

# JĘZYKI WYSOKIEGO POZIOMU: $\lambda$ – FUNKCJE I PROGRAMOWANIE BEZKLASOWE

**Sebastian Poręba**

# 1. WPROWADZENIE

# Klasyfikacja języków programowania

## Ze względu na:

- Łatwość nauki
- Popularność
- Poziom abstrakcji języka

## Przykłady:

- PHP4 – łatwa nauka
- Assembler, Cobol – niski poziom abstrakcji
- C++, Java – wysoki (?) poziom abstrakcji, akceptowalny poziom trudności



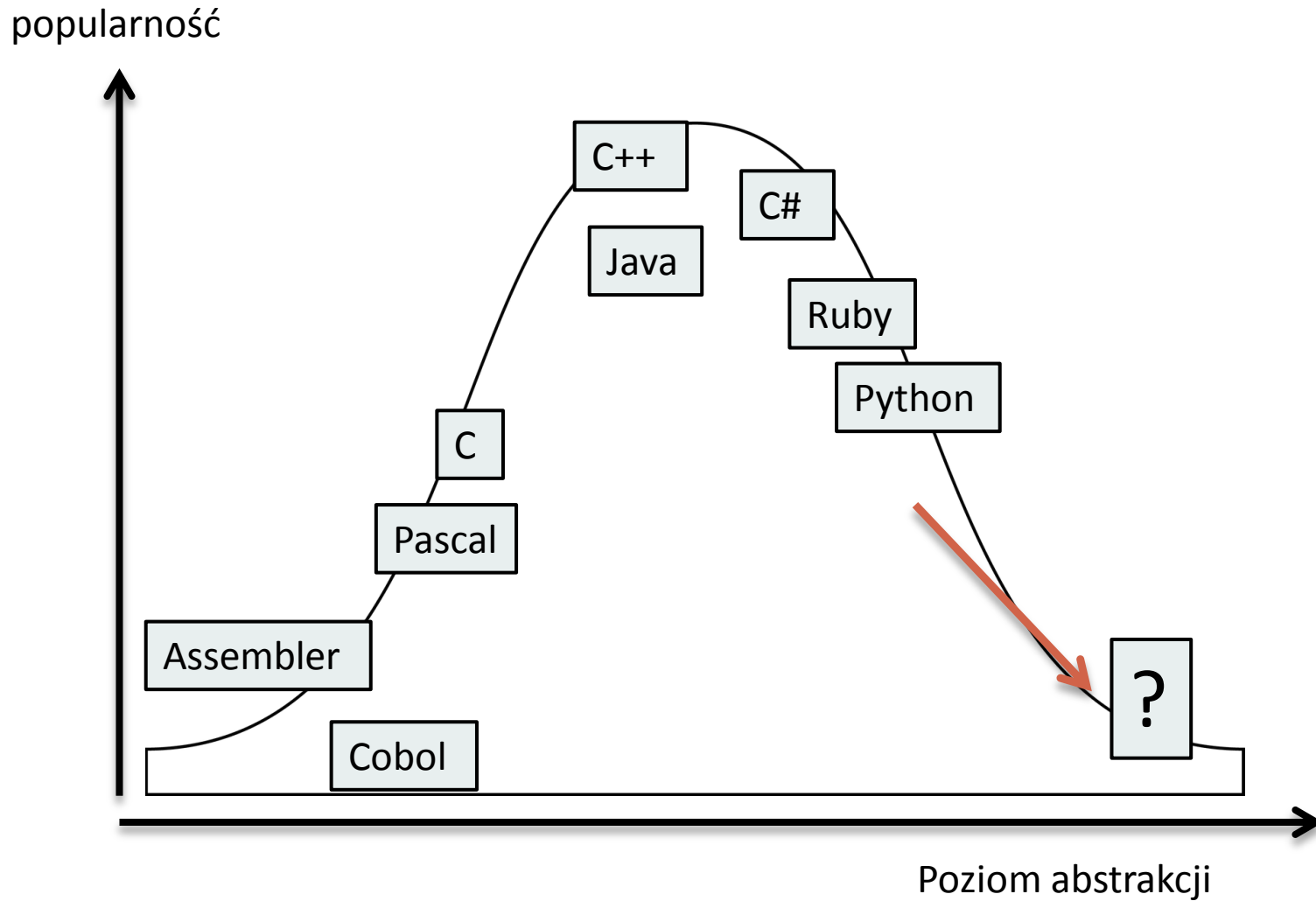
niski

Poziom abstrakcji

wysoki



# Klasyfikacja języków programowania



## Po co programićcie gier języki wyższego poziomu?

DirectX  
OpenGL  
performance



gamedev == C++  
(Java, C, Assembler)

