ES6 OOP

YAROSHEVICH
Andrey Olegovich

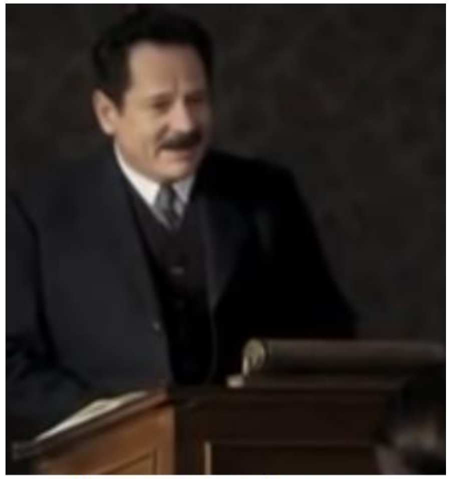
www.YAROSHEVICH.com
ao@asp.net.by
+375 29 254-07-92

Ed Sheeran - Thinking Out Loud

https://www.youtube.com/watch?v=lp-EO5I60KA

Thinks out loud

This is how Heidegger read his lectures – here is a fragment



Martin Heidegger
86: 1889 — 1976

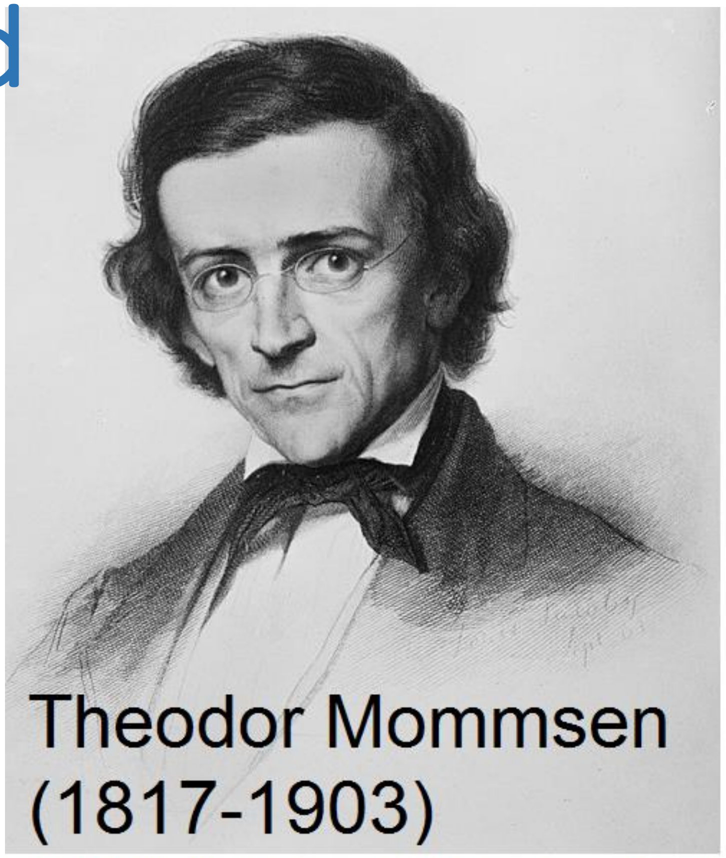
https://youtu.be/VZtXKJU7E9s?t=3516

And Prof. Mc. Key from Cambridge too

Prof. Mc Key
1967 – 2016

https://www.youtube.com/watch?v=BCiZc0n6COY&list=PLruBu5BI5n4aFpG32iMbdWoRVAA-Vcso6

Old teaching method



1902 Nobel Prize in Literature for "A History of Rome"

If someone's head doesn't understand syllogisms, it understands a stick

It must be said that it was effective, children showed good results, but its disadvantages outweighed its benefits, because corporal punishment humiliates human dignity, so by the middle of the 20th century it was completely abandoned.

OOP in Ecma6 (JavaScript)

Javascript › 1995: “Mocha” project at Netscape

JavaScript was first created by Brendan Eich at Netscape in 1995; it was nicknamed Mocha during development, and ultimately named JavaScript to piggyback on the popularity of Java (another programming language).

On March 24, 2014, Mozilla made the decision to appoint Eich as CEO of Mozilla Corporation. After 11 days as CEO, Eich resigned on April 3, 2014, and left Mozilla over his opposition to same-sex marriage.

Brendan Eich on Javascript

JS had to “look like Java” only less so, be Java’s dumb kid brother or boy-hostage sidekick.

Plus, I had to be done in ten days or something worse than JS would have happened.

—Brendan Eich on Javascript

>Javascript takes off, included in Microsoft’s IE

At first it was a simplified version of Java

Jeremy Keith

Keith in 2019

1996: submitted to Ecma as standard

today

› Java alive and well server-side

› but JS dominates client-side

› making inroads server-side too

(eg, node.js)

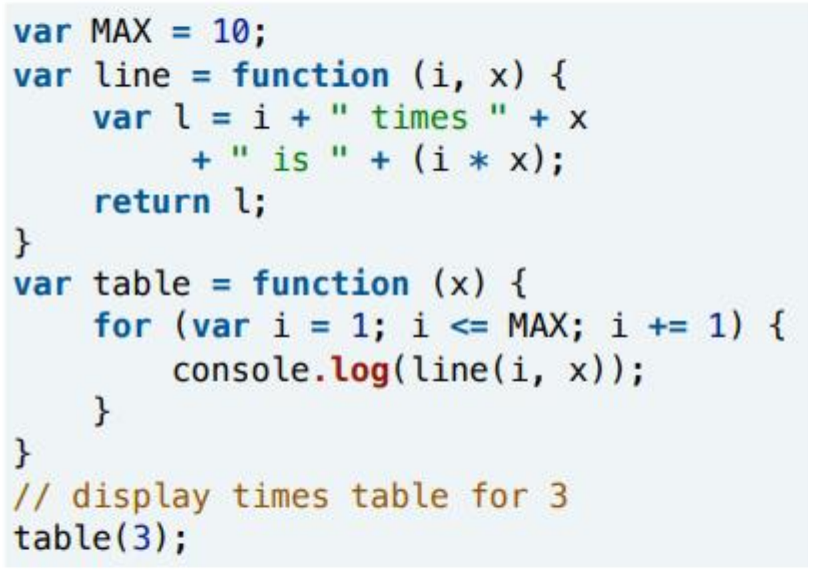
syntax

statements like Java

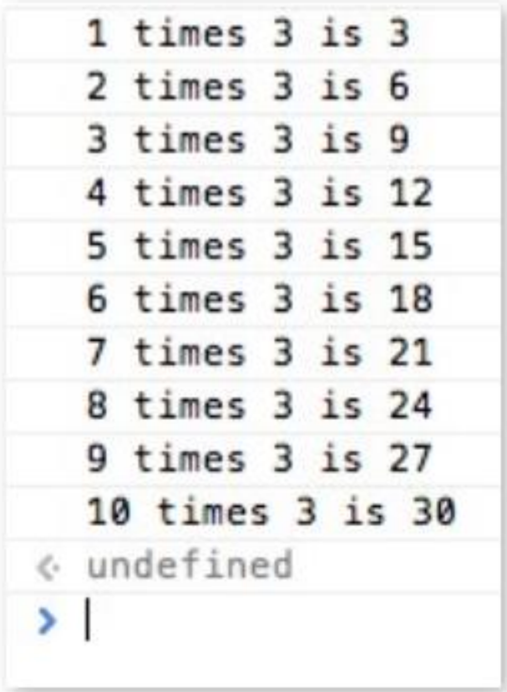
› while, for, if, switch, try/catch, return, break, throw

