Objektorientiertes Programmieren in OCaml

Fabian Streitel

Proseminar “Programming Languages from Hell”
Sommersemester 2010
TU München

Objekte

```
let lilly =
    object
        val name = "Lilly"
        method speak =
          print_string ("Hi, I'm " ^ name)
        method name = name
    end


lilly#speak
```

```
<speak : unit; name : string>
```

```
class person init =
    object
        val name = init
        method speak =
          print_string ("Hi, I'm " ^ name)
        method name = name
    end
```

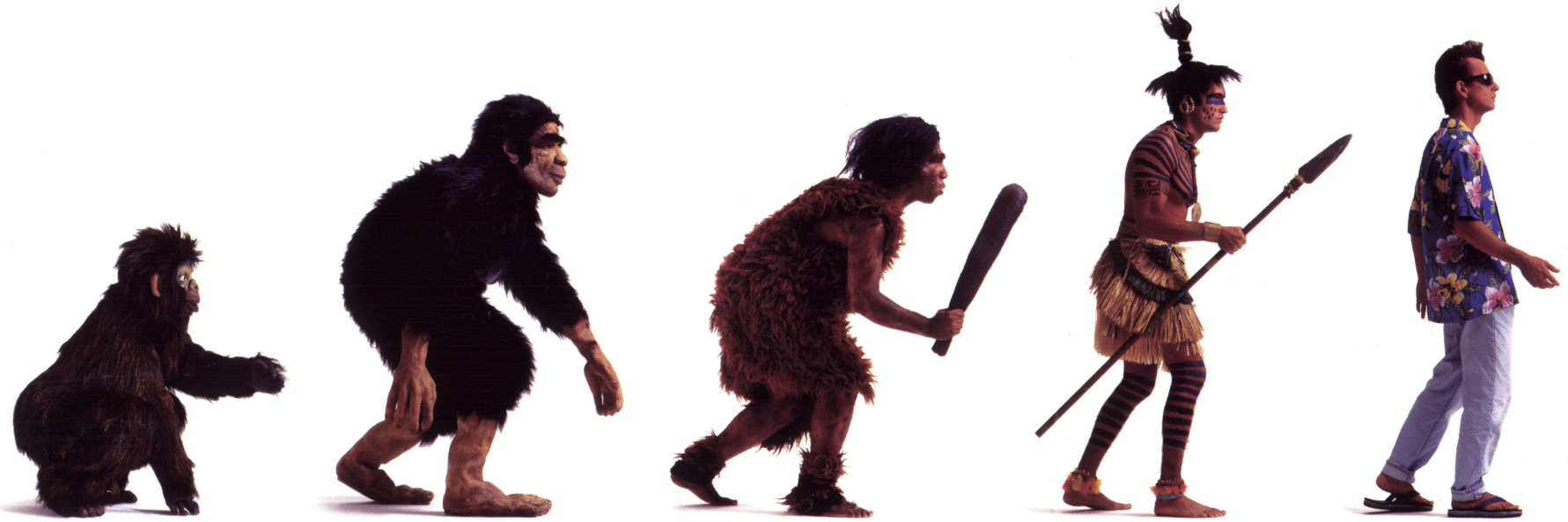
```
class person init =
    object
        val name = init
        method speak =
          print_string ("Hi, I'm " ^ name)
        method name = name
    end

let lilly = new person "Lilly"
```

```
class person init =
    object
        val name = init
        method speak =
          print_string ("Hi, I'm " ^ name)
        method name = name
    end

let lilly = new person "Lilly"
```

```
person   ⇒   <speak : unit; name : string>
```

```
string ->
object
    val name : string
    method speak : unit
    method name : string
end
```