Języki skryptowe w  
silnikach gier

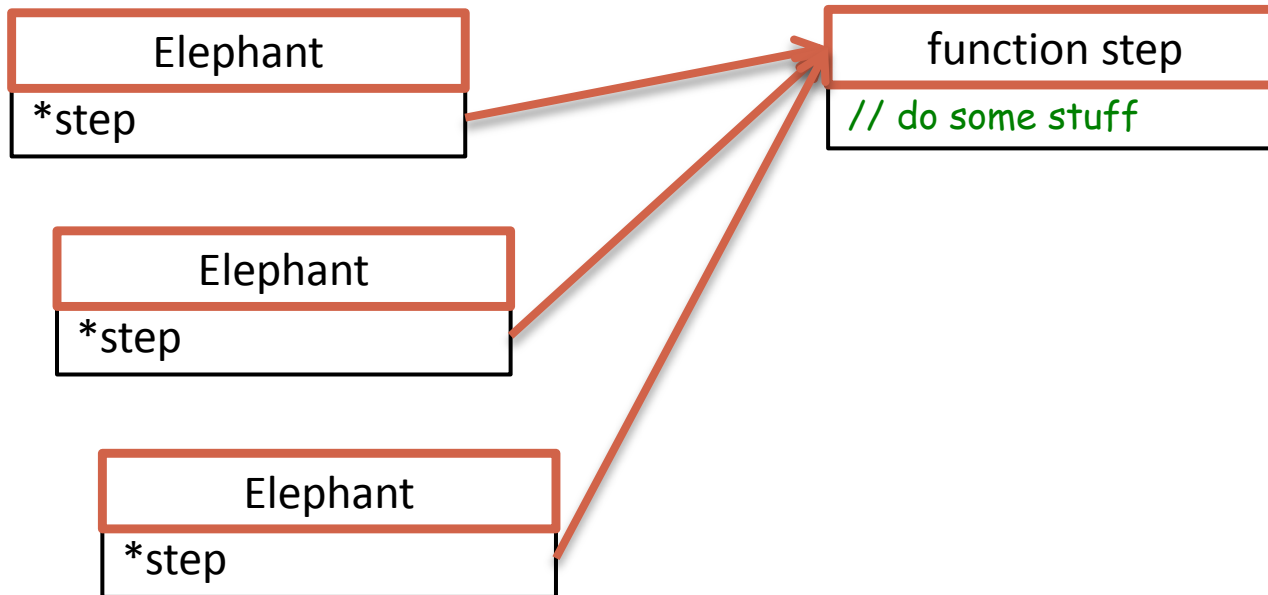


QuakeC, UnrealScript  
Lua, Python

## 2. $\lambda$ – FUNKCJE

## Co się dzieje kiedy definiujemy funkcję w C++

```
class Elephant {  
    void step() {  
        // do some stuff  
    }  
};
```



Funkcje lambda pozwalają na samodzielne zarządzanie wskaźnikiem do funkcji

```
step = function () {  
    // do some stuff  
}  
  
step();
```

Funkcję można:

- zmieniać
- zwracać jako wynik innej funkcji
- podawać jako argument do funkcji

## Zmiana wskaźnika na funkcję

C++:

#define RK4  
#ifdef, #ifndef



Niewygodne  
zarządzanie kodem  
Zarządzanie kodem  
tylko przy kompilacji

If/switch  
wybierający funkcję



ify przy każdej iteracji  
pętli programu

Przechowywanie  
niepotrzebnych  
funkcji w pamięci

### Funkcja lambda

- Podmiana funkcji w locie
- Brak ifów
- Zwolnienie pamięci po funkcji do której nie ma żadnego wskaźnika

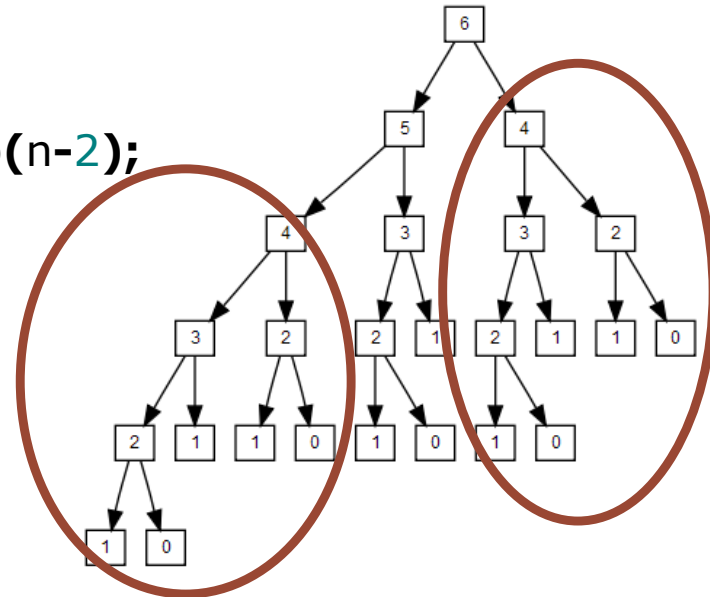
Możliwość zwracania funkcji jako wynik funkcji

Przykład:

funkcja zwracająca algorytm kompresujący na podstawie parametrów pliku do skompresowania

```
int fib(int n) {  
    return n <= 2 ? 1 : fib(n-1) + fib(n-2);  
}
```

fib(100) -> 708 \* 10<sup>18</sup> wywołań fib( )



```
long long memo[101]; // zainicjalizowane na 0  
int memofib(int n) {  
    if(memo[n]) return memo[n];  
    memo[n] = memofib(n-1) + memofib(n-2);  
    return memo[n];  
}
```

