Informatika pro moderní fyziky (12)
API - dokončení, úvod do OOP, zadání zápočtových úloh

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2020/2021
21. prosince 2020

Obsah

- práci s cizím API
- vytvoření jednoduché aplikace s interaktivní mapou

Obsah

Propojení více tabulek - opakování

- procvičíme: k výpisu všech závodů přidáme i získávání výsledků
- Seznam závodů:
  `https://oris.orientacnisporty.cz/API/?format=json&method=getEventList&sport=3&datefrom=2019-01-01&dateto=2019-12-31`
- Výsledky: `https://oris.orientacnisporty.cz/API/?format=json&method=getEventResults&eventid=2077`
- vyrobte skript, který v letech 2015 až 2019 vypíše všechny závody a pro každý i vítězku ženské elitní kategorie (W21E)

Připomenutí / co potřebujeme

```
require "open-uri"
raw_data = URI.open("https://.....").read
hash = JSON[raw_data]
```

Obsah

Co je OOP (1)

- zatím jsme používali tzv. procedurální programování – máme data a pak procedury/funkce
- Ruby je nicméně čistě objektový jazyk, i když jsme se tomu zatím spíš vyhýbali
- objekt je entita, která má *vlastnosti* (properties) a *metody* (methods) a poskytuje okolnímu světu nějaké *rozhraní* (interface)

Co je OOP (2)

- většinou se jako hlavní důvody pro OOP uvádí polymorfismus a dědičnost (inheritance), ale to hlavní je posun uvažování od dat a operací nad nimi k “inteligentním” objektům – a spolu s tím zapouzdření (encapsulation)
- je potřeba si uvědomit, že všechna ‘zázračná’ paradigmata v programování jsou pouze odlišné formalismy, může nám to hodně pomoct a stojí za to se tomu věnovat, na druhou stranu je potřeba se z toho nezbláznit a nedělat z toho náboženství

Třída

- ‘typ’ objektu - definuju, jak se objekty chovají
- neboli definuju metody
- zjednodušeně řečeno jsou to jen metody a ne data
- potkali jsme třeba `File`, `CSV`

Instance

- konkrétní objekt – má svoje data
- většinou vytváříme pomocí `Class.new`, resp `Class.new(arg1, arg2, ...)`
- k datům přistupuju pomocí *instance variables* – `@data`
- leckdy se hodí speciální metoda `initialize`, tzv. *konstruktor* – volá se při `new` a nastavují se zde výchozí hodnoty vlastností

Class method

- můžu mít i metody, které nepatří k žádné instanci – nepracují s žádnými daty
- dávat je do třídy má pak smysl pouze organizační, nemá to pak žádnou reálnou výhodu proti obyčejným funkcím
- např `File.foreach` nebo `CSV.read`

Jak to vypadá v Ruby

```
class Table
  def initialize
    @data = {}
  end
  def get(key)
    @data[key]
  end
  def set(key, value)
    @data[key] = value
  end
  def print
    @data.each do |key, value|
      puts "#{key} #{value}"
    end
  end
end

t = Table.new
t.set("a", 123)
t.print
```

Obsah

Zadání

- vytvořte skript, který pro zadané chemické vzorce sloučenin:
- spočítá molární hmotnost (g/mol)
- vypíše slovy, z jakých prvků a z kolika atomů se skládá
- (použijte OOP řešení)

Příklad

```
NaCl
2
1 atom of sodium + 1 atom of chlorine
M = 58.443

H2O
3
2 atoms of hydrogen + 1 atom of oxygen
M = 18.015

CH3CH2OH
9
2 atoms of carbon + 6 atoms of hydrogen + 1 atom of oxyg
M = 46.069
```

Jak by to mělo fungovat

```
c = Compound.new("H3BO3")
puts c
puts c.size
puts c.as_text
puts "M = #{c.mass}"
```

Různé rady

- začnu třídou `Compound` a jejím konstruktorem
- (zde využiju metodu `scan` s regulárním výrazem)
- budu chtít vyrobit hash `{"H"=>3,"B"=>1,"O"=>3}` a uložit si ho jako instance variable
- (to jsou moje vlastnosti na třídě `Compound`)
- pak můžu udělat metody `size` a `as_text`

Různé rady (2)

- při počítání molární hmotnosti je třeba odolat několika svodům:
- nenahrávejte tabulku prvků pro každou instanci Compound
- nezavádějte třídu pro jednotlivé prvky
- dobrá rada: vytvořte třídu pro tabulku prvků (nebo prostě udělejte hash), jednu její instanci a tu předávejte do konstruktoru instance Compound

Obsah

Obecně:

- ke každému zadání jsou k dispozici vzorová data
- já to budu testovat i na datech jiných
- očekávám, že všechno proběhne na jedno spuštění skriptu
- každý má k dispozici jeden pokus řádný a jeden opravný

Klasifikace

- F - nejde to spustit, ani pro zadaná data to v podstatných bodech nesplňuje zadání
- E - pro zadaná data to funguje, ale pro jiná čísla to nechodí
- D - obecně to funguje, ale stejně chybí drobnosti ze zadání
- C - všechno funguje jak má
- B - funguje a navíc jsou výstupy hezké a přehledné, soubory nejsou generovány “na velkou hromadu”, ale roztříděné do složek apod.
- A - kromě výše uvedeného jsou splněny i požadavky formy (správné odsazování, rozumná jména funkcí a proměnných) a efektivity (je to rozumně naprogramované - vhodné použití funkcí, datových struktur atd.)

Známka se snižuje o stupeň, pokud:

- jsou někde ve skriptech použity absolutní cesty, takže je budu muset upravovat (výjimkou jsou cesty k programům jako např. gnuplot, které ovšem musí být umístěny v proměnné někde na začátku skriptu (abych to nemusel lovit)
- bude v kódu něco, co limituje použití na OS Windows (backslash v cestě, kódování win1250 atd.)

(a podobně)

A to je vše, přátelé!