объекта есть  
**свойства**  
(еще называют параметры)



У объекта есть  
**методы**  
(методы – это действия,  
то есть что может  
делать объект)

Например, машина – это объект.



У любой машины есть такие  
свойства: модель, цвет, размер и т.д.

Методы машины:  
затормозить ();  
нажатьНаГаз ();  
открытьДверь ();  
закрытьДверь ();  
и т.д.

Например, котёнок – это объект.



У любого котенка есть свойства:  
порода, имя, цвет, длина шерсти,  
возраст и т.д.

Методы котенка:  
спать();  
кушать();  
играть ();  
шкодить ();  
и т.д.

## Что такое ООП

Можно сказать, что Ваш друг - объект класса "Человек", маленькая дворняжка или большой ротвейлер - объекты класса "Собака", с общими методами и свойствами, но разными индивидуальными параметрами.

Кроме базовых понятий "класса" и "объекта" рассмотрим основные принципы ООП - **наследование, инкапсуляция, полиморфизм** (запомните эти три слова - они наверняка пригодятся на интервью!).

Что такое ООП. Наследование, инкапсуляция, полиморфизм.

## Принцип 1. Наследование

Благодаря классам и объектам мы можем не прописывать код каждый раз заново, а просто создавать объекты класса.

Наследование позволяет экономить время при создании нового класса.

Представьте, что у Вас есть класс "Кошка". У нее есть свойства - имя, цвет и порода. У этого класса есть методы - спать, кушать, играть, мурлыкать. А теперь, представим, нам нужно создать новый класс - "Котенок". У него тоже есть имя, цвет и порода. Он тоже умеет спать, кушать, играть, мурлыкать. Но, кроме того, он еще может, например, "искать маму". Благодаря наследованию, можно не прописывать все методы и свойства класса заново. Можно просто указать, что класс "Котенок" наследует класс "Кошка", и добавить недостающие методы и параметры. Удобно, правда?

Что такое ООП. Наследование, инкапсуляция, полиморфизм.

## Принцип 2. Инкапсуляция

"Инкапсуляция" можно расшифровать как "заключение в капсулу". Если проще - это "обеспечение безопасности" в Java.

Дело в том, что в каждой программе есть параметры, к которым нельзя давать доступ всем. Например, разве будет интересно играть в игру, если пользователь сможет свободно изменять свои очки сам? Инкапсуляция позволяет регулировать уровни доступа в программе.

Что такое ООП. Наследование, инкапсуляция, полиморфизм.

## Принцип 3. Полиморфизм

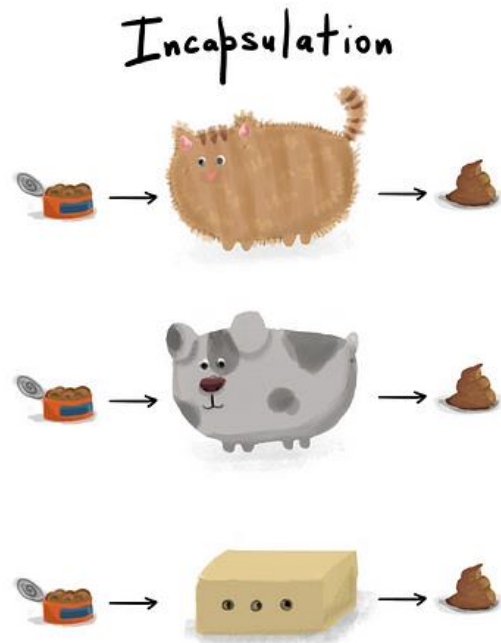
Смысл полиморфизма в том, что Вы можете давать одно и то же название для методов, которые имеют одинаковый смысл, но принимают разные типы данных.

Хотя это может Вам показаться странным, но в других языках - где полиморфизма нет - нужно иногда запоминать названия 5 методов вместо одного - только из-за того, что они работают с разными типами данных.