## Funkcja jako argument – modyfikatory funkcji

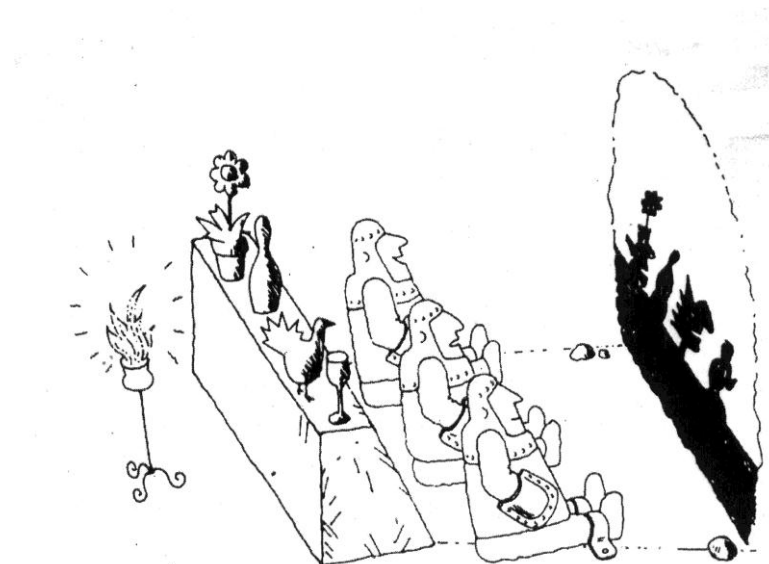
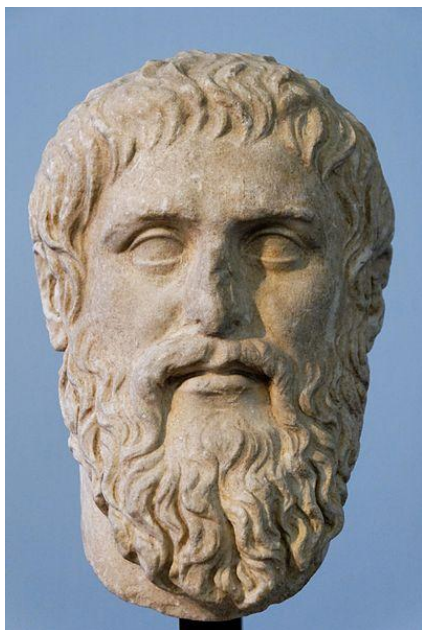
```
function memoize (func) {  
  var memo = {};  
  return function (key) {  
    if(!memo[key]) {memo[key] = func(key); }  
    return memo[key];  
  };  
};
```

```
var fib = function (n) {return n<=2 ? 1 : fib(n-1) + fib(n-2);}  
var memofib = memoize(fib);  
memofib(100);
```

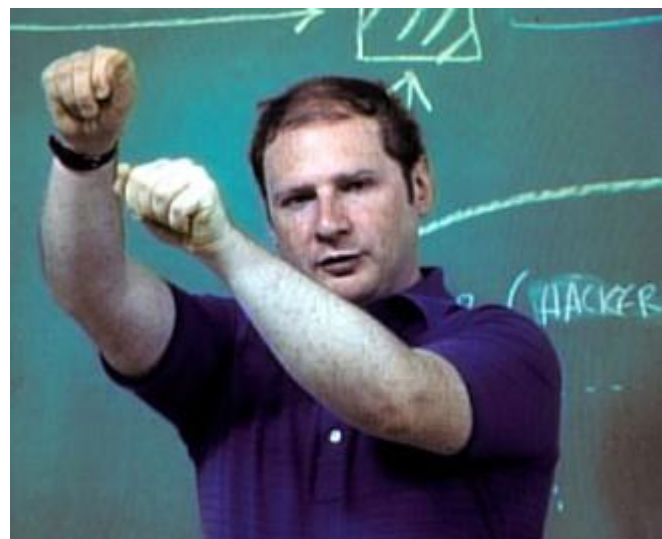
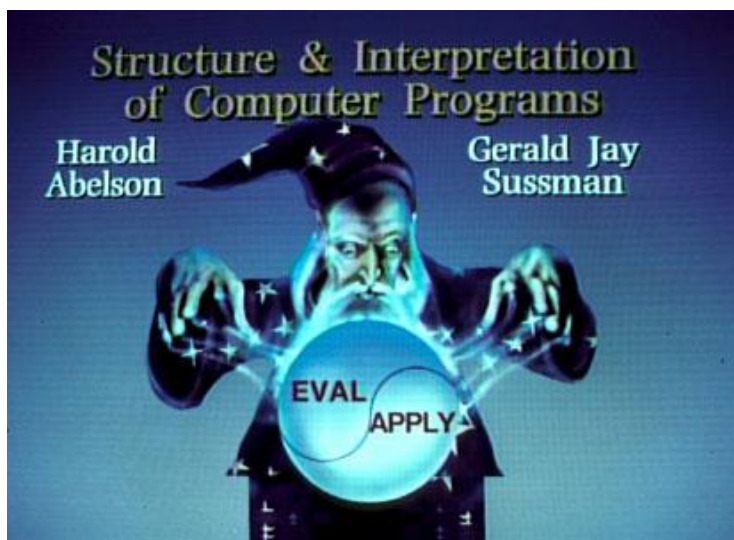
Wprowadzenie tak nieznaczej  
modyfikacji jak funkcje lambda  
udostępnia cały wachlarz nowych  
technik programowania

