ООП

...

И три слона, на которых это все держится

ООП - представление программы в виде совокупности объектов, являющихся экземплярами классов, образующих иерархию.

Класс

```
class Dog {
        int age;
        String name;


        public void voice() {
                System.out.println("Woof!");
        }
}
```

---

int age, String name - переменные класса

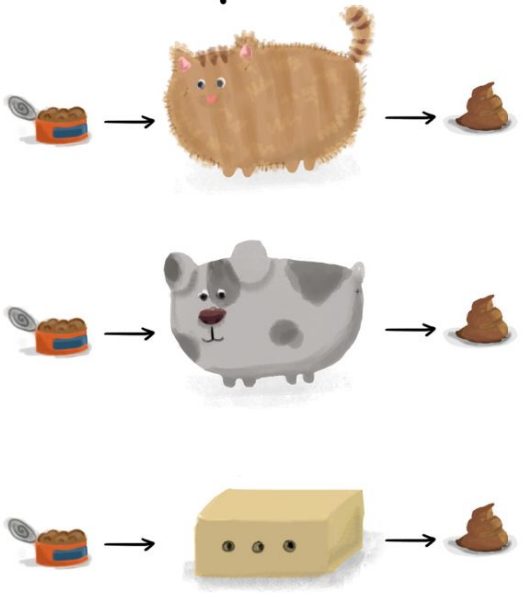
Объект

```
public static void main() {
        Dog jack = new Dog();
        jack.voice();
        //Woof!
}
```

---

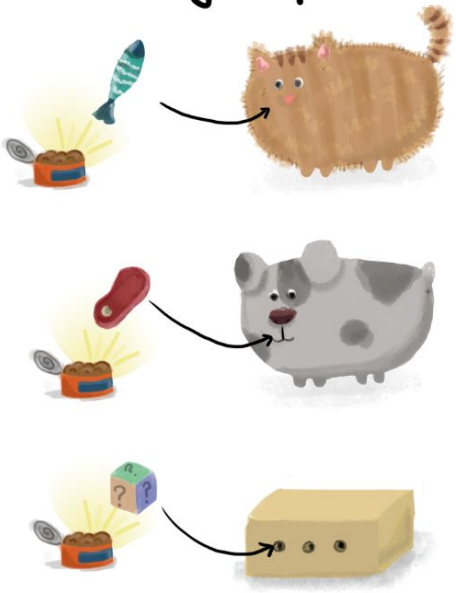
jack.age, jack.name - переменные экземпляра

Incapsulation



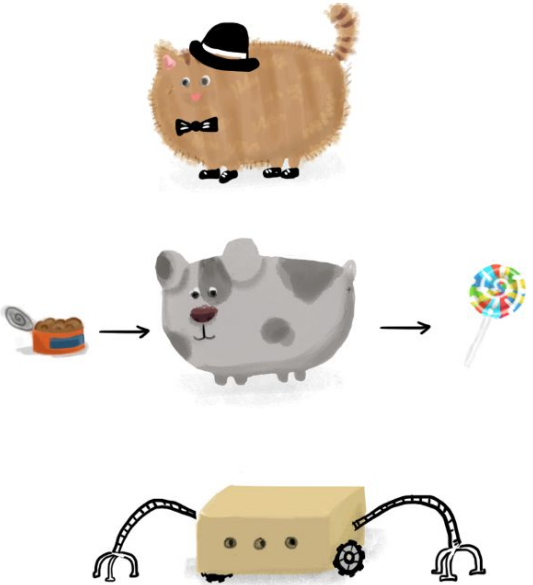
every animal eats
and then poop

Polymorphism



each animal can eat
its own type of food

Inheritance

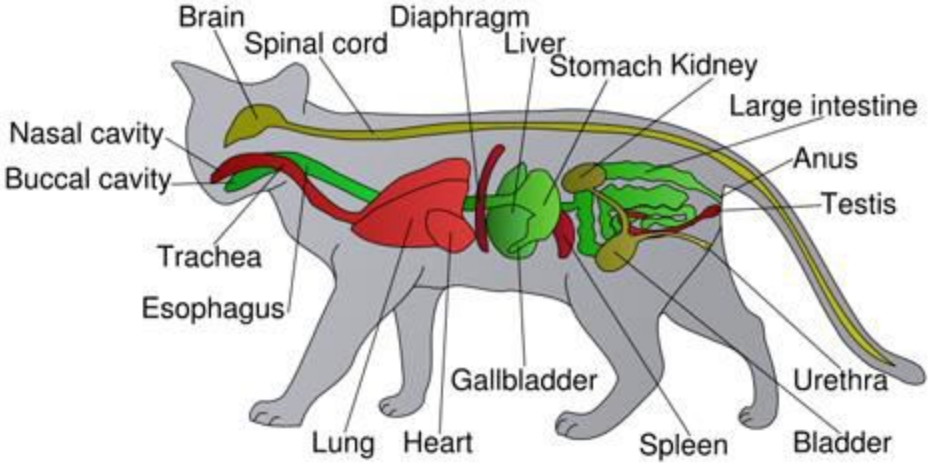


you can create new type of animal
changing or adding properties

Инкапсуляция

понятно, что делать с объектом
(кормить, что же еще)