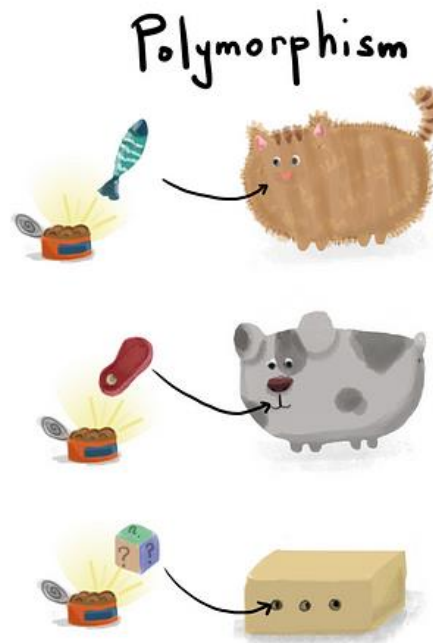
# Java



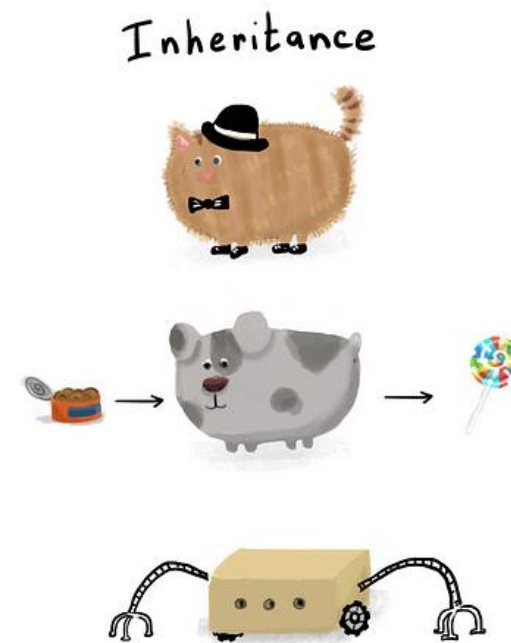
Что такое ООП. Наследование, инкапсуляция, полиморфизм.



every animal eats  
and then poop



each animal can eat  
its own type of food



you can create new type of animal  
changing or adding properties



# Java



## Что такое "класс"

Класс - это "шаблон для объекта", который создаётся в Java. Понятие класса, как и понятие объекта, являются основой ООП.

Мы уже создавали классы, например:

```
class Test {  
    public static void main(String args[]){  
        int k;  
        k = 10;  
        System.out.println (k);  
    }  
}
```

# Java



## Что такое "класс"

Создание **класса**, в общей форме, будет выглядеть так:

- Сначала указывается тип класса, например **public**
- Потом, мы присваиваем имя конкретному классу - по этому имени мы будем к нему обращаться далее в программе. Например, Dog
- Затем, можно добавить параметры.

```
public class Dog {  
    String breed;  
    int age;  
    String color;
```

```
    void barking() {  
    }
```

```
    void hungry() {  
    }
```

```
    void sleeping() {  
    }  
}
```

# Java



## Что такое "класс"

Создание **объекта** класса, в общей форме, будет выглядеть так:

- Сначала указывается название класса, например "Cat"
- Потом, мы присваиваем имя конкретному экземпляру класса - по этому имени мы будем к нему обращаться далее в программе. Например, "Васька".
- Дальше ставится равно, пишется "new" и снова повторяется название Вашего класса.

```
MyClass name = new MyClass ( );
```

↑  
название  
класса

↑  
**ИМЯ** объекта  
класса

↑  
название  
класса  
(да, еще раз)

```
MyClass name = new MyClass ( );
```

↑  
ключевое  
слово "new"

↑  
тут могут  
задаваться доп.  
параметры

# Java



## Конструкторы в Java

Конструкторы - это специальные методы, которые вызывается при создании объекта. Они "конструируют" новый объект определенного класса.

Представим, как работает программа:

- Создаём основное "тело" программы, прописывая метод main

```
public class Test {  
    public static void main(String[] args){  
  
    }  
}
```

## Конструкторы в Java

- Создаём класс Cat (можно отдельным файлом):

```
class Cat{  
    private String name;  
    private String color;  
  
    public String getName(){  
        return name;  
    }  
  
    public void setName(String a){  
        name = a;  
    }  
  
    public String getColor(){  
        return color;  
    }  
  
    public void setColor(String color){  
        this.color = color;  
    }  
}
```

# Java



## Конструкторы в Java

- Пишем строку, которая должна создать **объект** класса Cat:

```
public class Test {  
    public static void main(String[] args){  
        Cat cat1= new Cat();  
    }  
}
```


