

## Lista zadań nr 4

**Zadanie 1 (2 pkt)** Zdefiniuj dwie klasy:

Klasa `Pupil` zawierająca:

- konstruktor z dwoma parametrami `imię` i `nazwisko` - wykorzystaj właściwości do kontroli poprawności atrybutów `name` i `surname` - ciągi tekstowe składające się z co najmniej 3 liter; konstruktor powinien także definiować atrybut `marks` - słownik przechowujący oceny (kluczem nazwa przedmiotu a wartością ocena - liczba rzeczywiste ze zbioru  $\{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6\}$ );
- `complete_marks()` - dodaje przedmioty oraz oceny do słownika i kontroluje ich poprawność (słownik uzupełnia użytkownik);
- `print_marks()` - wyświetla przedmioty i oceny;
- metoda `mean()` - zwraca średnią z ocen,
- `__str__()` - zwraca napis składający się z imienia i nazwiska oraz średniej ocen.

Klasa `Student` dziedzicząca z klasy `Pupil`:

- zawiera konstruktor z parametrami wywołujący konstruktor klasy nadrzędnej z dodatkowym atrybutem `weights` - słownik z takimi samymi kluczami jak `marks` i wartościami oznaczającymi wagi (liczby rzeczywiste z przedziału  $(0, 1]$ )
- `complete_weights()` - przypisuje wagę dla każdego przedmiotu i kontroluje jej poprawność (słownik uzupełnia użytkownik);
- zawiera przesłoniętą metodę `mean()`, która ma liczyć średnią ważoną.
- zawiera metodę `__str__()` - zwraca napis składający się z imienia i nazwiska oraz średniej ocen - wykorzystaj metodę klasy bazowej.

Następnie utwórz jeden obiekt klasy `Pupil` i jeden klasy `Student` z takimi samymi ocenami i sprawdź, jak są liczone ich średnie.

**Zadanie 2 (2 pkt)** Opracuj abstrakcyjną klasę bazową o nazwie `Temperature`, która przechowuje pojedynczą wartość temperatury. Klasa powinna mieć następujący nagłówek metody `__init__()` (rzeczywiście zaimplementowana metoda abstrakcyjna):

```
def __init__(self, temperature)
```

Oprócz tego abstrakcyjna klasa `Temperature` powinna zawierać następujące metody:

- `__str__()` - powinien zwrócić ciąg postaci "75 stopni w skali Celsjusz" (metoda konkretna);
- `__repr__()` - powinien zwrócić ciąg postaci "ClassName(temperature)" (metoda konkretna);
- `above_freezing()` - zwraca `True` jeśli temperatura powyżej punktu zamarzania (metoda konkretna);
- `convert_to_Fahrenheit()` - zwraca nowy obiekt temperatury przekształcony na stopnie Fahrenheita (metoda abstrakcyjna);
- `convert_to_Celsius()` - zwraca nowy obiekt temperatury przekształcony na stopnie Celsjusza (metoda abstrakcyjna);
- `convert_to_Kelvin()` - zwraca nowy obiekt temperatury przekształcony na stopnie Kelvina (metoda abstrakcyjna);
- właściwości `temperature` typu *setter* i *getter* (abstrakcyjne właściwości).

Opracuj podklasy `Fahrenheit`, `Celsius` i `Kelvin` i odpowiednio wdróż każdą z metody abstrakcyjnej klasy `Temperature`. (Należy pamiętać, że gdy stosowana jest bezsensowna metoda konwersji, np. wywołanie metody, `temp1.convert_to_Fahrenheit()`, gdzie `temp1` jest instancją klasy `Fahrenheit`, powinno zwracać ten sam obiekt temperatury. Sprawdź poprawność swoich klas w następujący sposób:

- stwórz listę zawierającą 12 instancji klas (każdej po cztery egzemplarze) `Kelvin`, `Celsius` i `Fahrenheit`;
- wydrukuj obiekty listy, a dla temperatur które są powyżej temperatury zamarzania wody dodaj adnotację "powyżej zera";
- utwórz trzy listy zawierające każdą temperaturę z pierwotnej listy przekształconą do wspólnej skali temperatur (Fahrenheita, Celsjusza, lub Kelvina);
- z każdej z utworzonych list wydrukuj tylko te, które są poniżej temperatury zamarzania wody.

Oto potrzebne przeliczniki:

`Celsjusz = 0.556 * (Fahrenheit - 32.0)`

`Kelwin = Celsjusz + 273.16`

**Zadanie 3 (2 pkt)** Zaprojektuj klasę `SortedList` - przechowującą listę elementów posortowanych w kolejności. Klasa powinna być oparta o abstrakcyjną klasę bazową `Sequence` z modułu `collections.abc`. Klasa powinna tworzyć posortowaną listę elementów w oparciu o opcjonalny klucz sortowania. Stworzona klasa powinna mieć interfejs zbliżony do wybudowanego typu `list` - (bez metod `insert()`, `reverse()` oraz `sort()`). Zamiast metody `append()` zastosuj metodę `add()` wstawiającą dany element w odpowiednie miejsce - w tym celu wykorzystaj metodę chronioną `_find_index()`, która zwraca odpowiednią pozycję indeksu (zastosuj algorytm wyszukiwania binarnego)).