### 3. PROGRAMOWANIE BEZKLASOWE

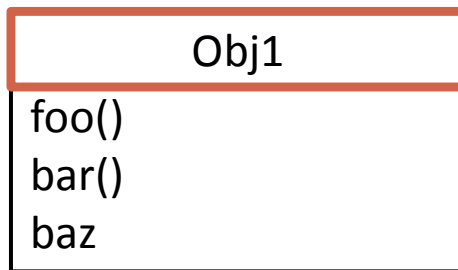
# Platon



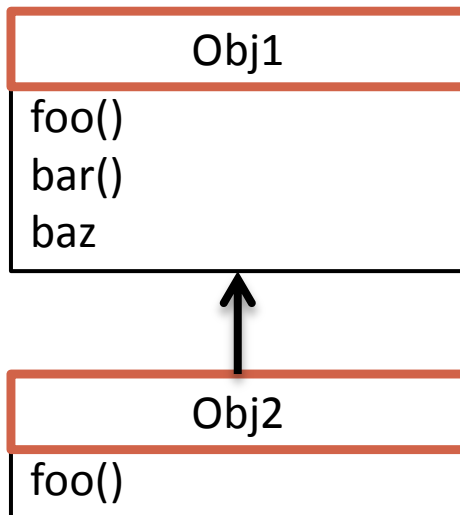
## Dlaczego w takim razie programujemy z wykorzystaniem klas?



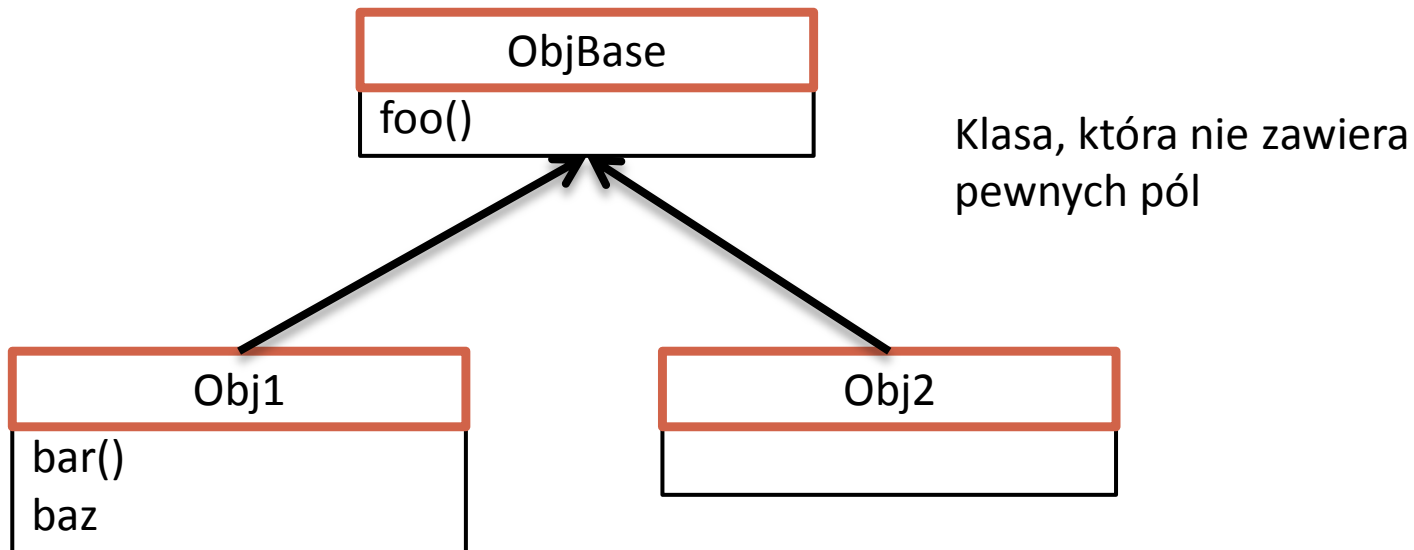
MIT – Structure and Interpretation of Computer Programs,  
Wykład 2b, ~36:00  
prof. Harold Abelson (rok 1986)



