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 zrobionymi zadaniami proszę przesłać na platformie Moodle. Plik musi mieć nazwe `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:

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

```
>>> print(Animal.oldest_animal(animals))
Shere Khan is the oldest animal at 8 years old
>>> print(lion.is_endangered())
False
>>> print(tiger.is_endangered())
True
>>> print(lion.calculate_bmi())
200.0
>>> print(tiger.calculate_bmi())
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 muszę 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:

```
>>> farm = Farm()
>>> cow = Animal("Berta", 5, "cow", 400)
>>> farm.add_animal(cow)
>>> chicken1 = Animal("Chirpy", 1, "chicken", 1)
>>> farm.add_animal(chicken1)
>>> chicken2 = Animal("Cluck", 2, "chicken", 1.2)
>>> farm.add_animal(chicken2)
>>> print(farm.animals)
[Animal(name='Berta', age=5, species="cow", weight=400),
Animal(name='Chirpy', age=1, species="chicken", weight=1),
Animal(name='Cluck', age=2, species="chicken", weight=1.2)]
>>> print(farm.feed_all('corn'))
Feeding all animals on farm with corn:
Berta the cow is being fed.
Chirpy the chicken is being fed.
Cluck the chicken is being fed.
All animals have been fed.
>>> animals = [
    Animal("Berta", 5, "cow", 400),
    Animal("Chirpy", 1, "chicken", 1),
    Animal("Cluck", 2, "chicken", 1.2)
]
>>> farm1 = Farm.create_farm_with_animals(animals)
>>> print(farm1.animals)
[Animal(name='Berta', age=5, species="cow", weight=400),
Animal(name='Chirpy', age=1, species="chicken", weight=1),
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:

```
>>> s1 = Student("Jan", "Kowalski", 20, 2, 3.5)
>>> s2 = Student("Anna", "Nowak", 22, 3, 2.8)
>>> s3 = Student("Piotr", "Czerwinski", 19, 1, 2.1)
>>> s4 = Student("Katarzyna", "Wójcik", 21, 4, 4.0)
>>> print(s1.is_on_probation())
False
>>> print(s2.is_on_probation())
True
>>> students = [s1, s2, s3, s4]
>>> print(Student.get_average_age(students))
20.5
>>> students_by_year = Student.get_students_by_year(students)
>>> print_students_by_year(students)
1 rok:
- Piotr Czerwinski (19 lat, średnia ocen: 2.8)
2 rok:
- Jan Kowalski (20 lat, średnia ocen: 3.5)
3 rok:
- Anna Nowak (22 lat, średnia ocen: 2.8)
4 rok:
- Katarzyna Wójcik (21 lat, średnia ocen: 4.0)
5 rok:
Brak
```

2.4 Atleta

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:

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