- Когда программа приступает к созданию объекта cat1, она идет в class Cat:

### Класс Test

```
1 public class Test {  
2     public static void main(String[] args){  
3         Cat cat1= new Cat();  
4     }  
5 }
```

### Класс Cat

```
1 class Cat{  
2  
3     private String name;  
4     private String color;  
5  
6     public String getName(){  
7         return name;  
8     }  
9  
10    public void setName(String a){  
11        name = a;  
12    }  
13  
14    public String getColor(){  
15        return color;  
16    }  
17  
18    public void setColor(String color){  
19        this.color = color;  
20    }  
21  
22 }
```



и что теперь мне делать?

# Java



## Конструкторы в Java


- Тут и появляется необходимость в конструкторах. В первую очередь Java ищет именно конструкторы, которые укажут, как именно создавать объект.:

### Класс Test

```
1 public class Test {  
2     public static void main(String[] args) {  
3         Cat cat1= new Cat();  
4     }  
5 }
```

### Класс Cat

```
1 class Cat {  
2  
3     private String name;  
4     private String color;  
5  
6     public String getName() {  
7         return name;  
8     }  
9  
10    public void setName(String a) {  
11        name = a;  
12    }  
13  
14    public String getColor() {  
15        return color;  
16    }  
17  
18    public void setColor(String color) {  
19        this.color = color;  
20    }  
21  
22 }
```



Где мой конструктор?

## Конструкторы в Java

- Пишем конструктор (метод) для класса Cat:

```
public Cat(String x, String y){  
    this.name = x;  
    this.color = y;  
}
```

Или

```
public Cat() {}
```

```
1 public class Test {  
2     public static void main(String[] args){  
3         Cat cat1= new Cat();  
4  
5         cat1.setName("Murka");  
6         cat1.setColor("grey");  
7     }  
8 }
```

3 и 1

```
1 public class Test {  
2     public static void main(String[] args){  
3         Cat cat1= new Cat("Murka", "grey");  
4     }  
5 }
```



## Задание Класс Phone.

- Создайте класс Phone, который содержит переменные number, model и weight.
- Создайте три объекта этого класса.
- Выведите на консоль значения их переменных.
- Добавить в класс Phone методы: receiveCall(), имеет один параметр - имя звонящего. Выводит на консоль сообщение "Звонит {name}". Метод getNumber() - возвращает номер телефона. Вызвать эти методы для каждого из объектов.
- Добавить конструктор в класс Phone, который принимает на вход три параметра для инициализации переменных класса - number, model и weight.

## Задание Класс Phone.

- Добавить конструктор, который принимает на вход два параметра для инициализации переменных класса - `number`, `model`.
- Добавить конструктор без параметров.
- Добавьте метод `receivCall()`, который принимает два параметра - имя звонящего и номер телефона звонящего. Вызвать этот метод.

## Задание LibraryApp.

- Создать класс LibraryApp в котором будут 2 класса: Reader и Book (отдельными файлами).
- Класс Book, хранит следующую информацию о каждой книге:
  - Название книги
  - Автор книги
  - Методы getTitle() (возвращает название книги), toString() (возвращает название книги и автора одной строкой).
- Класс Reader, хранит следующую информацию о пользователе библиотеки:
  - ФИО
  - номер читательского билета
  - факультет
  - дата рождения
  - телефон
  - Методы takeBook(), returnBook()

## Задание LibraryApp.

- Создать методы `takeBook()`, `returnBook()`:
  - `takeBook()`, который будет принимать количество взятых книг. Выводит на консоль сообщение "Петров В. В. взял 3 книги".
  - `takeBook()`, который будет принимать переменное количество названий книг. Выводит на консоль сообщение "Петров В. В. взял книги: Приключения, Словарь, Энциклопедия".
  - `takeBook()`, который будет принимать переменное количество объектов класса `Book`. Выводит на консоль сообщение "Петров В. В. взял книги: Приключения, Словарь, Энциклопедия".
  - Аналогичным образом создать метод `returnBook()`. Выводит на консоль сообщение "Петров В. В. вернул книги: Приключения, Словарь, Энциклопедия". Или "Петров В. В. вернул 3 книги".



Thanks for your time 😊