Comments

› use //, avoid /**/



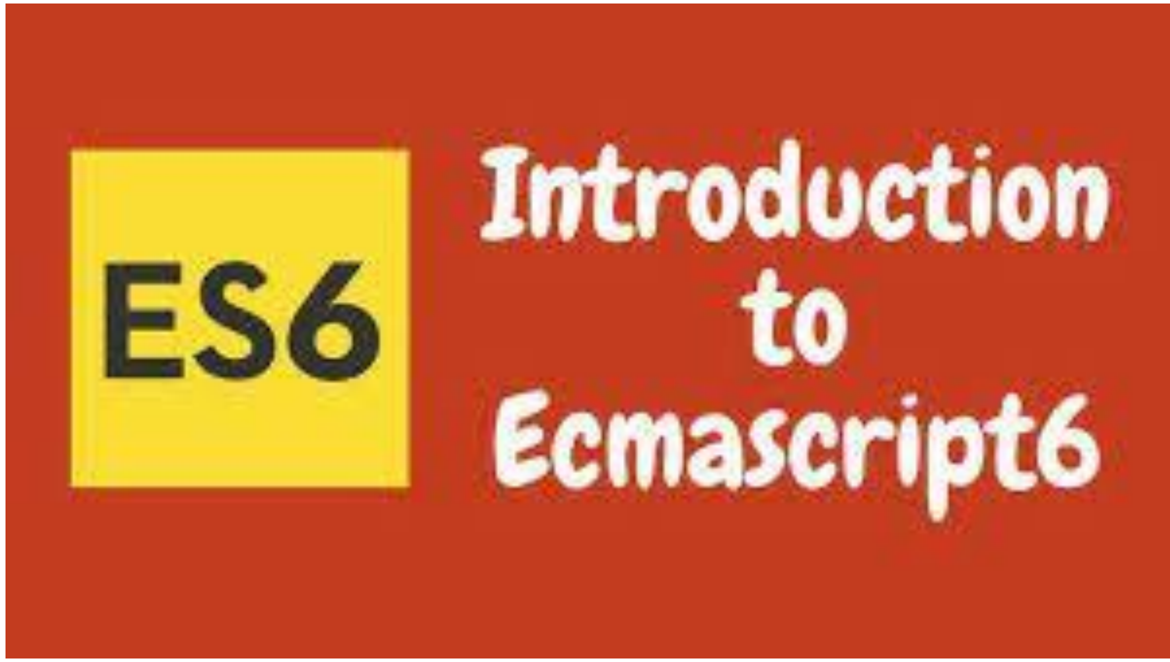
```
var MAX = 10;
var line = function (i, x) {
    var l = i + " times " + x
        + " is " + (i * x);
    return l;
}
var table = function (x) {
    for (var i = 1; i <= MAX; i += 1) {
        console.log(line(i, x));
    }
}
// display times table for 3
table(3);
```



```
  1 times 3 is 3
  2 times 3 is 6
  3 times 3 is 9
  4 times 3 is 12
  5 times 3 is 15
  6 times 3 is 18
  7 times 3 is 21
  8 times 3 is 24
  9 times 3 is 27
  10 times 3 is 30
<· undefined
> |
```

JavaScript

Smalyuk Antonov Fedorovich



ES6

Introduction to Ecmascript6

https://www.youtube.com/watch?v=tt1SkTpHUUU

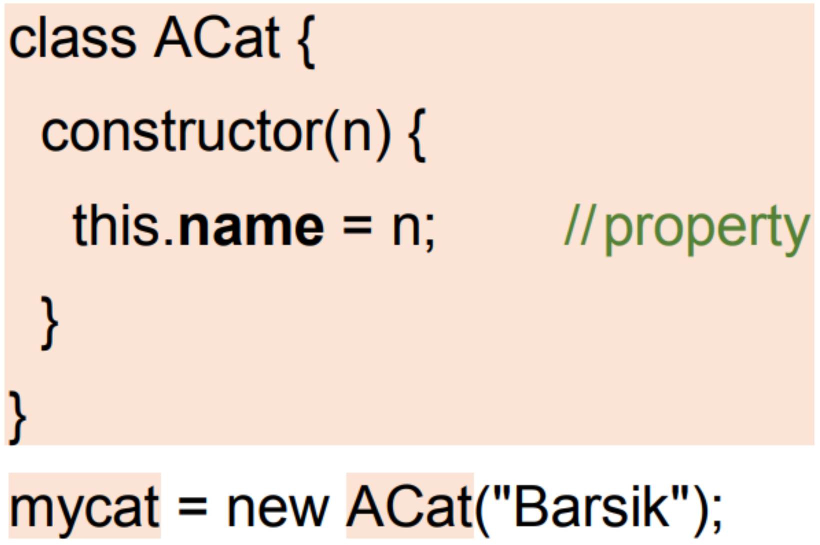
Classes

As for classes, in Ecma 6 the syntax has become very similar to Java syntax (C++ and C#), which is good

Java syntax

```
class ACat {
string name;
public ACat(n){
  this.name=n;
 }
}
ACat mycat=
new ACat(“Barsik");
```

JavaScript syntax



```
class ACat {
  constructor(n) {
    this.name = n;       //property
  }
}
mycat = new ACat("Barsik");
```

Functions

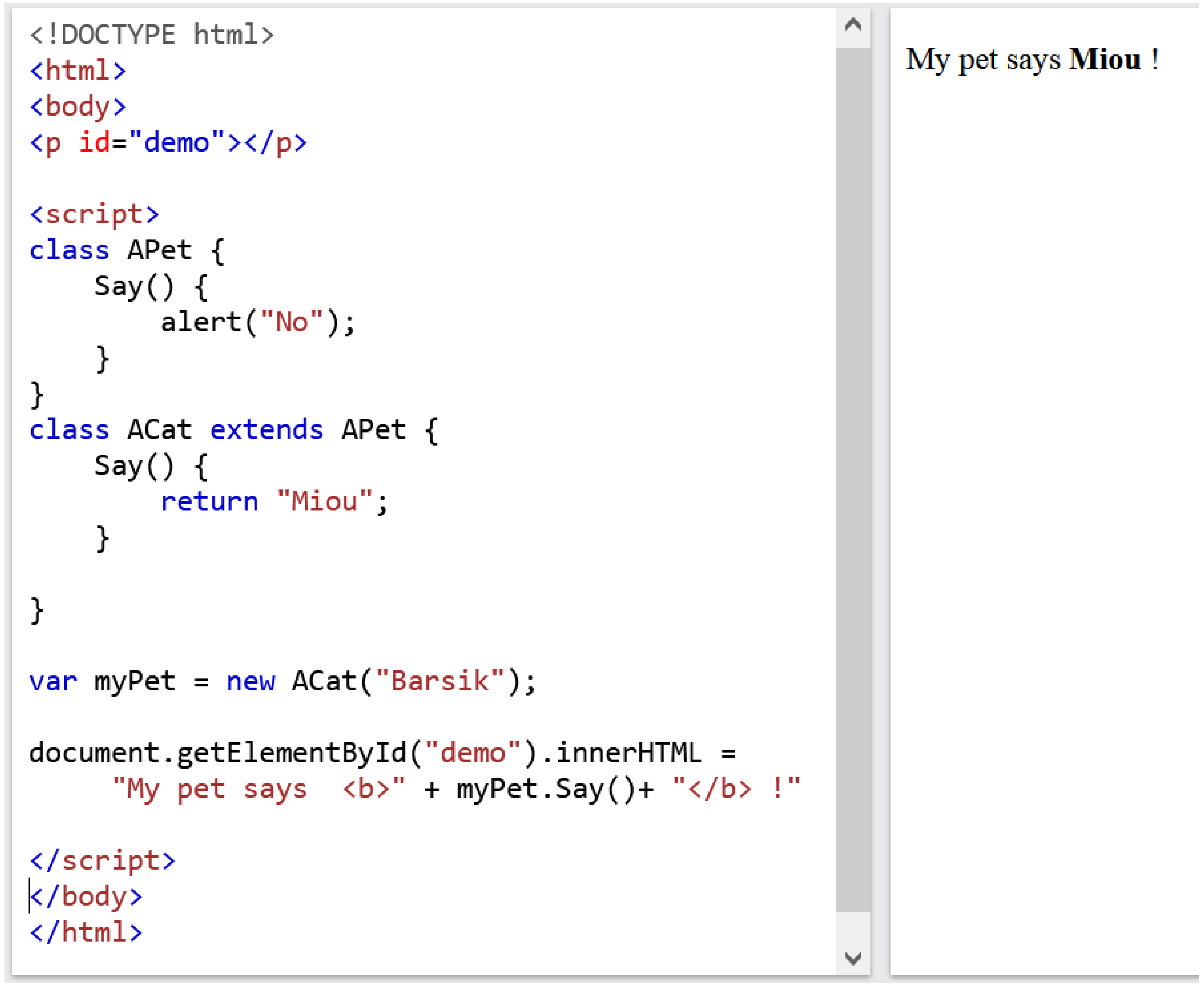
```
class Cat {
  constructor(n) {
    this.name = n;
  }
  Say() {
     return "meou";
  }
}

var myCat = new Cat("Barsik");

document.getElementById("demo").innerHTML ="My cat says  <b>" + myCat.Say()+ "</b> !"
```