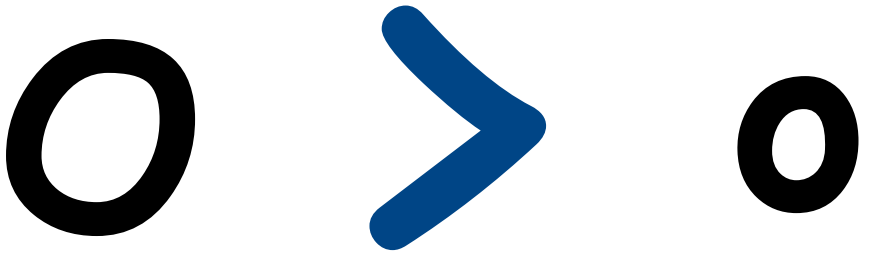
Vererbung

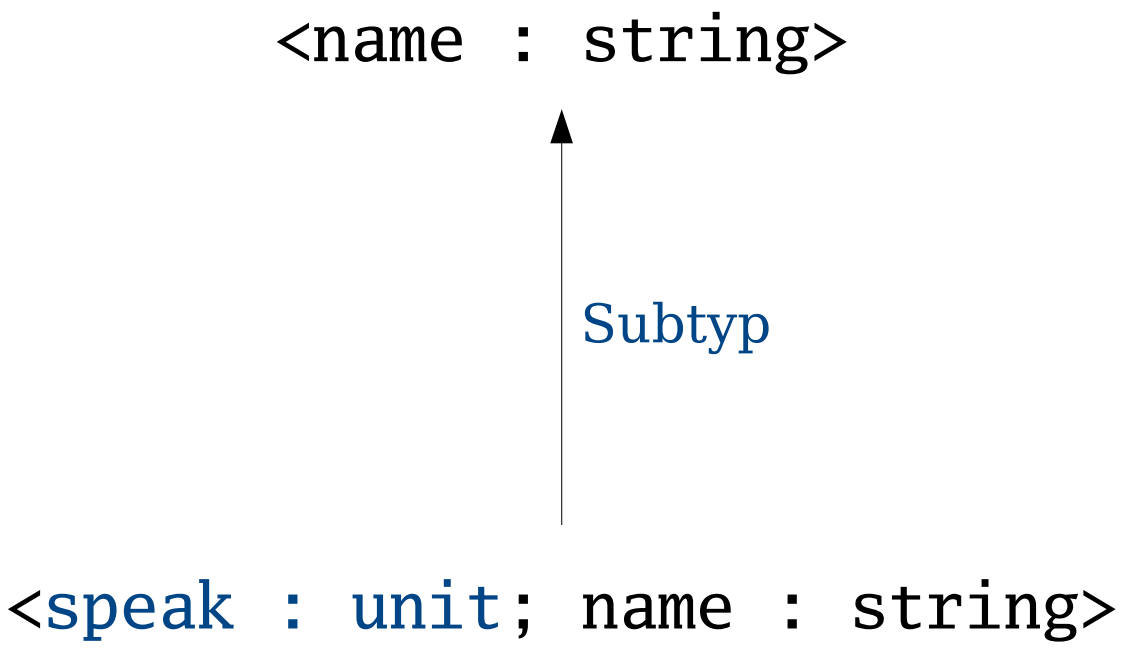


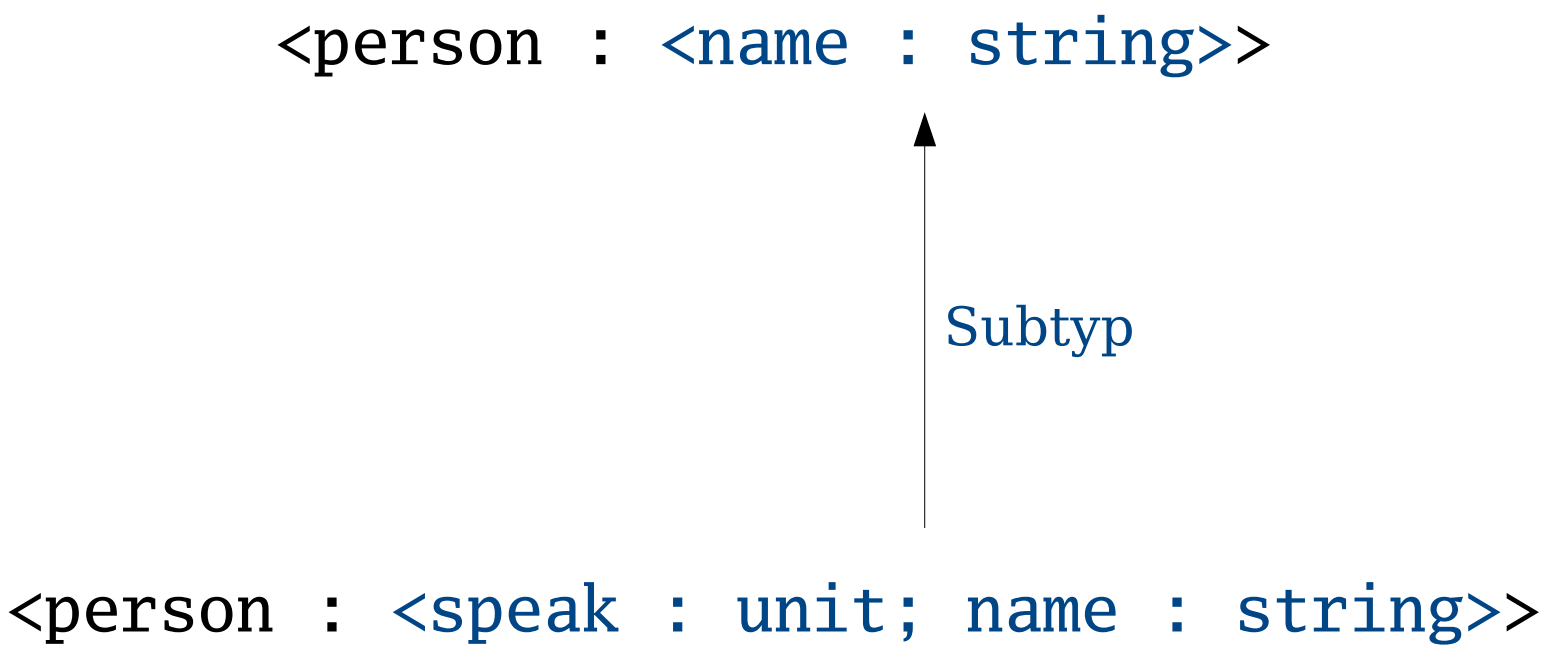
```
class person name =
    object
        method speak =
            print_string ("Hi, I'm " ^ name)