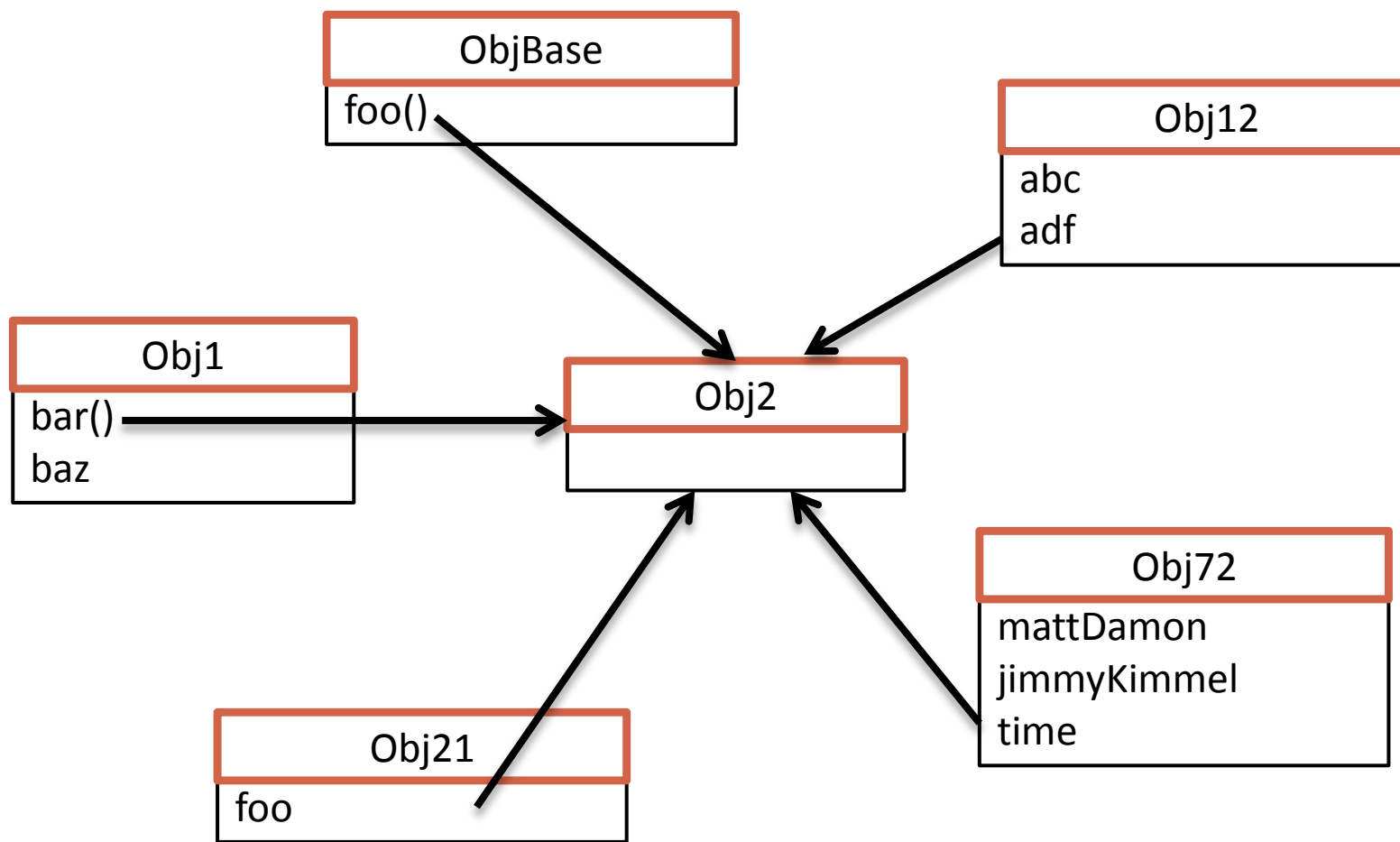
## Dziedziczenie klasowe



Klasa, która wykonuje pewne zadania w inny sposób



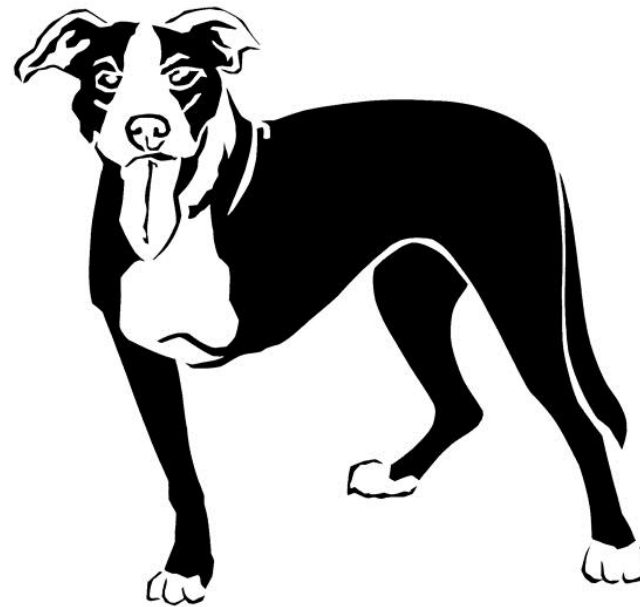
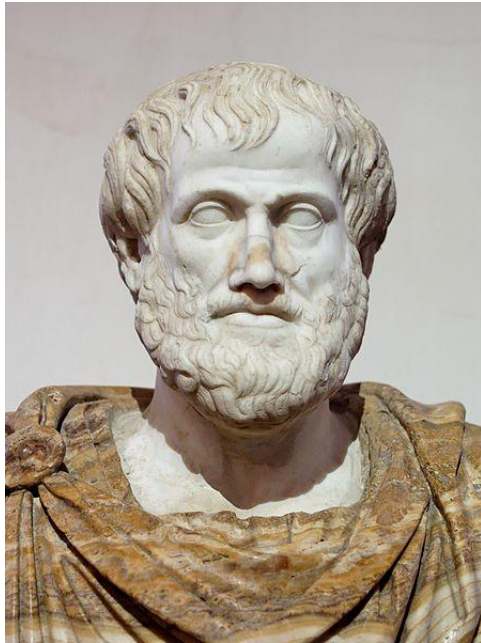
# Dziedziczenie klasowe