Class Inheritance

To create a class inheritance, use the extends keyword.

A class created with a class inheritance inherits all the methods from another class:

Example
Create a class named “ACat" which will inherit the methods from the “APet" class:



```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>

<script>
class APet {
    Say() {
        alert("No");
    }
}
class ACat extends APet {
    Say() {
        return "Miou";
    }

}

var myPet = new ACat("Barsik");

document.getElementById("demo").innerHTML =
     "My pet says  <b>" + myPet.Say()+ "</b> !"

</script>
</body>
</html>
```

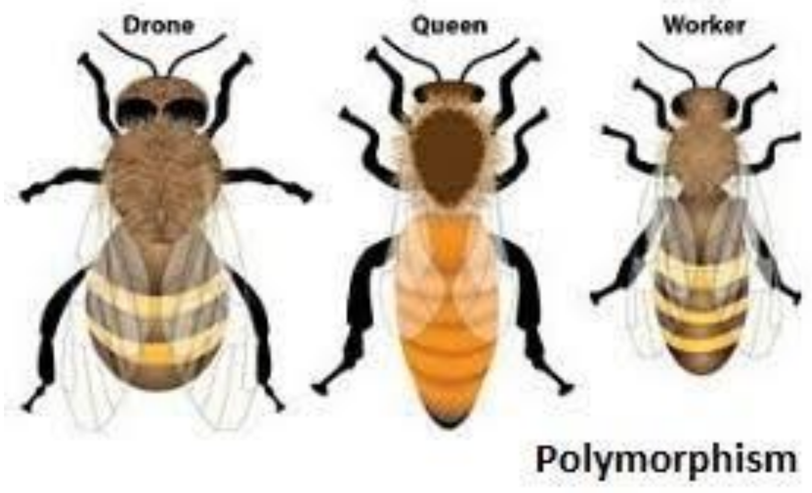
My pet says **Miou** !

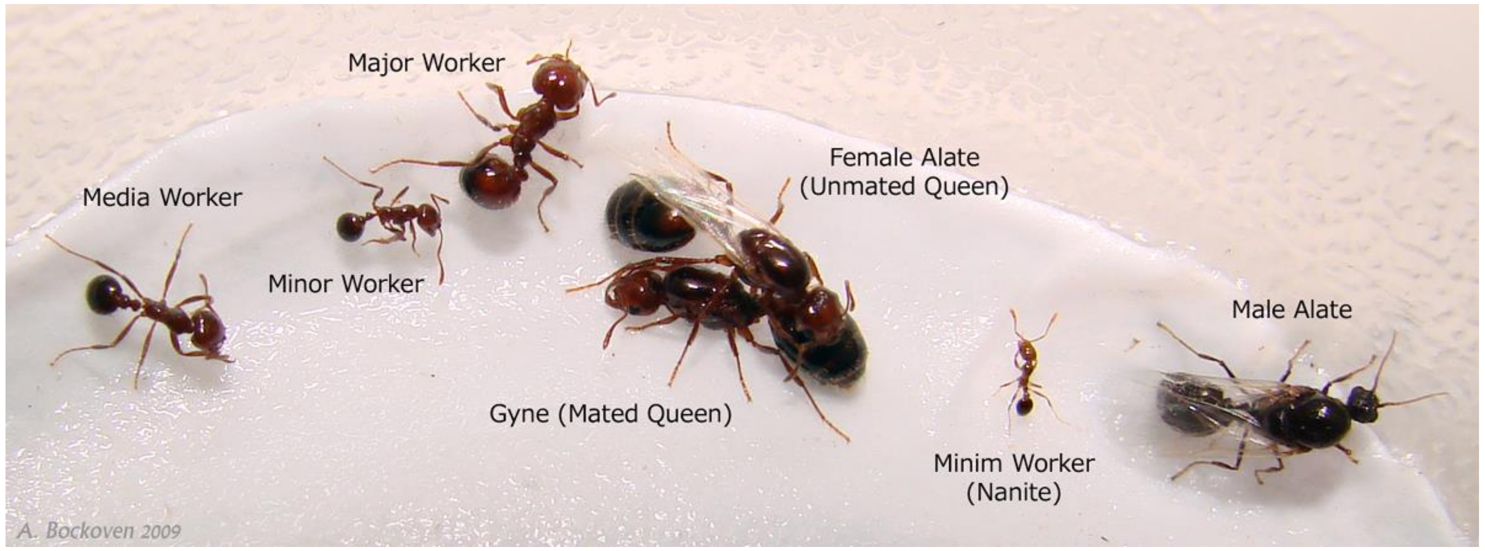
Polymorphism

What is Polymorphism?