не понятно, как именно взаимодействовать с объектом
(слишком много деталей)



Модификаторы доступа

Private, Protected, Public

Private означает, что к этому члену класса нельзя будет обратиться из методов других классов.

Public означает, что данный член класса является доступным. Если это поле, его можно использовать в выражениях или изменять при помощи присваивания, а если метод, его можно вызывать.

Protected означает, что доступ к полю или методу имеет сам класс и все его потомки.

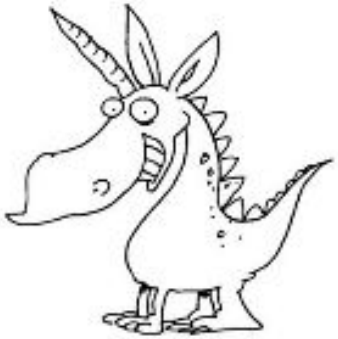
Интерфейс

```
interface Zoo {

        public void voice();

}
```

Полиморфизм

Животное умеет:

- ходить
- подавать голос



Мяу

Муу

Гав

[Интерфейс животное]

[Реализация интерфейса животное]

Полиморфизм - это один интерфейс для множества реализаций

override

```
public void some(String name) {
    System.out.println('name:' + name);
}

public double some(double age)
    return age/12;
}
```

Наследование

Абстрактные классы

```
public abstract class Animal {
    protected String name;
    protected int age;

    public void setInfo(name, age) {
            this.name = name;
            this.age = age;
    }

    public String getInfo() {
            System.out.println('name: ' +
                    this.name + ', age: ' + age);
    }
    public abstract void voice();
}
```

Extends

```
class Dog extends Animals {
        public void voice() {
                System.out.println("Woof!");
        }
}

//...//

Dog jack = new Dog();
jack.setInfo('Jack', 5);
jack.getInfo();

//name: Jack, age: 5
```

Модификаторы доступа (2)

Final, Static

Final означает запрет на переопределение {method | class | class variable} в подклассах.

Static class variable означает, что значение class variable - одно для всех экземпляров класса, и изменение его в одном экземпляре означает изменение его во всех экземплярах.

Static method (он же - метод класса) применяется к статической переменной класса. Метод может применяться, даже если не создано еще ни одного экземпляра класса.

Static блок кода - применяется для сложной инициализации static class variable.

final

class, method, class variable

http://kit.znu.edu.ua/iLec/6sem/PMES/Labs/1/class-doc.pdf, стр.10

```
public final class DogVoice {
private final int count = 2; //a constant
public final void voice() {
try {

 AudioInputStream audio =
        AudioSystem.getAudioInputStream(new
        File("D:/Voices/woof.wav").getAbsoluteFile());

        final Clip clip = AudioSystem.getClip();
        clip.open(audio);
//easy way
       for (int i=0;i<count;i++) clip.start();

//heavy way (Java 1.8!!! lambda-calculus)
       IntStream.range(1,count).forEach(i->{clip.start();});
       }
catch(Exception ex) {
System.out.println("Error with playing sound.");
ex.printStackTrace(); }
}

}
```

static

method, class variable, block of code

http://kit.znu.edu.ua/iLec/6sem/PMES/Labs/1/class-doc.pdf, стр.12-14

```
class ZooAnimal extends Animals {
public void voice() {//voice function    }
private static int uid; // last generated id in Zoo

static { uid=0; }
private int id; // animal id in Zoo
public String name;
public int age;
public static int getNextId() {
        return ++uid; }
//ZooAnimal constructor
ZooAnimal(int age, String name) {
       id=getNextId();
       System.out.println("Animal id is:" + id);
       this.age = age;
       this.name=name; }
}
```

static

можно/нельзя

1. В статическом методе нельзя использовать this, super
2. В статическом методе нельзя напрямую, без создания экземпляра класса, ссылаться на нестатические поля и методы
3. Статические методы не могут быть абстрактными
4. Статические методы переопределяются в подклассах только как статические