

# Programowanie Obiektowe

## Instrukcja do laboratorium 3

### 1 Zasady oceniania

Zadanie	Ocena
Zwierzęta (2.1)	3,0
Farma (2.2)	3,5
Student i Notatnik (2.3)	4,0
Atleta (2.4)	5,0

Warunkiem uzyskania oceny za zadanie n jest wykonanie poprzednich zadań.

W przypadku nie oddania zadania w terminie (tydzień po zajęciach), uzyskana ocena jest zmniejszana o 0,5 za każdy tydzień opóźnienia.

Plik (wkleić cały kod do jednego pliku, podpisać poszczególne zadania za pomocą komentarzy) ze зробionymi zadaniami proszę przesłać na platformie Moodle. Plik musi mieć nazwę `numeralbumu_lab1.py`. **Plik musi być plikiem tekstowym z rozszerzeniem .py.**

**UWAGA:** Termin oddania zadania jest ustawiony w systemie moodle. W przypadku nie oddania zadania w terminie, uzyskana ocena będzie zmniejszana o 0,5 za każdy zaczęty tydzień opóźnienia. Zadania oddawane później niż miesiąc po terminie ustawionym na moodle są oddawane i rozliczane w trybie indywidualnym na zajęciach lub po umówieniu się z prowadzącym.

**UWAGA:** W przypadku wysłania zadania w formie niezgodnej z opisem w instrukcji prowadzący zastrzega prawo do wystawienia oceny negatywnej za taką pracę. Przykład: wysłanie `.zip` lub `.pdf` tam, gdzie był wymagany plik tekstowy z rozszerzeniem `.py`.

### 2 Zadania do wykonania

#### 2.1 Zwierzęta

Utwórz klasę `Animal`, która będzie posiadała pola `name`, `age`, `species` oraz `weight`. Klasa ta powinna posiadać również następujące trzy metody:

1. Metodę statyczną o nazwie `oldest_animal`, która będzie przyjmować listę obiektów klasy `Animal` i zwróci nazwę i wiek najstarszego zwierzęcia na liście.
2. Metodę instancji o nazwie `is_endangered`, która będzie zwracać wartość `True` lub `False`, w zależności od tego, czy gatunek zwierzęcia jest zagrożony wyginięciem (przyjmijmy, że tylko gatunek `"tiger"` jest zagrożony wyginięciem).
3. Metodę instancji o nazwie `calculate_bmi`, która będzie zwracać wartość współczynnika BMI (Body Mass Index) dla danego zwierzęcia na podstawie jego masy ciała i wzrostu. Przyjmij, że wzrost zwierzęcia to 1 metr, a BMI obliczamy ze wzoru:  $waga / (wzrost^2)$ .

**Przykład:**

```
1 >>> lion = Animal("Simba", 5, "lion", 200)
2 >>> tiger = Animal("Shere Khan", 8, "tiger", 150)
3 >>> elephant = Animal("Dumbo", 3, "elephant", 400)
4 >>> animals = [lion, tiger, elephant]
```

```

5 >>> print(Animal.oldest_animal(animals))
6 Shere Khan is the oldest animal at 8 years old
7 >>> print(lion.is_endangered())
8 False
9 >>> print(tiger.is_endangered())
10 True
11 >>> print(lion.calculate_bmi())
12 200.0
13 >>> print(tiger.calculate_bmi())
14 150.0

```

## 2.2 Farma

Utwórz klasę `Farm`, która będzie posiadała pole `animals` - listę, w której będą przechowywane wszystkie zwierzęta na farmie. Zwierzęta, które będą dodawane do listy muszą być obiektami klasy `Animal`. Klasa ta powinna posiadać również następujące trzy metody:

1. Metodę instancji `add_animal`, która będzie dodawać nowe zwierzę na farmę.
2. Metodę instancji `feed_all`, która będzie symulować karmienie wszystkich zwierząt na karmie i zwracać informację o tym jakie jedzenie zostało im podane.
3. Metodę klasową `create_farm_with_animals`, która będzie tworzyć obiekt klasy `Farm` i dodawać do niego określoną listę zwierząt. Metoda ta powinna przyjmować jeden argument - listę zwierząt, które mają zostać dodane na farmę.

### Przykład:

```

1 >>> farm = Farm()
2 >>> cow = Animal("Berta", 5, "cow", 400)
3 >>> farm.add_animal(cow)
4 >>> chicken1 = Animal("Chirpy", 1, "chicken", 1)
5 >>> farm.add_animal(chicken1)
6 >>> chicken2 = Animal("Cluck", 2, "chicken", 1.2)
7 >>> farm.add_animal(chicken2)
8 >>> print(farm.animals)
9 [Animal(name='Berta', age=5, species="cow", weight=400),
10 Animal(name='Chirpy', age=1, species="chicken", weight=1),
11 Animal(name='Cluck', age=2, species="chicken", weight=1.2)]
12 >>> print(farm.feed_all('corn'))
13 Feeding all animals on farm with corn:
14 Berta the cow is being fed.
15 Chirpy the chicken is being fed.
16 Cluck the chicken is being fed.
17 All animals have been fed.
18 >>> animals = [
19     Animal("Berta", 5, "cow", 400),
20     Animal("Chirpy", 1, "chicken", 1),
21     Animal("Cluck", 2, "chicken", 1.2)
22 ]
23 >>> farm1 = Farm.create_farm_with_animals(animals)
24 >>> print(farm1.animals)
25 [Animal(name='Berta', age=5, species="cow", weight=400),
26 Animal(name='Chirpy', age=1, species="chicken", weight=1),
27 Animal(name='Cluck', age=2, species="chicken", weight=1.2)]

