Marzena Nowakowska, Wydział Zarządzania i Modelowania Komputerowego, PŚk

Temat 6. Elementy programowania obiektowego

Zad. 1.
Napisać program *Zad06_01* o treści, jak podano poniżej. Uruchomić program i przeanalizować jego działanie.

```
# Uwaga: w programie nie ma kontroli poprawności danych.

class wycieczki():
    nazwaBiura="Holiday travel"

    def __init__(self, nazwa, kraj, cena, maxUczest, termin):    # konstruktor
        # Parametry: nazwa wycieczki, zwiedzany kraj, cena, maksymalna liczba uczestników
        self.nazwaW = nazwa
        self.nazwaK = kraj
        self.cenaW = cena
        self.terminW = termin
        self.maxLOsb = maxUczest
        self.lUczest=0
        self.uczestnicy=[]

    def oferta(self):
        print("Wycieczka " + self.nazwaW +" organizowana przez", self.nazwaBiura)
        print("Kraj docelowy: ", self.nazwaK)
        print("Cena za osobę: ", self.cenaW)
        print("termin wycieczki: ", self.terminW)

    def zapisy(self, osoba):
        if self.lUczest <self.maxLOsb:
            self.lUczest+=1
            self.uczestnicy.append(osoba)
            print("Osoba",osoba,"wpisana na listę")
             # jak dostać się do ostatniej osoby poprzez właściwość obiektu?
        else:
            print("Niestety lista osób jest wyczerpana")

    def listaUcz(self):
        i=0
        for el in self.uczestnicy:
            i+=1
            print(i,"--",el)

    def przychod(self):
        return self.lUczest*self.cenaW

wycieczka1=wycieczki("'Dawna Jugosławia'", "Słowenia", 2500, 26, "2022/10/30-2022/11/07")
wycieczka1.oferta()
wycieczka1.zapisy("Czerwony Kapturek")
wycieczka1.zapisy("Zły wilk")
wycieczka1.zapisy("Królewna Śnieżka")
wycieczka1.zapisy("Książę wybawca")
wycieczka1.zapisy(input("Podaj imię i nazwisko: "))
print("Lista uczestników wycieczki")
print("============================")
```

```
wycieczka1.listaUcz()
print ("Przychód z wycieczki:",wycieczka1.przychod())
print("Liczba niewykorzystanych miejsc:", wycieczka1.maxLOsb-wycieczka1.lUczest) # ostatnia instrukcja
```

Zmodyfikować ww. program w sposób opisany poniżej.

- Przy wyświetlaniu listy uczestników ma być podana nazwa wycieczki, a sama lista ma być uporządkowana alfabetycznie.
- Zdefiniować drugi obiekt *wycieczka2* z własnymi danymi. W tym celu zdefiniować obiekt typu *list* o nazwie *mojeDane*, w którym umieścić dane uczestników drugiej wycieczki, tak aby liczba elementów listy była większa niż maksymalna liczba uczestników w obiekcie *wycieczka2*. Przetestować działanie instrukcji warunkowej w metodzie *zapisy*, wykorzystując ww. listę do zwartościowania odpowiednich danych obiektu *wycieczka2*.
- Wyświetlić na ekranie listę uczestników obiektu *wycieczka2* oraz przychód z tej wycieczki. Wyświetlić komunikat, która wycieczka przyniosła większy przychód.
- W klasie *wycieczki* zdefiniować metodę *wakaty*, która wyświetla liczbę niewykorzystanych miejsc (w szczególności 0) oraz informację, czy można jeszcze kupić wycieczkę. Przyjmuje się, że ostatni dzień wykupu to dzień poprzedzający dzień wyjazdu.
  Do sprawdzenia dat należy skorzystać z operacji na datach, przyłączając klasy *datetime* i *date* z modułu *datetime*. Sprawdzić w helpie lub Internecie (np. *https://www.dataquest.io/blog/python-datetime-tutorial/*). W miejsce ostatniej instrukcji programu wywołać metodę, tak aby wyświetlała właściwy komunikat. Przetestować działanie metody.
  Przykładowe wyniki testowania;

```
'Dawna Jugosławia' wakaty:  22 pozostało dni: 0
'Meteory' wakaty:  0 pozostało dni: 132
'Śladami Odrodzenia' wakaty:  15 pozostało dni: 163 <-- można jeszcze kupić wycieczkę
'Drakula' wakaty:  0 pozostało dni: 0
```

Zad. 2.

Utworzyć w pliku *Zad06_02* kopię programu *Zad06_01*. Pozostawić w programie głównym definicje obiektów *wycieczka1* i *wycieczka2* wraz z ich danymi. Zmodyfikować program tak, aby obiekty definiujące wycieczki i ich uczestników były elementami listy *wycieczkiBiura* (powstaje lista obiektów). Dopisać do listy dwie wycieczki. Zmodyfikować dane tak, aby uzyskać wyniki testowania metody *wakaty* jak w programie *Zad06_01* dla kolejnych elementów listy *wycieczkiBiura*; napisać pętlę, która dla każdego obiektu listy *wycieczkiBiura* wyświetla informację o wakatach.

Zad. 3.

W programie *Zad06_03* zdefiniować klasę *Film*. Klasa ma zawierać następujące składowe charakteryzujące film: tytuł, reżyser, kraj produkcji, czas trwania projekcji i opcjonalnie opis (parametr domyślny o wartości "" (pusty łańcuch)). Konstruktor klasy przekazuje parametry dla danych składowych instancji klasy. Metoda *opisFilmu* wyświetla komunikat "Brak opisu", jeżeli właściwa składowa jest pustym łańcuchem. W przeciwnym przypadku jest wyświetlany opis filmu. Przetestować działanie tak skonstruowanej klasy.

Zad. 4.

W programie *Zad06_03* zdefiniować dwie klasy pochodne: *FilmFblr* (dla filmu fabularnego) oraz *filmAnmw* (dla filmu animowanego).

- Klasa *FilmFblr* oprócz odziedziczonych elementów z klasy bazowej zawiera atrybuty instancji: *aktorzy* typu *list* zawierający wykaz aktorów grających najważniejsze role oraz *kategoria* opisujący kategorię filmu (np. fantazy, sf, horror, komedia itp.).
- Klasa *FilmAmnw* oprócz odziedziczonych elementów z klasy bazowej zawiera atrybut instancji klasy *tecAmn* zawierającay opis techniki animacji (rysunek, lalki, grafika komputerowa, ruchome malarstwo, non-camera).

Uzupełnić klasy pochodne o definicje konstruktorów oraz funkcje wyświetlające opis filmu. Obie klasy można dodatkowo zmodyfikować wg uznania.