Duck-Typing

Duck-Typing

If it looks like a duck and quacks like a duck,
it (probably) is one.

Was wie ein Ente aussieht und wie eine quakt,
ist (wahrscheinlich) auch eine.

Duck-Typing

Duck-Typing

- Ein Programmierkonzept:
  - Nicht die Klasse oder Elternklasse überprüfen
  - Sondern verfügbare Eigenschaften und Methoden testen

Implizites “Prüfen”

HPI Hasso Plattner Institut

Passendes Objekt

- Einfach erwartete Methode aufrufen
- Exception bei fehlender Methode oder falschem Verhalten

```
def select(ary)
  new_ary = []

  ary.each do |e|
    new_ary << e if yield e
  end

  new_ary
end
```

Ist `ary` nicht iterierbar, wird eine `NoMethodError`-Exception geworfen

Object#respond_to?

#respond_to?(method_name)

- Antwortet ein Objekt auf eine bestimmte Methode?

```
[].respond_to? :each                          # => true
{}.respond_to? :each                          # => true
File.new("foo.txt", "r").respond_to? :each # => true
"".respond_to? :each                          # => false
```

Object#respond_to?

Explizite Prüfung

- Eigene Exception werfen

```ruby
def select(ary)
  unless ary.respond_to? :each
    raise ArgumentError, "Argument not iterable"
  end

  new_ary = []

  ary.each do |e|
    new_ary << e if yield e
  end

  new_ary
end
```

Object#respond_to?

Explizite Prüfung

- Alternativen Code-Pfad ausführen

```ruby
def make_summary(object)
  if object.nil?
    "..."
  elsif object.respond_to?(:take)
    object.take(3).join(", ")
  else
    object.to_s
  end
end

make_summary nil                  # => "..."
make_summary ["A", "B", "C", "D"] # => "A, B, C"
make_summary Object.new           # => "#<Object:0x00007f60a49ffae0>"
```

Duck-Typing