Różne pola w różnych klasach



# Arystoteles



## Programowanie klasowe vs bezklasowe

### Klasowe

- Definiujemy klasę pewnych obiektów i zależności między nimi
- Tworzymy instancje klasy
- Obiekty zawierają pola swoje i swoich klas nadrzędnych oraz tablicę wskaźników na metody

### Bezklasowe

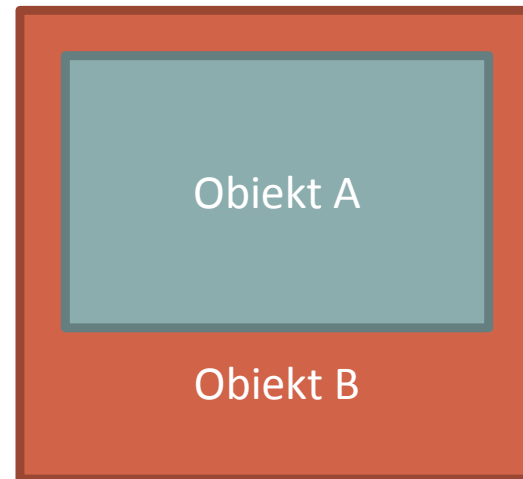
- Tworzymy obiekty
- Obiekty zawierają pola, wskaźniki na swoje metody oraz wskaźnik na prototyp

```
class A {  
int pamparam;  
method abc (param : int) { ... }  
}
```

```
class B inheriting A {  
method abc (param: int) { ... }  
}
```

```
b = new B( );
```

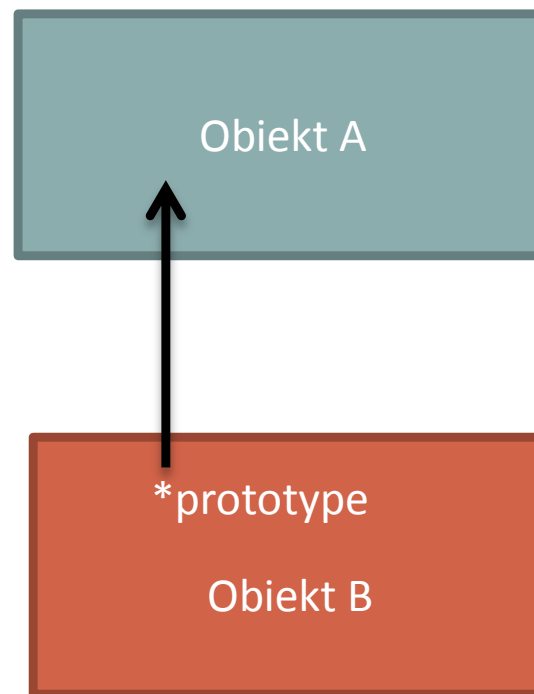
## Enkapsulacja



```
a = {  
  int pamparam;  
  method abc (param : int) { ... }  
}
```

```
b = {  
  method abc (param: int) { ... }  
}  
b.prototype = a;
```