        method name = name
    end

class dog_owner name pet_name =
    object
        inherit person name


        method pet = pet_name
    end
```

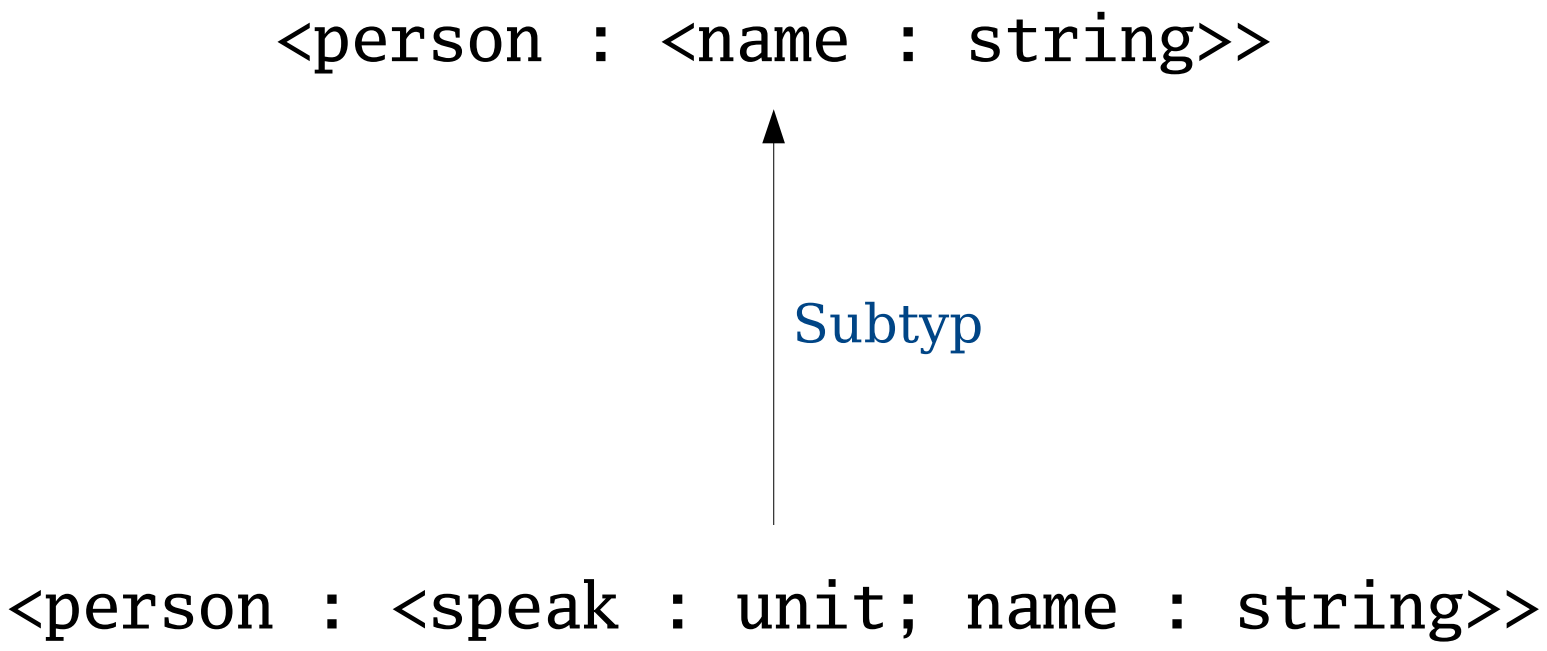
```
class person name =
    object
        method speak =
            print_string ("Hi, I'm " ^ name)
        method name = name
    end

class dog_owner name pet_name =
    object
        method speak =
            print_string ("Hi, I'm " ^ name)
        method name = name
        method pet = pet_name
    end
```

