Test-Fragen: OOP mit Ruby

Bitte versucht die Aufgaben selbstständig zu lösen nicht im Internet Beispiele zu kopieren! Schliesslich soll jeder möglichst viel von dieser praktischen Übung lernen.

WICHTIG: Ihr müsst keine UI's bauen, ausschliesslich über das Terminal die App laufen lassen.

Übung 1:

Erkläre in 4-6 Sätzen, was im folgenden Code passiert. Verwende wenn möglich die entsprechenden Fachbegriffe.

```
class Page
  STATE_INACTIVE = 'inactive'
  STATE_ACTIVE = 'active'


  def initialize(attributes = {})
      @state = attributes[:state] || STATE_INACTIVE
      @title = attributes[:title]
  end

  def publish
      @state = STATE_ACTIVE
  end
end

page = Page.new(:title => 'My first page', :state => Page::
                                                    STATE_INACTIVE)
page.publish
```

Übung 2:

Ein Stack beruht auf dem Prinzip „last in, first out" (LIFO) und ist ein abstrakter Datentyp. Baue in Ruby eine eigene Stack-Implementation nach, und versuche mittels einem main.rb den Stack zu befüllen und wieder verschiedene Elemente auszulesen.

Damit du weisst, welche Eigenschaften du implementieren musst, findest du hier abstrakte Definitionen:

```
init: -> Stack
push(N): Stack -> Stack
pop(N): Stack -> (N or ERROR)
clear: Stack -> Stack
isempty: Stack -> Boolean
size: Stack -> Integer
inspect: Stack -> All Elements comma-separated
```

Starten im main.rb mit folgendem Code:

```
stack = Container::Stack.new([1,2,3,4])
stack.push(5)
```

Erweiterung:
Falls diese Aufgabe keine Herausforderung darstellt, wäre es schön, wenn man anstatt eines Stacks eine PriorityQueue implementieren könnte. Unterschied zum Stack: „ elements are pulled highest-priority-first".

Man kann beim hinzufügen von Elementen in die PriorityQueue noch eine Priorität (als Integer) mitgeben, um die Position innerhalb der Queue zu bestimmen.

```
priority_queue = Container::PriorityQueue.new(["EV Zug",
1])
```

Übung 3:

Ziel dieser Übung ist es, ein fahrtüchtiges Auto mittels Ruby zu konstruieren. Folgende Definitionen sind beim Autobau einzuhalten:

Das Auto hat die Eigenschaften „Farbe" und „Automarke". Das Auto besitzt einen Motor, welcher gestartet und wieder abgestellt werden kann. Zudem hat das Auto 4 Türen. Pro Türe kann man individuell die Fenster runter und wieder hochfahren. Beim Abstellen des Motors werden alle offenen Fenster automatisch hochgefahren.

Damit der Benutzer deines Autos visuell sieht, was passiert, gib bitte folgende Outputs während des Programmablaufs aus.

- Roter Ferrari wurde gebaut
- Motor gestartet
- Der rote Ferrari fährt an (gib Gas lieber Michael Schumacher)
- Fenster auf Fahrerseite (vorne) wurde geöffnet
- Auto wird gebremst
- Motor wird abgestellt (alle offenen Fenster werden automatisch geschlossen)