23 kwietnia 2020

Programowanie obiektowe

Lista 8.

Poniższa lista zadań jest do zrobienia w języku Ruby. Każde zadanie to 4 punkty. Wybierz 2 zadania.

Zadanie 1. Rozszerz standardową klasę ***Fixnum*** o metody:

- zeroargumentową metodę `czynniki` zwracającą tablicę wszystkich dzielników liczby włącznie z jedynką i nią samą (kolejność nie jest ważna). Przykładowo `6.czynniki` powinno zwrócić tablicę `[1, 2, 3, 6]`;
- jednoargumentową metodę `ack(y)` obliczającą funkcję Ackermanna zdefiniowaną następująco:

$$Ack(n,m) = \begin{cases} m+1 & \text{gdy } n = 0 \\ Ack(n-1,1) & \text{gdy } m = 0 \\ Ack(n-1, Ack(n,m-1)) & \text{w pozostałych przypadkach} \end{cases}$$

  Na przykład `2.ack(1)` powinno dać 5. Uwaga: funkcja ta bardzo długo liczy, nawet dla niedużych argumentów, więc nie testujcie jej na dużych (> 2) liczbach;
- zeroargumentowa metoda `doskonala`, która zwraca `true` gdy liczba jest doskonała[1];
- zeroargumentową metodę zamieniającą liczbę na jej postać słowną. Można przyjąć, że postać słowna jest uproszczona, np. `123.slownie` powinno zwrócić "jeden dwa trzy".

Zadanie 2. Zaimplementuj dwie klasy: ***ImageBW*** i ***ImageC*** implementujące odpowiednio bitmapowe obrazy czarno-białe i kolorowe. Rozmiary obrazów są ustalane przy tworzeniu obiektów jako parametry konstruktora. Zaprogramuj w tych klasach metody `+(arg)`, `*(arg)`, które zwracają nowy obiekt klasy ***ImageBW*** bądź ***ImageC***. Metoda `+(arg)` tworzy nowy obraz, którego każdy piksel jest alternatywą bitową odpowiednich pikseli obiektu i argumentu; odpowiednio `*(arg)` oznacza utworzenie nowego obiektu z koniunkcji bitów pikseli. Takie operacje mają sens, gdy operacje wykonujemy na obrazkach o tych samych rozmiarach i tym samym typie (tj. tylko kolorowe z kolorowymi albo czarno–białe z czarno–białymi). Można przyjąć, że zawsze wykonujemy metody na poprawnych danych. Dodaj też do tych klas metodę `narysuj` rysującą obrazy w postaci ascii-artu.

Zadanie 3. Jedną z najprostszych metod szyfrowania jest szyfr podstawieniowy, w którym za literę podstawia się inną literę. Zaprogramuj dwie klasy:

- klasę ***Jawna*** przechowującą napis w postaci jawnej i implementującą metodę `zaszyfruj(klucz)` zwracającą obiekt klasy ***Zaszyfrowane***;
- klasę ***Zaszyfrowane*** przechowującą napis zaszyfrowany i implementującą metodę `odszyfruj(klucz)` zwracającą obiekt klasy ***Jawna***.

Obydwie klasy winne implementować metodę `to_s`. Argument `klucz` to słownik postaci

```
{ 'a' => 'b',
  'b' => 'r',
  ...
  'r' => 'y',
  'y' => 'u',
  'u' => 'a'
}
```

[1] definicję można znaleźć m. in. w Wikipedii

Dla takiego klucza napis *'ruby'* zostanie zamieniony na *'yaru'*. Można założyć, że będziemy używać tylko małych liter. Nie jest też konieczne sprawdzanie czy klucz (słownik) jest poprawny.

Marcin Młotkowski