Subtyping



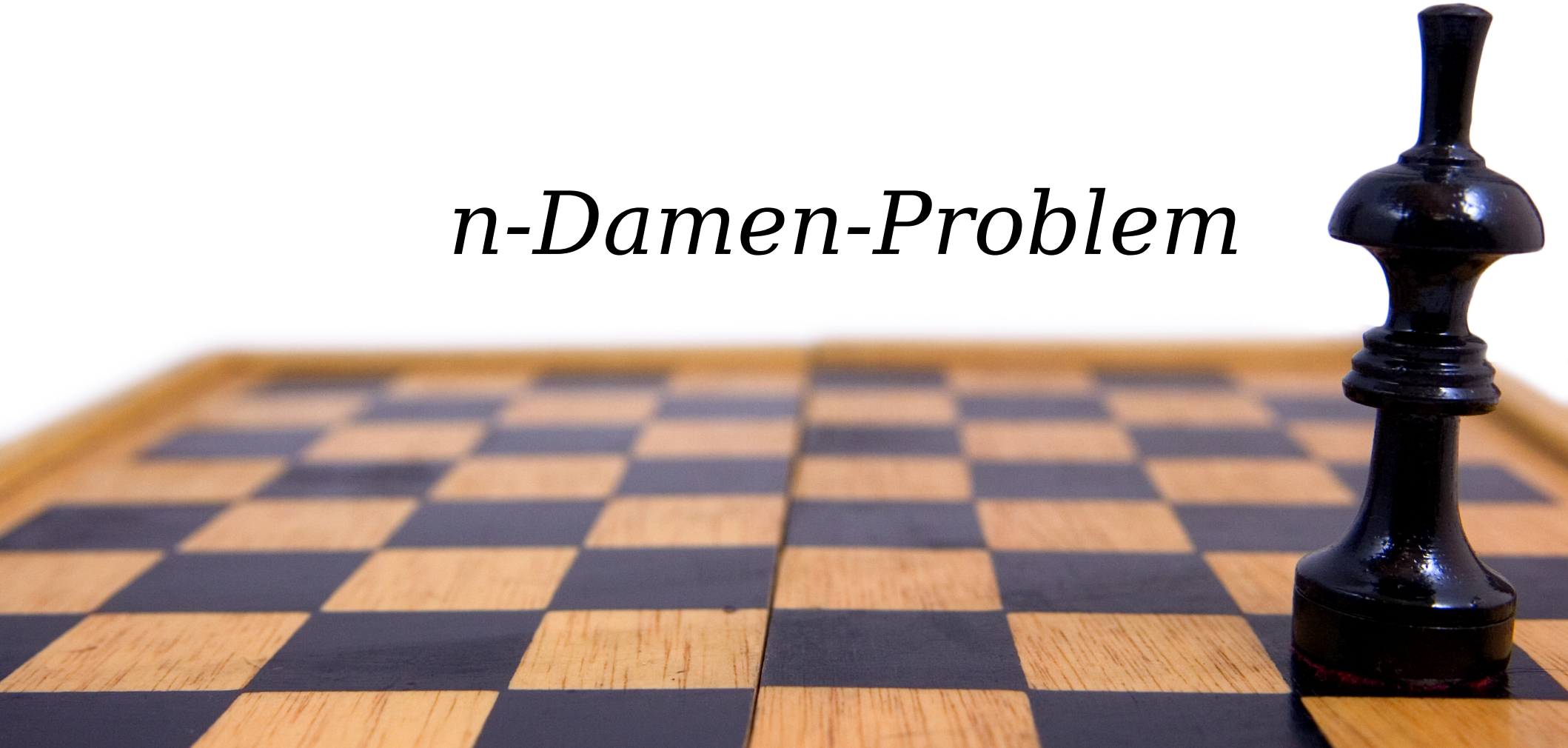

<name : string>
Subtyp
<speak : unit; name : string>

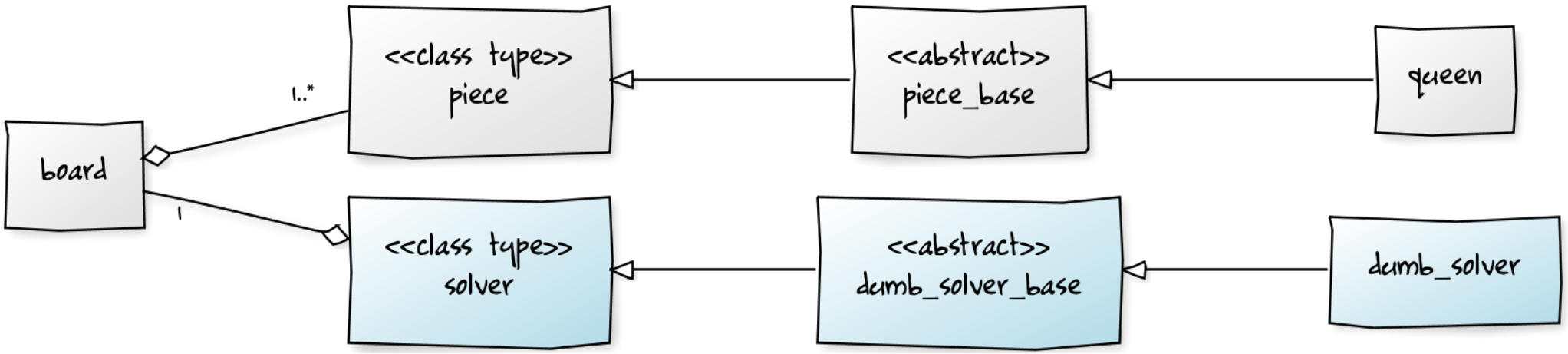

<person : <name : string>>
Subtyp
<person : <speak : unit; name : string>>


<person : <name : string>>
Subtyp
<person : <speak : unit; name : string>>

```
class person =
    object
        method speak =
          print_string "Hi, I'm a person."
    end

class magic_dog =
    object
        method speak =
          print_string "Wuff!"
        method fly =
          print_string "Swooosh!"
    end
```