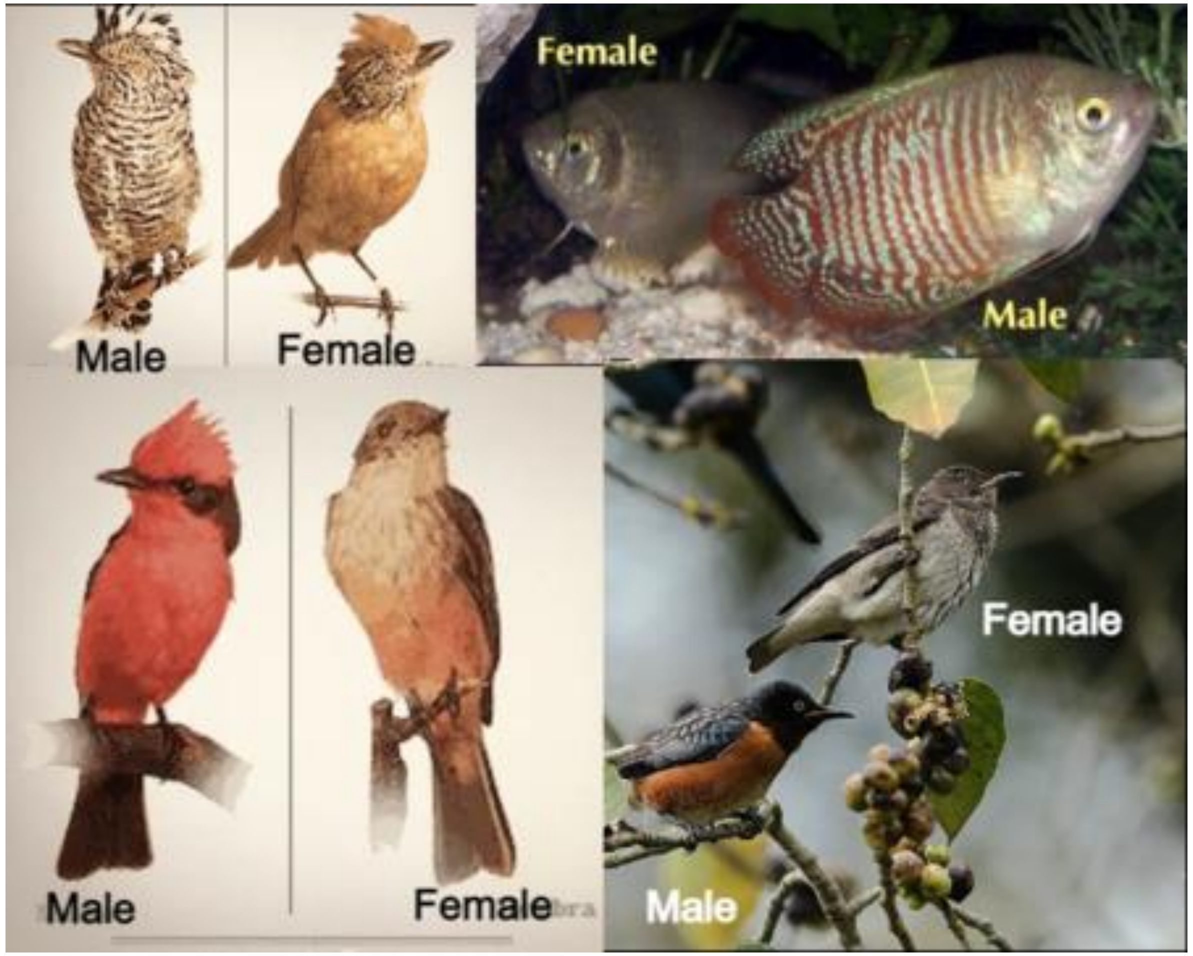
Polymorphism is one of the core concepts of object-oriented programming languages
where **poly** means **many** and **morphism** means **transforming one form into another**.

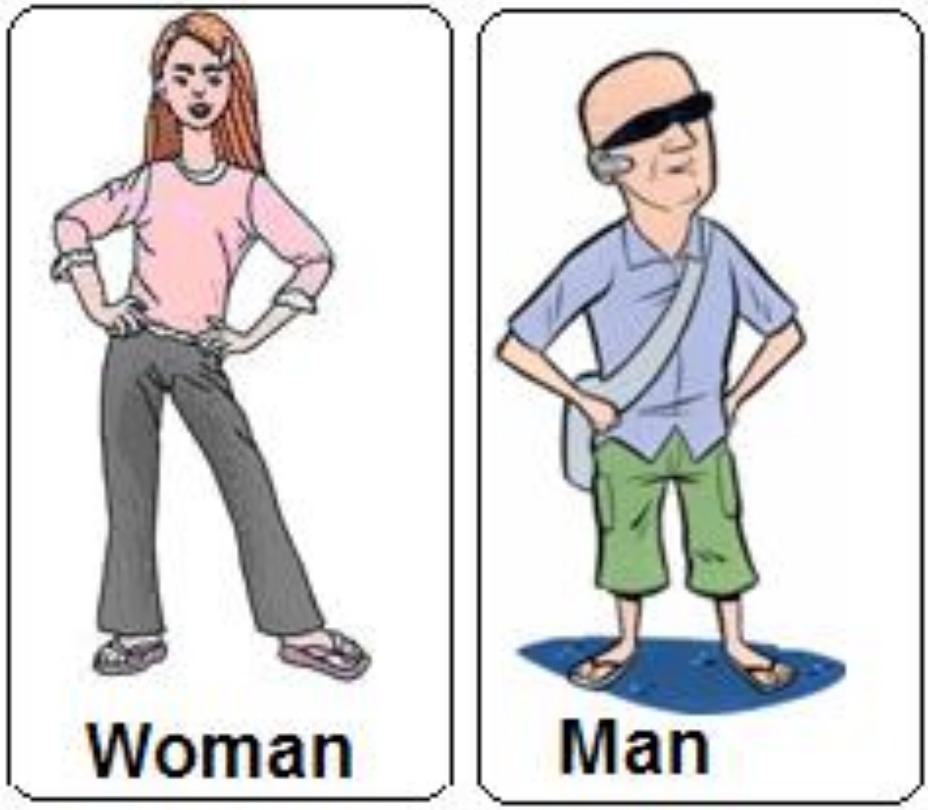
Polymorphism in Biology





Bimorphism





Polymorphism means the same function with different signatures is called many times.

function **WashDishes()**

Men's version
wipe dry

Female version
wet with water

function SaturdayRest()

male version

and

female version

Features of Polymorphism:

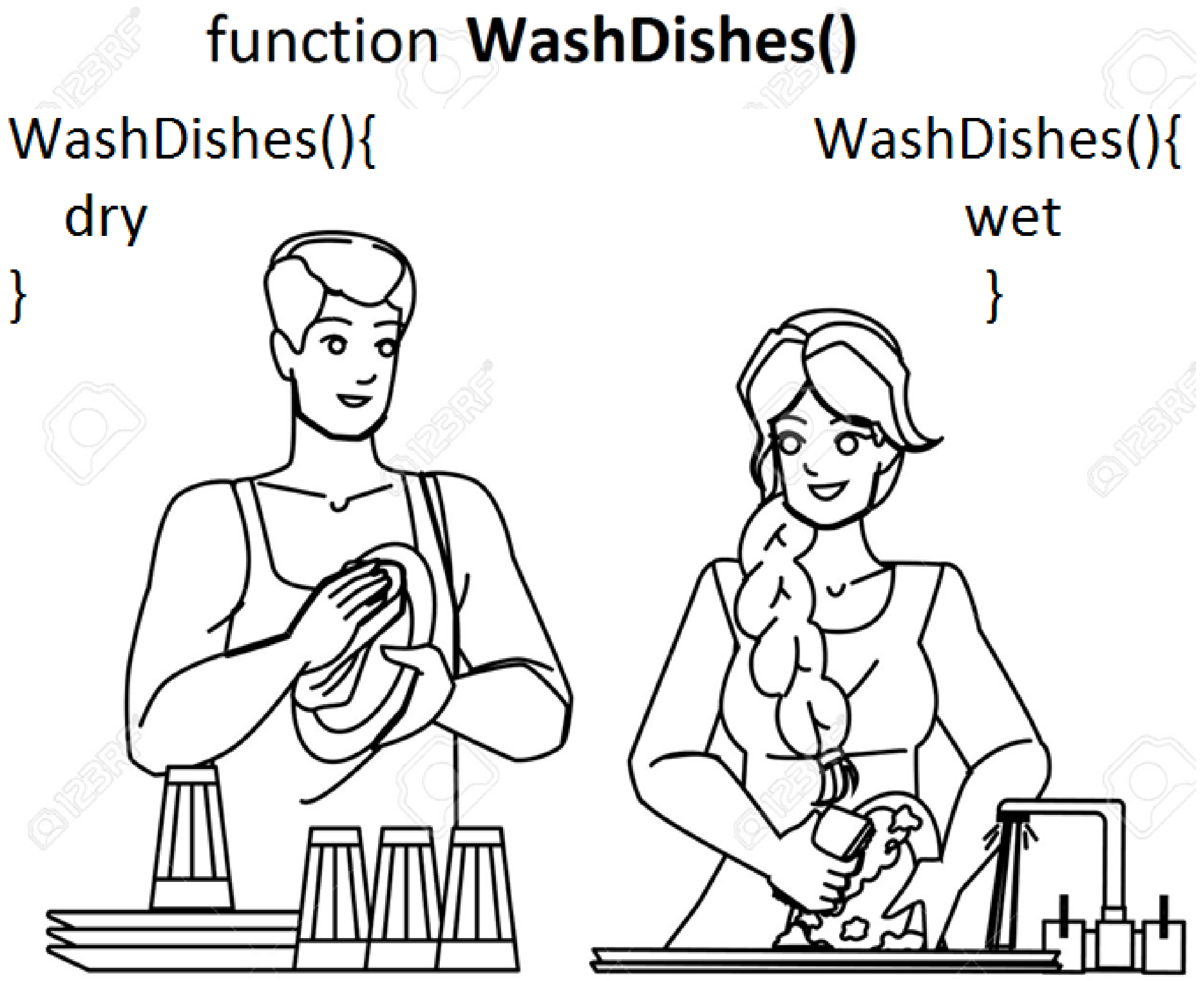
Programmers can use the same method name repeatedly.