## Wskaźnik na prototyp



## Jaka jest praktyczna różnica?

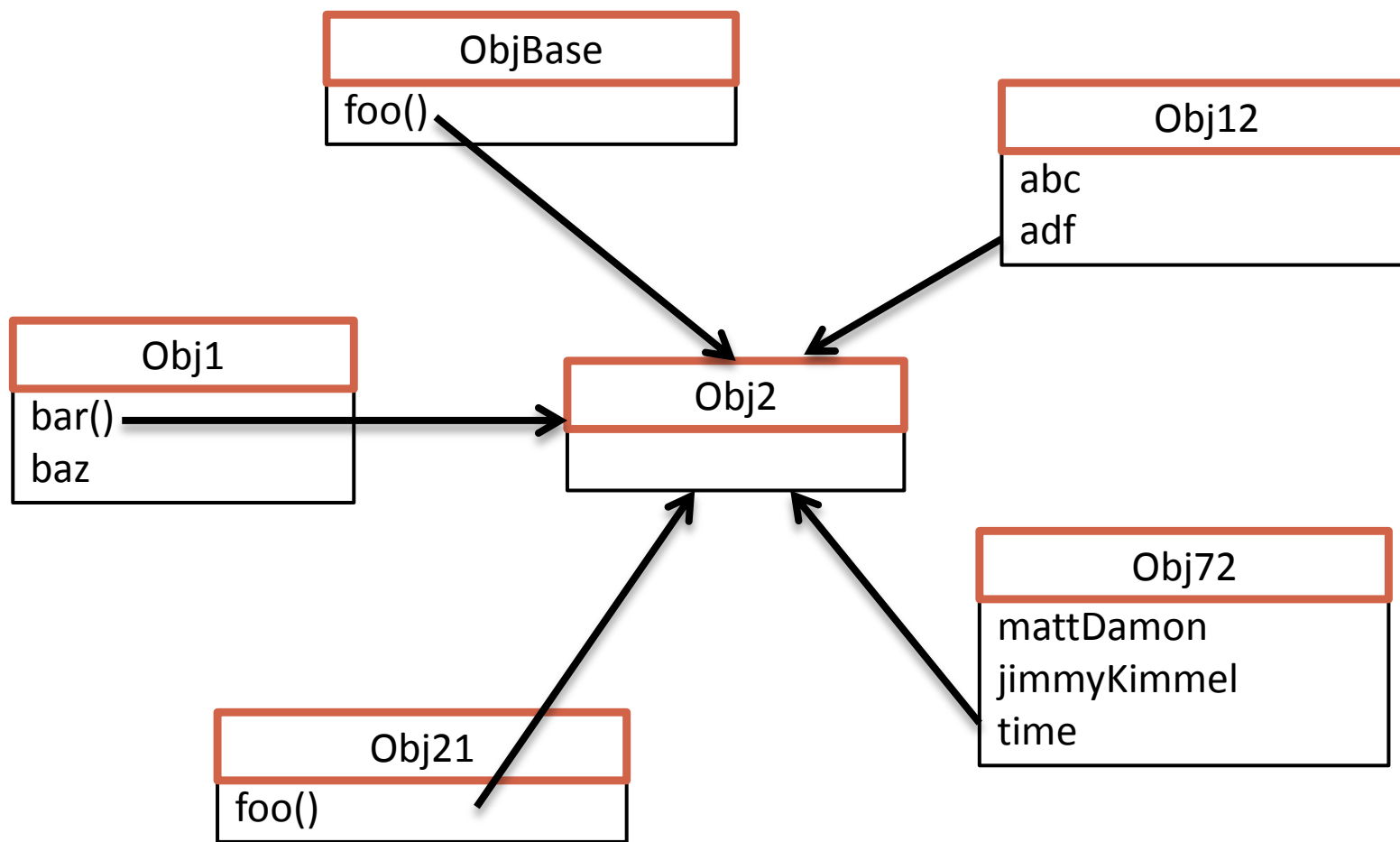
### Klasowe

- Każdy obiekt klasy potomnej zawiera w sobie obiekt klasy nadrzędnej
- Obiekty klas dziedziczących po innych klasach mogą zawierać pola, które nie są używane

### Bezklasowe

- Obiekty przechowują jedynie wskaźnik na obiekt nadrzędny
- Jeżeli odczytujemy wartość pola pampararam (z przykładu, w klasie nadrzędnej), jest on szukany w prototype chain
- Jeżeli chcemy zapisać wartość pampararam, zostanie utworzone nowe pole w obiekcie B, prototype chain nie będzie potrzebny

# Dziedziczenie klasowe



Różne pola w różnych klasach

```
Obj2 = {};
```

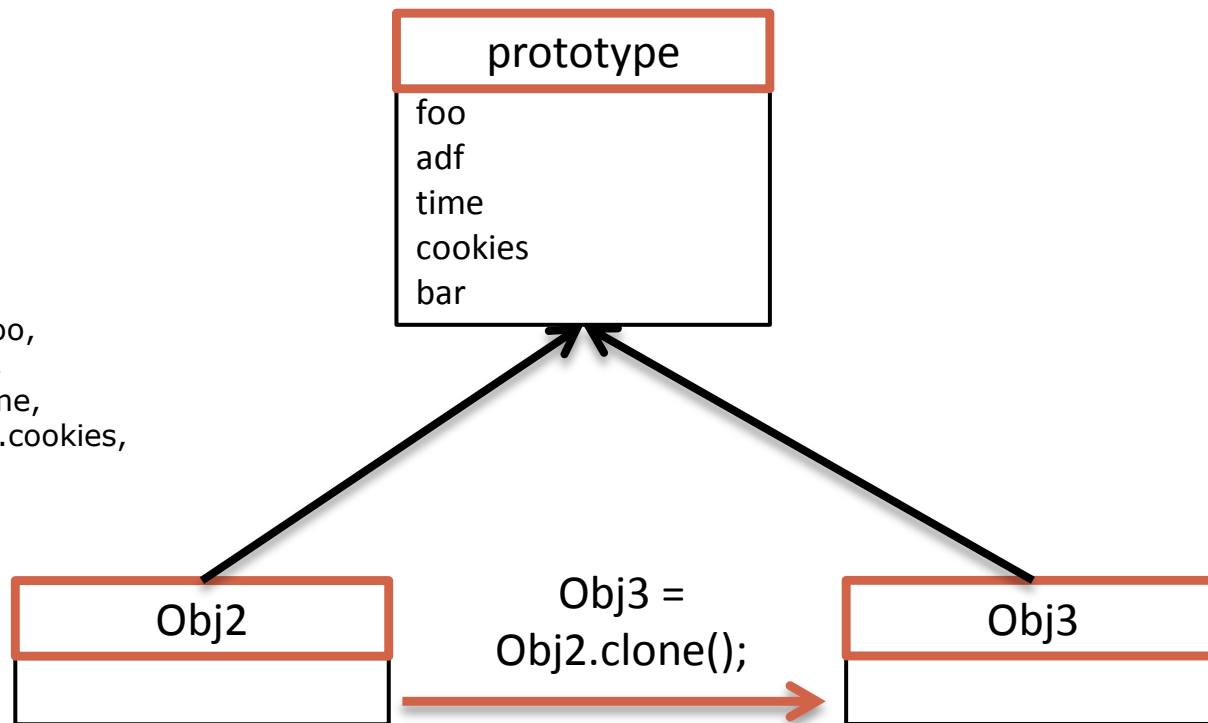
```
Obj2.prototype = {  
  foo: ObjBase.foo,  
  adf: Obj12.adf,  
  time: Obj72.time,  
  foo2: Obj21.foo,  
  bar: Obj1.bar  
};
```

Możemy stworzyć nowy, złożony obiekt  
nie modyfikując w ogóle innych obiektów.