n-Damen-Problem





Modul Pieces

Modul Solving

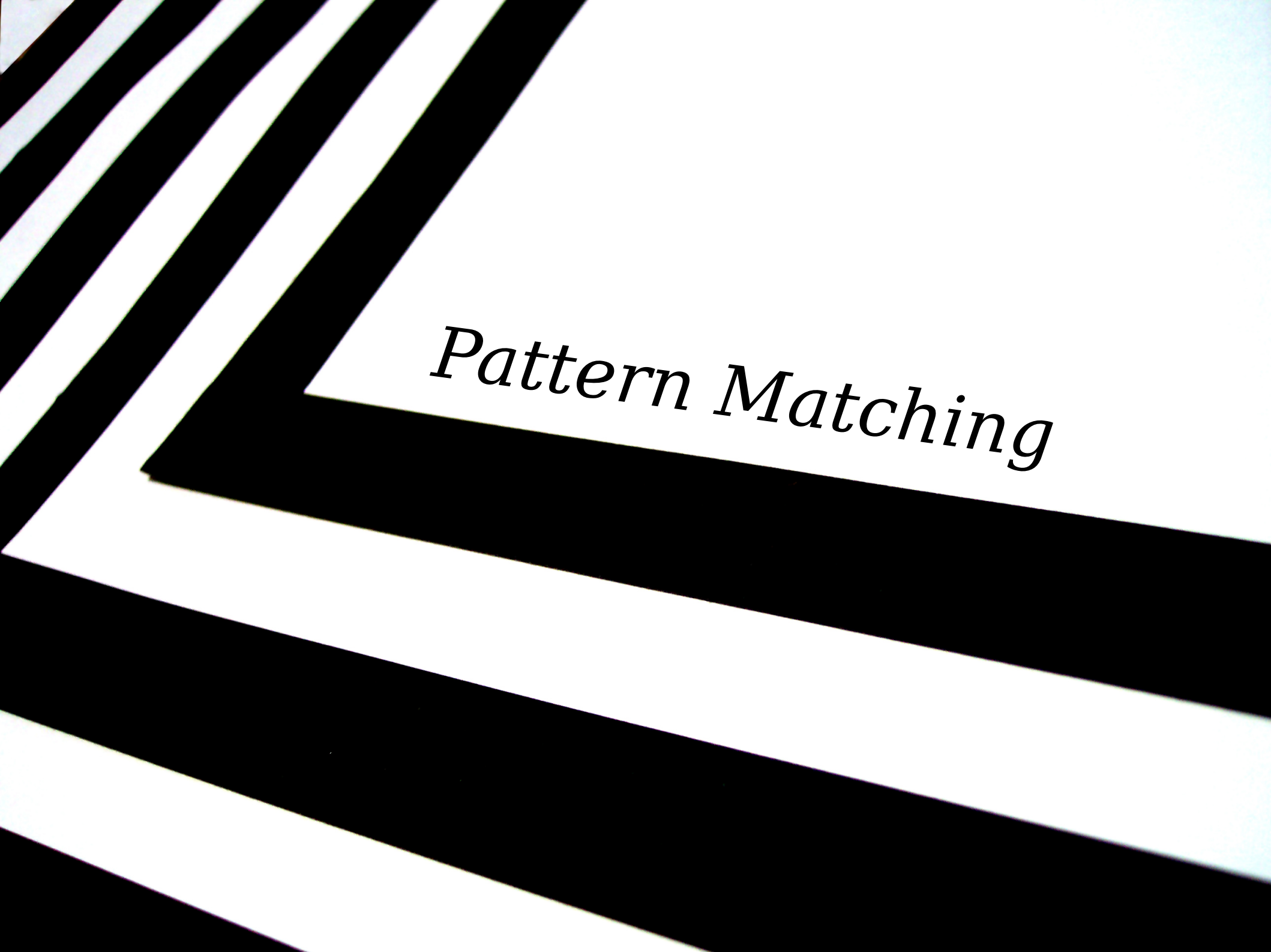
Warum objektorientiert

Probleme

<ziehen : unit>

?

<ziehen : unit>


Pattern Matching

```
class stock (init : float) =
    object
        val mutable repr = init
        method value = repr
        method set_value v = repr <- v
    end
```

Was solltet ihr hier mitnehmen

Was solltet ihr hier mitnehmen

- Werte und ihre Typen

Was solltet ihr hier mitnehmen

- Werte und ihre Typen
- Vererbung vs. Subtyping

Was solltet ihr hier mitnehmen

- Werte und ihre Typen
- Vererbung vs. Subtyping
- Probleme