Polymorphism has the effect of reducing the number of functionalities that can be paired together.

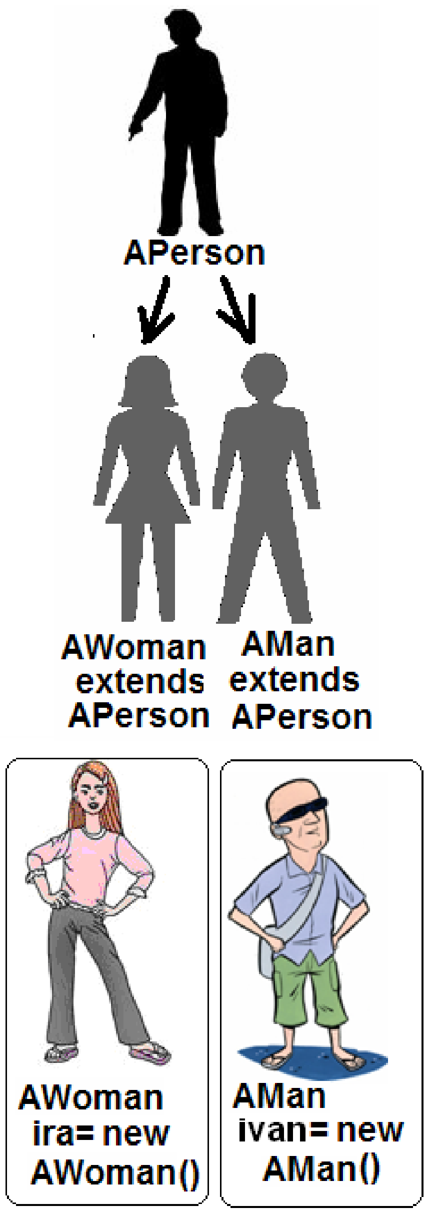
The super task of polymorphism is to establish joint actions

We create a class APerson, which inherits classes defining a man and a woman AMan and AWomen respectively.

*I have a habit of adding the letter 'A' - **APerson** – to classes I create myself to remember that it is my class.*



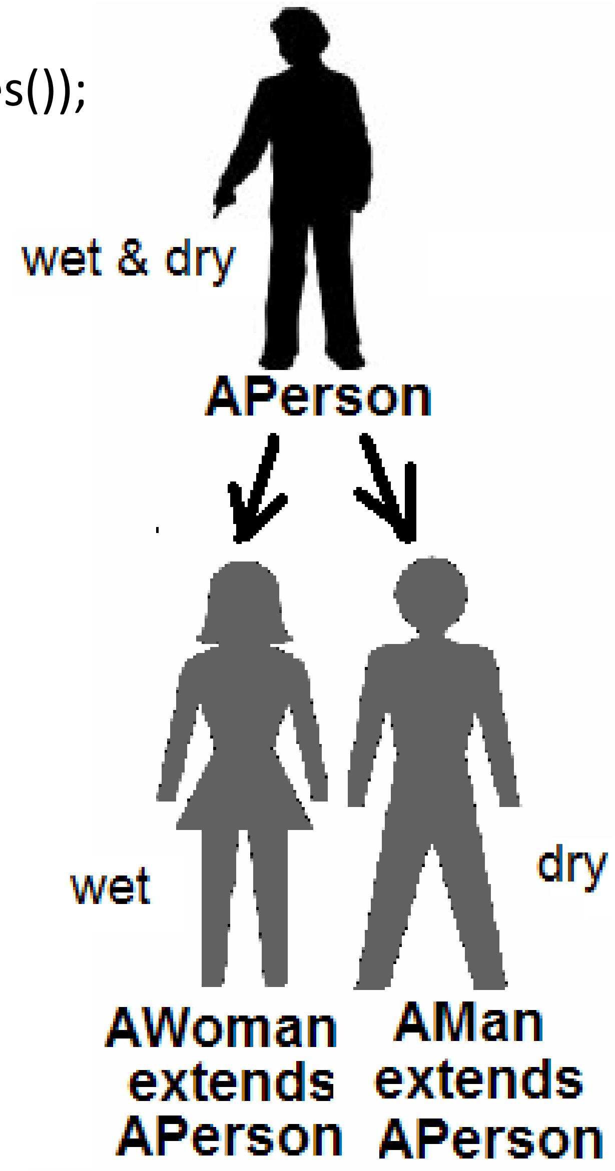
```
function WashDishes()

WashDishes(){
  dry
}

WashDishes(){
  wet
}
```



