Programowanie Obiektowe
Instrukcja do laboratorium 4

1 Zasady oceniania

Zadanie	Ocena
Zwierzęta (2.1)	3,0
Zegar (2.2)	3,5
Książka (2.3)	4,0
Meble (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 dwie klasy: jedna klasa bazowa oraz dwie klasy dziedziczące po niej. Klasę bazową nazwij *Ssak*, a dwie kolejne dowolnymi zwierzętami które są ssakami. Klasa bazowa powinna posiadać pole do definiowanego obiektu o nazwie `info` oraz metodę o nazwie `ciekawostka`, gdzie `info` jest polem publicznym, a `ciekawostka` metodą publiczną wyświetlającą pole `info`. Wszystkie klasy powinny posiadać zmienną (pole klasy) o nazwie `rodzaj`. W konstruktorze klasy *Ssak* powinna być wyświetlana informacja o rodzaju obiektu. Do dziedziczonego konstruktora klasy bazowej powinien być przekazywany parametr `info` z klasy dziedziczonej. Poniżej znajduje się przykład utworzenia dwóch obiektów wraz z wywoływaniem dla nich metody `ciekawostka`.

Listing 1: Przykład

```
>>> s = Ssak()
Stworzyłeś: Ssak
>>> s.ciekawostka()
Brak ciekawostki
>>> p = Pies()
Stworzyłeś: Pies
>>> p.ciekawostka()
Ma cztery łapy
```

2.2 Zegar

Napisz program, który definiuje trzy klasy tj. `Zegar`, `ZegarElektroniczny` oraz `ZegarCzwartyWymiar`, gdzie:

- `Zegar` ma pola: `godzina, minuta, sekunda`. Posiada także metodę `ustaw_czas()`, która pozwala na ustawienie godziny, minuty i sekundy.
- `ZegarElektroniczny` dziedziczy po klasie `Zegar` i dodaje pola: `dni_tygodnia`, `dzien_miesiaca`, `miesiac`, `rok`. Metoda `ustaw_czas()` zostaje nadpisana w celu uwzględnienia nowych pól.
- `ZegarCzwartyWymiar` dziedziczy po klasie `ZegarElektroniczny` i dodaje pole `czas_kwantowy`. Metoda `ustaw_czas()` zostaje ponownie nadpisana.

Listing 2: Przykład

```
>>> zegar = Zegar()
>>> zegar.ustaw_czas(12, 30, 0)
>>> print("Zegar:", zegar)
Zegar: 12:30:00
>>> zegar_elektroniczny = ZegarElektroniczny()
>>> zegar_elektroniczny.ustaw_czas(12, 30, 0, 2, 22, 3, 2023)
>>> print("ZegarElektroniczny:", zegar_elektroniczny)
ZegarElektroniczny: 12:30:00, 22-03-2023
>>> zegar_czwarty_wymiar = ZegarCzwartyWymiar()
>>> zegar_czwarty_wymiar.ustaw_czas(12, 30, 0, 2, 22, 3, 2023, 0.5)
>>> print("ZegarCzwartyWymiar:", zegar_czwarty_wymiar)
ZegarCzwartyWymiar: 12:30:00, 22-03-2023, 0.5
```

2.3 Książka

Utwórz klasę Ksiazka, reprezentującą książkę, posiadającą pola takie jak `tytul, autor` oraz `cena`. Klasa powinna posiadać metody magiczne `__repr__` oraz `__str__`. A następnie stwórz:

- Klasę pochodną po klasie `Ksiazka` o nazwie `KsiazkaFantasy`, która będzie posiadała dodatkowe pole o nazwie `podgatunek_fantasy`.
- Klasę pochodną po klasie `Ksiazka` o nazwie KsiazkaKryminalna, która będzie posiadała dodatkowe pole o nazwie `liczba_zabojst`.
- Klasę o nazwie `KsiazkaFantastycznoKryminalna`, która będzie dziedziczyć zarówno po klasie `KsiazkaFantasy` oraz `KsiazkaKryminalna`.
- Klasę o nazwie `Biblioteka`, która będzie posiadała listę dostępnych książek oraz słownik wypożyczonych książek przez zadane osoby. Dodatkowo klasa musi posiadać metody `lista_ksiazek`, `dodaj_ksiazke`, `wypozycz` oraz `zwroc`. Funkcjonalność tych metod jest następująca:
  - `lista_ksiazek` - wyświetla dostępne książki w bibliotece (należy pamiętać o niewyświetlaniu książek wypożyczonych). W przypadku braku książek wyświetla że brakuje książek w bibliotece.
  - `dodaj_ksiazke` - dodaje zadaną książkę do listy książek
  - `wypozycz` - dodaje do słownika o kluczu książka zadaną osobę. Posiada takie argumenty jak `ksiazka` oraz `osoba`. Należy zaimplementować funkcję w taki sposób aby wyświetlała czy książka jest dostępna w bibliotece bądź czy jest wypożyczona przez inną osobę.
  - `zwroc` - usunie zadaną książkę ze słownika wypożyczonych książek. W innym przypadku wyświetli że książka nie jest wypożyczona.

