

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 *Integer*¹ 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 Ackermanna zdefiniowaną następująco:

$$Ack(n, m) = \begin{cases} m + 1 & \text{where } n = 0 \\ Ack(n - 1, 1) & \text{where } m = 0 \\ Ack(n - 1, Ack(n, m - 1)) & \text{otherwise} \end{cases}$$

Na przykład *2.ack(1)* powinno zwrócić 5. Uwaga: funkcja ta bardzo długo liczy, nawet dla niedużych argumentów, więc nie testujcie jej na dużych (i 2) argumentach;

- zeroargumentowa metoda *doskonała*, która zwraca *true* gdy liczba jest *doskonała*²; na przykład *6.doskonała* powinna zwrócić *true*;
- zeroargumentowa *słownie* zwracającą string będący słownym zapisem liczby. Można przyjąć, że postać słowna jest uproszczona, np. *123.słownie* 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. W każdej klasie zaprogramuj metody

- *+(arg)*, która tworzy nowy obraz (obiekt), którego każdy piksel jest alternatywą bitową odpowiednich piksli obiektu i argumentu;
- metoda **(arg)*, która tworzy nowy obraz (obiekt) poprzez koniunkcję bitów piksli.
- *narysuj* rysującą obrazy w postaci ascii-artu.

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.

Zadanie 3

edną 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(key)* zwracającą obiekt klasy *Zaszyfrowane*;

¹We wcześniejszych wersjach Ruby korzystano z klasy *Fixnum*

²definicję można znaleźć m. in. w Wikipedii

- klasę ***Ciphered*** przechowującą napis zaszyfrowany i implementującą metodę *oszyfruj(key)* zwracającą obiekt klasy ***Jawna***.

Obydwie klasy winne implementować metodę *to_s* zwracającą przechowywany tekst (zaszyfrowany lub nie). Argument *key* jest słownikiem np.

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

Dla takiego klucza słowo *'ruby'* jest szyfrowane *'yaru'*. Dla uproszczenia można ograniczyć się do małych liter.

Marcin Młotkowski