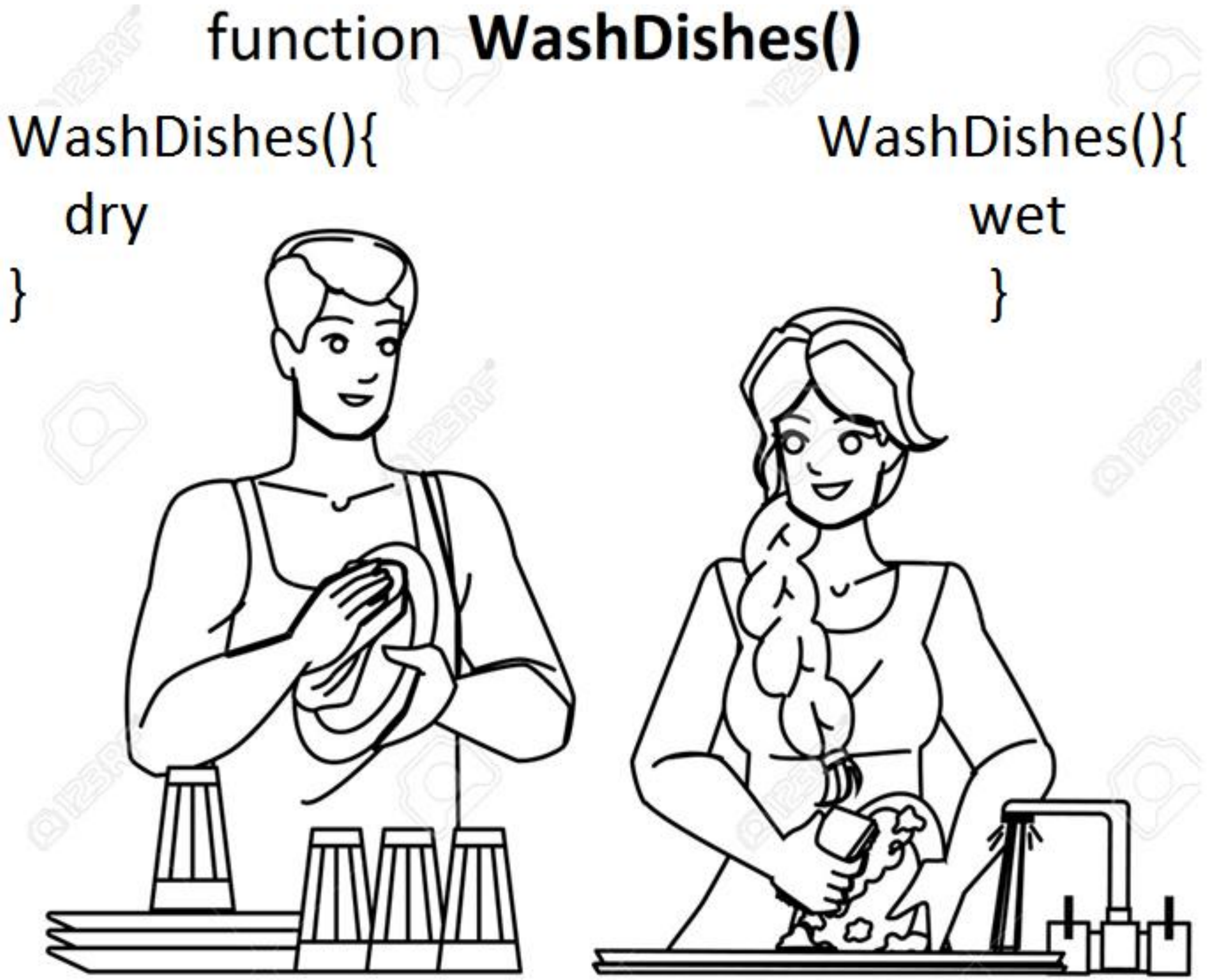
```
APerson

AWoman extends APerson
AMan extends APerson

AWoman ira= new AWoman()
AMan ivan= new AMan()
```

```
<script>
class APerson {
  WashDishes() {
    return 'dry and wet';
  }
}

class AWomen extends APerson
{
  WashDishes() {
    return 'wet';
  }
}

class AMan extends APerson {
  WashDishes() {
    return 'dry';
  }
}
```

```
 single= new APerson();
document.write(" "+ single.WashDishes());
//wet & dry


women = new AWomen();
man = new AMan();

var family = [women, man];

for(i=0;i<2;i++)
{
  document.write(" "+
       family[i].WashDishes());
}

//only wet or dry
</script>
```



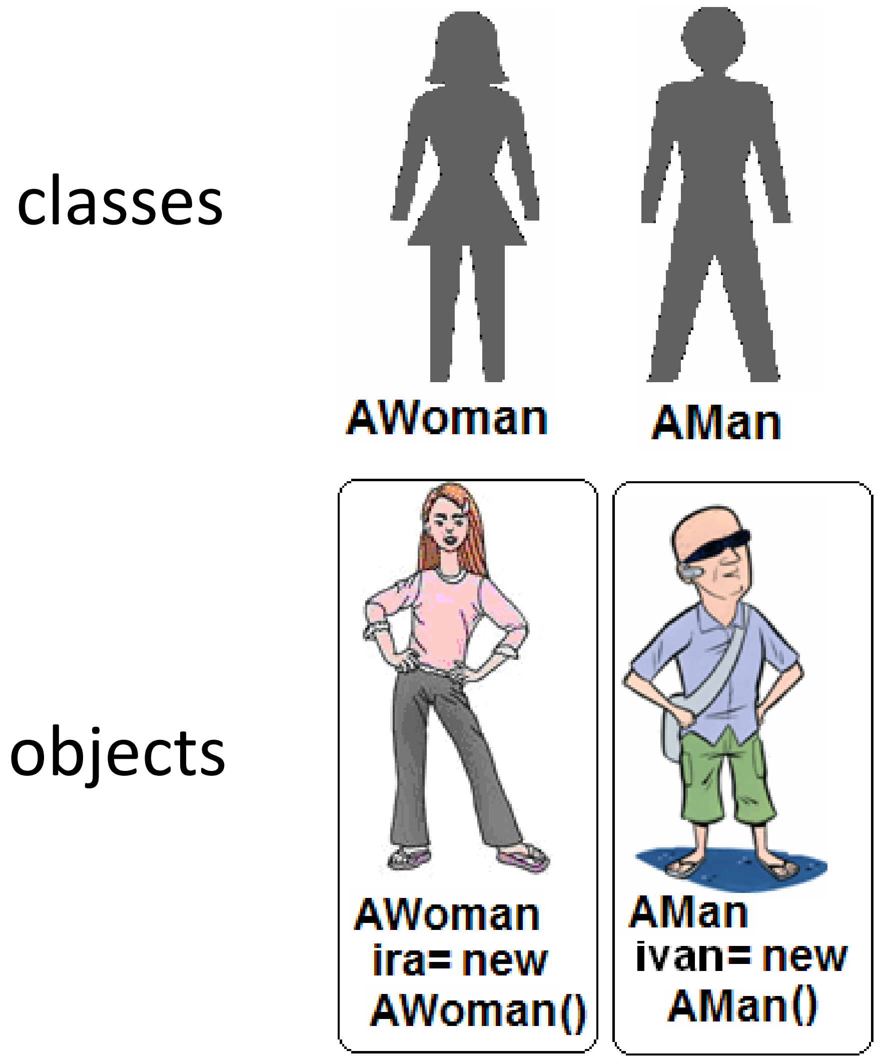
```
<script>
class APerson {
  WashDishes() {
    return 'dry and wet';
  }
}
class AWomen {
  WashDishes() {
    return 'wet';
  }
}

class AMan  {
  WashDishes() {
    return 'dry';
  }
}
women = new AWomen();
man = new AMan();
var family = [women,man];
for(i=0;i<2;i++)
{
  document.write(i+". "+

family[i].WashDishes()+"<br>");
}
</script>
```



Classes and objects

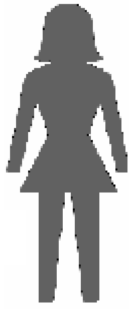
1. Class is not a real-world entity.
It is just a template
or blueprint
or prototype
from which objects are created.
2. Class does not occupy memory.