A co jeśli chcemy mieć wiele obiektów z tymi samymi metodami?

```
Obj2 = {};
```

```
Obj2.prototype = {  
  foo: ObjBase.foo,  
  adf: Obj12.adf,  
  time: Obj72.time,  
  cookies: Obj21.cookies,  
  bar: Obj1.bar  
};
```

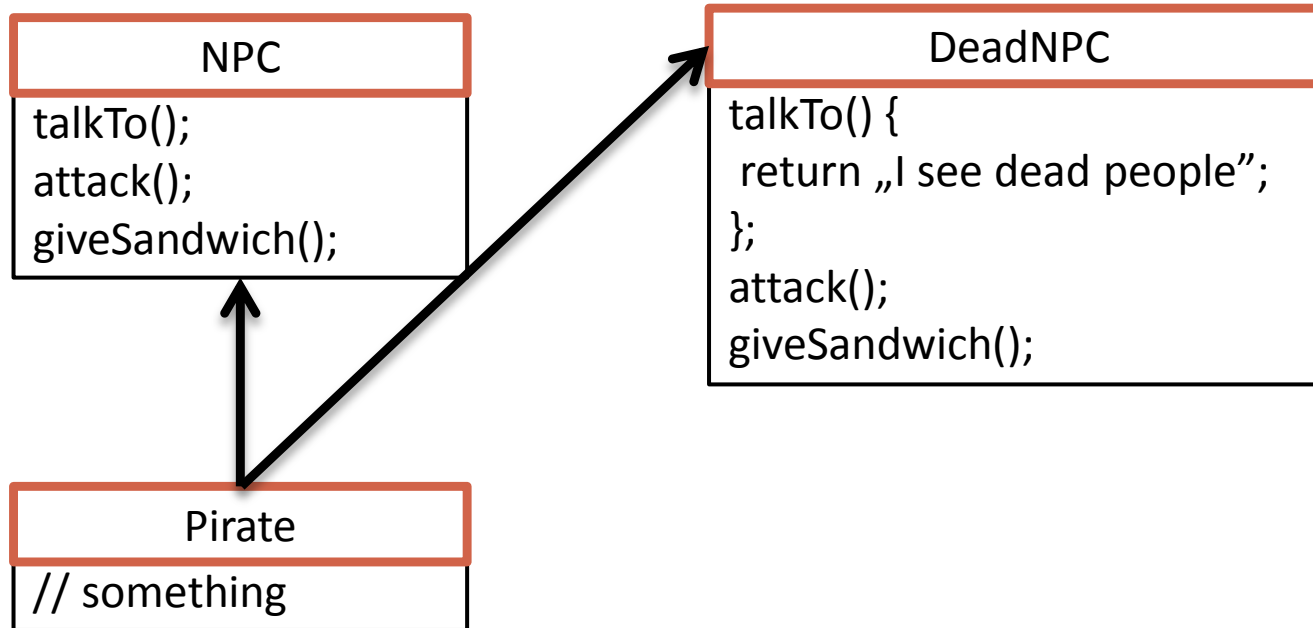


Prototyp obiektu nie ma nazwy, nie  
zaśmieca namespace – jedynym  
odwołaniem do niego jest wskaźnik w  
obiektach Obj2 i Obj3.

Prototyp obiektu może zawierać zmienne dzielone dla wielu obiektów. W C++ zmienne klasy nadrzędnej mają rezerwowaną osobną pamięć dla każdej instancji.

Prototyp obiektu jest określony za pomocą wskaźnika – nic nie stoi więc na przeszkodzie, żeby zmienić go w czasie wykonywania programu.

## Zmiana delegacji obiektu



```
If(Pirate.isA(DeadNPC)) { // do stuff }
```

## Benchmark kodeka MPEG 1/2 – C vs Lisaac

	C version	Lisaac version	Rate (%)
Code line count	9852	6176	37,31 % less
binary size	99 kB	109 kB	10,10% more
Memory used at runtime	1352 kB	1332 kB	
Run time (s)	3,60	3,67	1,94 % more

Poznawanie nowych języków  
programowania może być pożyteczne!

1. **MIT SICP 101** - <http://www.academicearth.org/>
2. Douglas Crockford, YIU Theater - <http://crockford.com/javascript/>
3. Funkcje Lambda w C#  
<http://blogs.msdn.com/b/sriram/archive/2005/08/07/448722.aspx>
4. Antero Taivalsaari, Classes vs. Prototypes:  
Some Philosophical and Historical Observations  
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.4713&rep=rep1&type=pdf>
5. Prototypes with multiple dispatch - <http://tunes.org/~eihrul/ecoop.pdf>
6. Lisaac - <http://isaacos.loria.fr/li.html>
7. Lua - <http://www.lua.org/>

Q&A?