

## 20. Dziedziczenie, domieszki

### 1 Dziedziczenie

W języku Ruby możliwe jest dziedziczenie tylko po jednej klasie:

```
1 class MySubClass < MyClass
2   def initialize(arg_1, arg_2)
3     super(arg_1)           # wywołanie metody #initialize nadklasy
4     @new_attribute = arg_2
5   end
6 end
```

**Zadanie 01** Stwórz klasę `Person`, która będzie posiadać atrybut `@name` do odczytu i jednoargumentową metodę `initialize`, ustawiającą jego wartość.

**Zadanie 02** Stwórz klasę `Player` dziedziczącą z klasy `Person` i rozszerzającą ją o atrybut `scores` (możliwy do odczytu), reprezentujący punkty zdobyte przez gracza, początkowo zainicjalizowany jako pusta tablica. Podczas tworzenia obiektów tej klasy konieczne będzie podanie imienia gracza.

Klasa `Player` powinna również posiadać metody `add_score` (dodaje punkt) oraz `all_scores` (zwraca sumę wszystkich punktów).

### 2 Klasa Object

Klasą nadrzędną dla wszystkich tworzonych klas jest klasa `Object`: <http://ruby-doc.org/core-1.9.3/Object.html>.

**Zadanie 03** W klasie `Player` przeciąż metodę `to_s` tak, aby zwracała imię gracza i sumę jego punktów, np. w postaci `"Foo Bar (3 points)"`. Przetestuj działanie metody.

**Zadanie 04** W klasie `Player` zmodyfikuj metodę `add_score` w taki sposób, aby jej argumentem mogła być zarówno tablica, jak i dowolny inny obiekt reprezentujący punkty. Wskazówka: wykorzystaj `Object#kind_of?` (dla ambitnych: jak można zrealizować powyższą funkcjonalność w prostszy sposób, bez metody `#kind_of?`).

```
1 player = Player.new("Foo", "Bar")
2 player.add_score([1, 2, 3])
3 player.add_score(4)
4
5 player.scores # => [1, 2, 3, 4]
```

### 3 Domieszki

Moduły lub domieszki (ang. *mixins*) to zbiory metod (i stałych), które mogą być dołączane do innych klas.

```
1 module MyModule
2   def my_extra_method
3     puts "Module method!"
4   end
5 end
6
7 class MyClass
8   include MyModule    # dołączenie modułu do klasy
9   # ...
10 end
```

Poprzez wmieszanie modułu do klasy, otrzymuje ona wszystkie metody z tego modułu.

```
1 obj = MyClass.new
2 obj.my_extra_method    # wywołanie metody z domieszki
```

Metody definiowane w module mogą odnosić się do metod lub atrybutów w domieszkowanej klasie (tzn. mogą zakładać, że w tej klasie są zdefiniowane metody lub atrybuty o określonych nazwach). Moduły są również używane jako przestrzeń nazw. Dostęp do stałych (w tym klas) zdefiniowanych wewnątrz modułu uzyskuje się poprzez operator `::`, np. `Math::PI`.

**Zadanie 05** Stwórz moduł `WithScores`, który dostarczy funkcji pomocniczych dla tablicy punktów. W module zdefiniuj następujące metody działające na atrybucie `@scores` (atrybut ten będzie pochodził z klasy, do której będzie wmieszany moduł):

- `#best_score` — zwróci najwyższy wynik,
- `#sum_of_scores` — zsumuje punkty,
- `#average_scores` — obliczy średnią wyników.

**Zadanie 06** Wmieszaj moduł `WithScores` do klasy `Player`. Przetestuj działanie dołączonych metod.

**Zadanie 07** Do klasy `Player` wmieszaj moduł `Comparable` (<http://ruby-doc.org/core-1.9.3/Comparable.html>) i zdefiniuj metodę `<=>`, która pozwoli porównywać graczy na podstawie sumy posiadanych punktów.

Jakie metody zostaną dodane do klasy `Player`? Przetestuj niektóre z nich:

```
1 player_1 < player_2
2 player_2.between? player_1, player_2
3 [player_1, player_2, player_3].sort
```