1. Object is a real-world entity.
2. The object takes up memory.

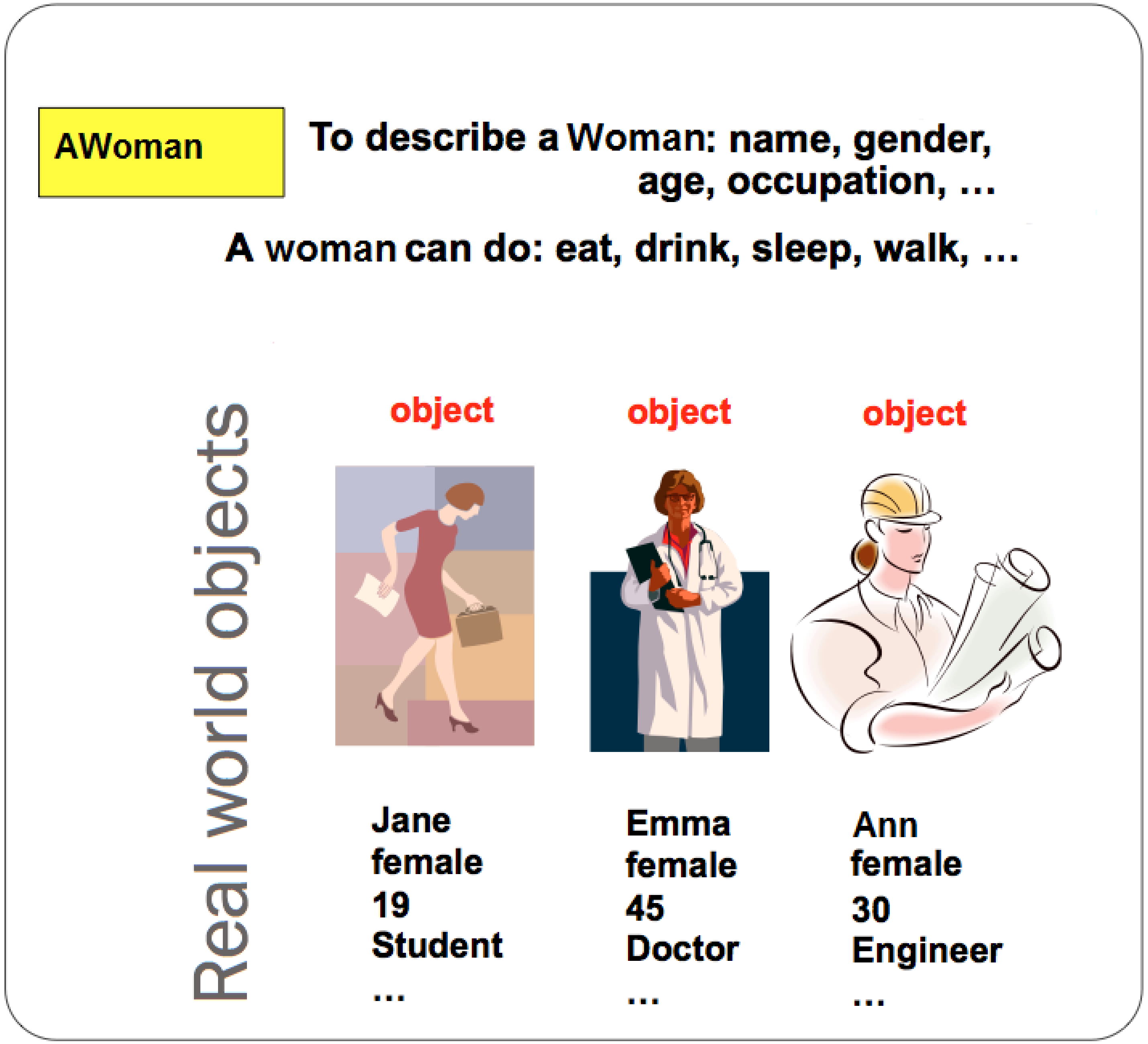


Class and Objects

```
class AWomen {
 constructor(name, gender, age, o)
 {
  this.name=name;
  this.gender=gender;
  this.age=age;
  this.occupation=o;
 }
}
```



```
jane =
new AWomen("Jane","female",19,"Student");

document.write("Name: "+ jane.name +"<br>"+
        "Gender: "+ jane.gender+"<br>"+
        "Age: "+ jane.age+"<br>"+
        "Gender: "+ jane.occupation+"<br>");
```



Die Bremer Stadt Musikanten

Example.

You need to write a program that simulates the actions of the Bremen Town Musicians.

The animals must scare the robbers - in the excerpt https://www.youtube.com/watch?v=8xXiLfSjT9w&t=569s - they use a clever trick - for example, a cat shows its face, and the voice is provided by a dog – “Woof”.

It seems the cat is talking "woof-woof".

The incomprehensible causes horror

The robbers cannot explain what they saw - because they didn't graduate from universities, and think in standards (poorly).

The mystical explanation –

"Evil spirits" –

is the only thing that comes to their minds.

They run.

The key to success is good interaction

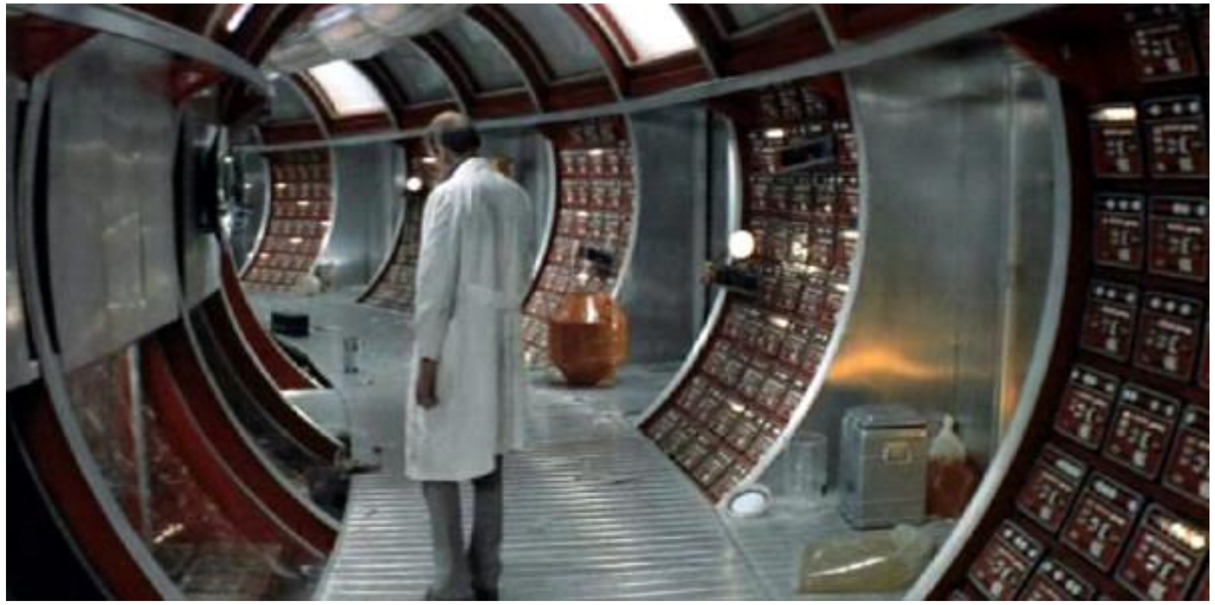
If a cat, a dog, a rooster, a donkey acted alone (like a crowd) - they would not have success.
But the thoughtful joint action described above brings success.

Nobody is afraid of a cat and a rooster

S. Lemm uses this technique

Lemm uses this technique in his novel Solaris.

The appearance of creatures ("guests") of an incomprehensible nature on a space station causes great fear. Because the nature of the creatures - the woman, the little monster - is incomprehensible: they could be called phantoms if they were not entirely material.



Still from Tarkovsky's film "Solaris"

In another Lem novel, "The Inquest," corpses systematically disappear in a morgue in a rural hospital in the outskirts of London. The strangest, most inexplicable and terrifying thing is that a small kitten is always found - and this kitten is more frightening than the monster.

The incomprehensible is frightening

So the thoughtful joint action described above brings success.

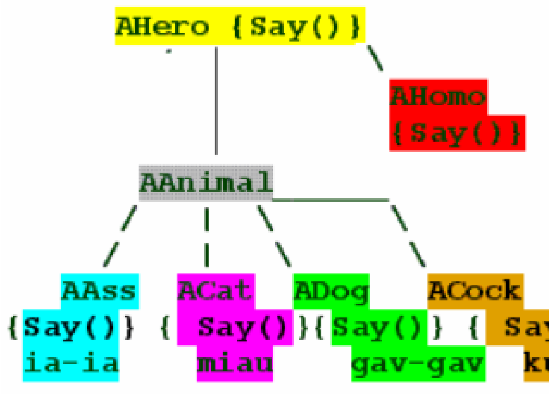
The joint action of different objects is implemented using the mechanism of polymorphism
(not only in programming,

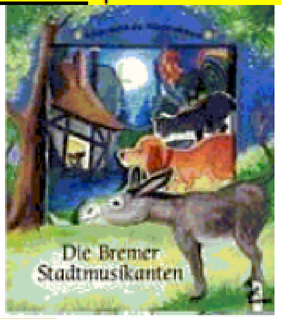
but also in an anthill,
a beehive,
in the army,
at an enterprise).
If a cat, a dog, a rooster, a donkey acted alone (like a crowd) - they would not have success.
But the thoughtful joint action described above brings success.