```

## 2.3 Student

Stwórz klasę `Student`, która będzie reprezentowała studenta na uczelni. Klasa powinna posiadać następujące pola: `first_name`, `last_name`, `age`, `gpa` (średnia ocen) oraz `year`. Klasa ta powinna posiadać również następujące trzy metody:

1. Metodę instancji `get_full_name`, która będzie zwracać pełne imię i nazwisko studenta
2. Metodę instancji `is_on_probation`, która zwraca wartość logiczną `True` lub `False`, w zależności od tego, czy student jest na warunkowym zawieszeniu (w zależności od średniej ocen).
3. Metodę statyczną `get_average_age`, która przyjmuje listę studentów jako argument i zwraca średnią wieku studentów na liście.
4. Metodę statyczną `get_students_by_year`, która przyjmuje listę obiektów klasy `Student` i zwraca słownik, którego kluczami są lata studiów (od 1 do 5), a wartościami są listy studentów należących do danego roku.
5. Metodę statyczną `print_students_by_year`, która wykorzystuje funkcjonalność metody `get_students_by_year` i wyświetla studentów według zadanego roku. Tak jak w przypadku metody `get_students_by_year` przyjmuje ona jako argument listę studentów. **UWAGA! Nie kopiować kodu z metody `get_students_by_year` do metody `print_students_by_year`!**

Przykład:

```
1 >>> s1 = Student("Jan", "Kowalski", 20, 2, 3.5)
2 >>> s2 = Student("Anna", "Nowak", 22, 3, 2.8)
3 >>> s3 = Student("Piotr", "Czerwinski", 19, 1, 2.1)
4 >>> s4 = Student("Katarzyna", "Wójcik", 21, 4, 4.0)
5 >>> print(s1.is_on_probation())
6 False
7 >>> print(s2.is_on_probation())
8 True
9 >>> students = [s1, s2, s3, s4]
10 >>> print(Student.get_average_age(students))
11 20.5
12 >>> students_by_year = Student.get_students_by_year(students)
13 >>> print_students_by_year(students)
14 1 rok:
15 - Piotr Czerwinski (19 lat, średnia ocen: 2.8)
16 2 rok:
17 - Jan Kowalski (20 lat, średnia ocen: 3.5)
18 3 rok:
19 - Anna Nowak (22 lat, średnia ocen: 2.8)
20 4 rok:
21 - Katarzyna Wójcik (21 lat, średnia ocen: 4.0)
22 5 rok:
23 Brak
```

## 2.4 Atlety

Stwórz klasę `Athlete`, która będzie reprezentować sportowca. Klasa ta powinna mieć pola instancji `name`, `age`, `height`, `weight`, `sport`, które będą przechowywać odpowiednio: imię i nazwisko, wiek, wzrost, wagę oraz dyscyplinę sportową. Dodatkowo, klasa `Athlete` powinna mieć dwa pola klasy: `team` oraz `country`, które będą przechowywać odpowiednio nazwę drużyny oraz nazwę kraju, do którego należy dany sportowiec. Wartość tych pól powinna być taka sama dla wszystkich instancji tej klasy.

Klasa powinna mieć metody:

1. Metodę instancji `get_bmi`, która będzie zwracać wartość BMI sportowca na podstawie jego wagi i wzrostu.

2. Metodę instancji `get_info`, która będzie zwracać informacje o sportowcu: imię i nazwisko, wiek, wzrost, wagę, dyscyplinę sportową, nazwę drużyny oraz nazwę kraju.
3. Metodę klasową `set_team(team)`, która będzie ustawiać nazwę drużyny
4. Metodę klasową `set_country(country)`, która będzie ustawiać nazwę kraju

**Przykład:**

```
1 >>> athlete1 = Athlete("Adam Nowak", 25, 175, 75, "football")
2 >>> athlete2 = Athlete("Ewa Kowalska", 30, 180, 68, "tennis")
3 >>> athlete1.set_team("Real Madrid")
4 >>> athlete2.set_country("Poland")
5 >>> print(athlete1.get_bmi())
6 24.49
7 >>> print(athlete1.get_info())
8 Name: Adam Nowak, Age: 25, Height: 175cm, Weight: 75kg, Sport: football, Team: Real Madrid,
   Country: No country
9 >>> print(athlete2.get_info())
10 Name: Ewa Kowalska, Age: 30, Height: 180cm, Weight: 68kg, Sport: tennis, Team: No team,
    Country: Poland
```