UWAGA! Zaopatrz klasy pochodne w obsługę metod magicznych związanych z wyświetlaniem.

Listing 3: Przykład

```
# Tworzenie kilku książek
>>> ksiazka1 = Ksiazka("Dune", "Frank Herbert", 39.99)
>>> ksiazka2 = KsiazkaFantasy("Władca Pierścieni", "J.R.R. Tolkien", 49.99, "high fantasy")
>>> ksiazka3 = KsiazkaKryminalna("Zbrodnia i kara", "Fiodor Dostojewski", 29.99, 10)
>>> ksiazka4 = KsiazkaFantastycznoKryminalna("Miasto Cienia", "Cassandra Clare", 59.99,
    "urban fantasy", 5)


# Tworzenie biblioteki i dodawanie książek
>>> biblioteka = Biblioteka()
>>> biblioteka.dodaj_ksiazke(ksiazka1)
>>> biblioteka.dodaj_ksiazke(ksiazka2)
>>> biblioteka.dodaj_ksiazke(ksiazka3)
>>> biblioteka.dodaj_ksiazke(ksiazka4)

# Wyświetlanie listy książek
>>> biblioteka.wyswietl_ksiazki()
Dune, autor: Frank Herbert, cena: 39.99
Władca Pierścieni, autor: J.R.R. Tolkien, cena: 49.99, podgatunek fantasy: high fantasy
Zbrodnia i kara, autor: Fiodor Dostojewski, cena: 29.99, liczba zabójstw: 10
Miasto Cienia, autor: Cassandra Clare, cena: 59.99, podgatunek fantasy: urban fantasy,
    liczba zabójstw: 5


# Wypożyczanie książek
>>> biblioteka.wypozycz_ksiazke(ksiazka2, "Jan Kowalski")
Wypożyczono książkę Władca Pierścieni osobie Jan Kowalski.
>>> biblioteka.wypozycz_ksiazke(ksiazka4, "Adam Nowak")
Wypożyczono książkę Miasto Cienia osobie Adam Nowak.

# Wyświetlanie listy wypożyczonych książek
>>> biblioteka.wyswietl_wypozyczone()
Wypożyczone książki:
Władca Pierścieni, wypożyczona przez: Jan Kowalski
Miasto Cienia, wypożyczona przez: Adam Nowak

# Zwracanie książek
>>> biblioteka.zwroc_ksiazke(ksiazka2)
Książka Władca Pierścieni została zwrócona.
>>> biblioteka.zwroc_ksiazke(ksiazka4)
Książka Miasto Cienia została zwrócona.

# Wyświetlanie listy książek i wypożyczonych książek po zwrocie
>>> biblioteka.wyswietl_ksiazki()
Dune, autor: Frank Herbert, cena: 39.99
Władca Pierścieni, autor: J.R.R
>>> biblioteka.wyswietl_wypozyczone()
```

2.4 Meble

Stwórz klasę `Furniture`, która będzie miała pola `__material` (pole prywatne) i `size` (pole publiczne), oraz metody `get_material()` i `set_material()`.

Następnie stwórz klasę `Table` dziedziczącą po klasie `Furniture` oraz dodaj do niej pole `__legs` (pole prywatne) i dekorator `@property` dla pola `__legs`. Dodaj metodę `set_legs()` pozwalającą na zmianę liczby nóg.

Stwórz klasę `Chair` dziedziczącą po klasie `Furniture` oraz dodaj do niej pole `__has_armrests` (pole prywatne) i dekorator `@property` dla pola `__has_armrests`. Dodaj metodę `set_armrests()` pozwalającą na zmianę wartości pola `__has_armrests`.

Listing 4: Przykład

```
>>> table = Table('wood', 'large', 4)
>>> chair = Chair('metal', 'medium', True)

>>> print('Table material:', table.get_material())
Table material: wood
>>> table.set_material('plastic')
>>> print('Table material after modification:', table.get_material())
Table material after modification: plastic
>>> print('Table size:', table.size)
Table size: large
>>> print('Table legs:', table.legs)
Table legs: 4

>>> table.set_legs(6)
>>> print('Table legs after modification:', table.legs)
Table legs after modification: 6

>>> print('Chair material:', chair.get_material())
Chair material: metal
>>> print('Chair size:', chair.size)
Chair size: medium
>>> print('Chair has armrests:', chair.has_armrests)
Chair has armrests: True

>>> chair.set_armrests(False)
>>> print('Chair has armrests after modification:', chair.has_armrests)
Chair has armrests after modification: False
```