Brüder Grimm Die Bremer Stadtmuſikanten O. Herrfurth pinx

In my opinion, with the help of this task you can feel what polymorphism (and function reloading) is, and understand this mechanism at a deep (essential) level.

```
class AHero    { virtual public void  Say(){Console.WriteLine("Hi"); } }
  class AAnimal : AHero {    }
```





Die Gebruder Grimm

```
class AHomo : AHero
{
 String name;
 String job;
 public AHomo(String name, String job)
   { this.name = name; this.job = job; }
 override public void Say()
  { Console.WriteLine("Hello"); }
    }
```

```
class AAss : AAnimal {
        override public void Say() { Console.WriteLine("Ii-aa"); }
     }
```

```
class ACat : AAnimal
{
override public void Say() {  Console.WriteLine("miau-miau"); }
}
```

```
class ADog : AAnimal
    {
         override public void Say() { Console.WriteLine("gav-gav"); }
    }
```

```
class ACock : AAnimal
 { override public void Say() { Console.WriteLine("ku-ka-re-ku"); }  }
```

```
ACat says miau-miau
ADog says gav-gav
AAss says Ii-aa
ACock says ku-ka-re-ku
AHomo says Hello
```

```
ACat says gav-gav
ADog says Ii-aa
AAss says ku-ka-re-ku
ACock says Hello
AHomo says miau-miau
```

```
class ADieBremenStadtMusicanten{
static void Main(string[] args){
   ACat cat = new ACat();
   AAss ass = new AAss();
   ADog dog = new ADog();
   ACock ck=new ACock();
   AHomo h = new AHomo("1","Trompeter");
   AHero[] ah = new AHero[5];
```

```
ah[0] = cat;
ah[1] = dog;
ah[2] = ass;
 ah[3]=  ck;
ah[4] = h;
int j;
int k = 0;
```

```
for (int i = 0; i < 5; i++)  {
Console.Write(" "+ah[i].GetType().Name+ " says ");
  ah[i].Say(); }
Console.BackgroundColor = ConsoleColor.Blue;
while (k<5) {
Console.Write("  "+ah[k].GetType().Name+" says ");
     j = k++ < 4 ? k : 0;
     ah[j].Say();       }
```

Let's create classes for a cat and a dog in ES6 (Javascript)

```
class ACat {
  Say() {
     return "Miow";
  }
}
```

```
class ADog {
  Say() {
     return “Woof";
  }
}
```

let's add a type property that returns the class type

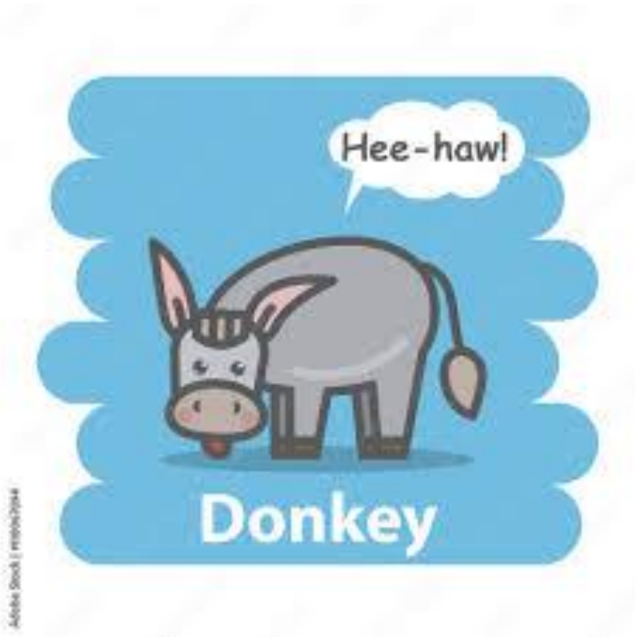
```
class ACat {
  Say() {
     return "Miow";
  }
  get type(){
      return “Cat”;
  }
}
```

```
class ADog {
  Say() {
     return “Woof";
  }
 get type(){
      return “Cat”;
}
```

Let's create classes for a Roster and a Donkey in ES6 (Javascript)

```
class ARooster {
  Say() {
     return
    “Cock-a-doodle-do";
  }
  get type(){
    return “Rooster”;
  }
}
```

```
class ADonkey {
  Say() {
     return “Hee-haw";
  }
 get type(){
      return “Donkey”;
}
```



```
for(var i=0; i<bremen.length; i++) { alert(bremen[i].cnam + " says " + bremen[i].Say()); }
```

Let's unite animals into one structure

```
bm=[new ADonkey(), new ADog(), new ACat(), new ARooster];
for(var i=0; i<bm.length; i++)
{
  alert(bm[i].type + " says " + bm[i].Say());
}
```

Each of our animals says what it should say.

But we need them to say other people's lines.

This can be done like this

```
for(var i=0; i<bm.length; i++)
{
  k=i+1;
  j=k<bm.length?k:0;
  document.write(bm[i].type + " says " + bm[j].Say());
}
```

```
class ACat {
  Say() {
     return "Miow";
  }
  get type(){
      return "Cat";
  }
}
```

```
class ARooster {
  Say() {
     return "Cock-a-doodle-do";
  }
  get type(){
    return "Rooster";
  }
}
```

```
class ADog {
  Say() {
     return "Woof";
  }
 get type(){
      return "Dog";
  }
}
```

```
class ADonkey {
  Say() {
     return "Hee-haw";
  }
 get type(){
      return "Donkey";
}
}
```

```
bm=[new ADonkey(), new ADog(), new ACat(), new ARooster()];
for(var i=0; i<bm.length; i++)
{
  document.write(bm[i].type + " says " + bm[i].Say()+"<br>");
}

for(var i=0; i<bm.length; i++)
{
  k=i+1;
  j=k<bm.length?k:0;
  document.write(bm[i].type + " says<b> " +
bm[j].Say()+"</b><br>");
}
```

Donkey says **Woof**
Dog says **Miow**
Cat says **Cock-a-doodle-do**
Rooster says **Hee-haw**

JS Output

From your JavaScript studies in previous courses, you used the alert() popup function

```
for(i=0;i<2;i++){
alert(" "+ family[i].WashDishes());}
```

The examples in this lecture used the document.write construct to output text to the end of an HTML page.

```
for(i=0;i<2;i++){
document.write(" "+ family[i].WashDishes()); }
```

querySelector

document.querySelector

www.asp.net.by/Projects/1/1.1.querySelector.htm

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Count</title>
        <script>
            let counter = 0;
            function count() {
                counter++;
                 document.querySelector('h1').innerHTML = counter
                //alert(counter);
            }
        </script>
    </head>
    <body>
        <h1>Hello!</h1>
        <button onclick="count()">Count</button>
    </body>
</html>
```

www.asp.net.by/Projects/1/1.2.querySelector.htm

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Count</title>
        <script>
            let counter = 0;
            function count() {
                heading= document.querySelector('h1');
                counter++;
                heading.innerHTML = counter;

            }
        </script>
    </head>
    <body>
        <h1>Hello!</h1>
        <button onclick="count()">Count</button>
    </body>
</html>
```

The same construction using the getElementById() directive

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Count</title>
        <script>
            let counter = 0;

            function count() {
               counter++;
              document.getElementById('h1_1').innerHTML =
                                                                                  counter;
               }
        </script>
    </head>
    <body>
        <h1 id="h1_1">Hello!</h1>
        <button onclick="count()">Count</button>
    </body>
</html>
```