Vežbe 5 - Zadaci

Zadatak 1. Realizovati klasu Zivotinja sa atributima *ime*, *vrsta*, *godine* i *zvuk*. Takođe, dodati metode *oglasi_se* koja ispisuje prosleđeni zvuk, *opis* koja daje osnovne informacije o zivotinji, *povecaj_starost* koja ostari zivotinju za prosledjeni broj godina i *promeni_ime* koja menja ime zivotinji. Instancirati objekat i testirati sve metode

```
class Zivotinja:
    def __init__(self, ime, vrsta, godine, zvuk):
        self.ime = ime
        self.vrsta = vrsta
        self.godine = godine
        self.zvuk = zvuk

    def oglasi_se(self):
        print(f"{self.ime} kaze {self.zvuk}")

    def opis(self):
        return f"{self.ime} je {self.vrsta} i ima {self.godine} godina."

    def povecaj_starost(self, godine=1):
        self.godine += godine

    def promeni_ime(self, novo_ime):
        self.ime = novo_ime

zivotinja = Zivotinja("Rex", "pas", 3, "vau")

zivotinja.oglasi_se()
zivotinja.povecaj_starost()
print(zivotinja.opis())
zivotinja.promeni_ime("Alfi")
print(zivotinja.opis())

macka = Zivotinja("Bleki", "macka", 2, "mijau")
macka.promeni_ime()
```

Zadatak 2. a) Realizovati klasu Osoba sa privatnim atributima *ime* i *godine*. Dodati metode *postavi_ime*, *postavi_godine* i *prikazi_podatke*, čime bi se modelovala enkapsulacija.

```
class Osoba:
    def __init__(self, ime, godine):
        self.__ime = ime    # privatni atribut
        self.__godine = godine    # privatni atribut

    def prikazi_podatke(self):
        print(f"Ime: {self.__ime}, Godine: {self.__godine}")

    # Metode za postavljanje vrednosti privatnih atributa
```

```
    def postavi_ime(self, novo_ime):
        self.__ime = novo_ime

    def postavi_godine(self, nove_godine):
        self.__godine = nove_godine

# Kreiranje nove instance klase Osoba
osoba = Osoba("Mika", 30)

# Pozivanje metoda za prikazivanje podataka o osobi
osoba.prikazi_podatke()
#ispisuje "Ime: Mika, Godine: 30"

# Pokusaj direktnog pristupa i promene privatnog atributa
osoba.__ime = "Pera"
osoba.prikazi_podatke()
#ispisuje "Ime: Mika, Godine: 30"

# Promena privatnog atributa pozivom metode
osoba.postavi_ime("Pera")
osoba.prikazi_podatke()
#ispisuje "Ime: Pera, Godine: 30"
```

Zadatak 2. b) Prepraviti prethodni primer tako da se koriste svojstva.

```
class Osoba:
    def __init__(self, ime, godine):
        self.__ime = ime
        self.__godine = godine

    def prikazi_podatke(self):
        print(f"Ime: {self.__ime}, Godine: {self.__godine}")

    # Svojstva
    @property
    def ime(self):
        return self.__ime

    @property
    def godine(self):
        return self.__godine

    @ime.setter
    def ime(self, novo_ime):
        self.__ime = novo_ime

    @godine.setter
    def godine(self, nove_godine):
        self.__godine = nove_godine
```

```python
# Kreiranje nove instance klase Osoba
osoba = Osoba("Mika", 30)

# Izmena podataka direktno koriscenjem setter metode nad svojstvom
osoba.ime = "Pera"
print(f"Ime: {osoba.ime}, Godine: {osoba.godine}")
```

Zadatak 2. c) Prepraviti prethodni primer tako da se dodaju statički atribut i statička metoda. Njihov zadatak je da broje koliko je osoba napravljeno i da prikaže trenutni broj osoba.

```python
class Osoba:
    # Staticki atribut
    broj_osoba = 0

    def __init__(self, ime, godine):
        self.__ime = ime
        self.__godine = godine
        # Uvecavanje broja osoba svaki put kada se kreira nova instanca
        Osoba.broj_osoba += 1

    def prikazi_podatke(self):
        print(f"Ime: {self.__ime}, Godine: {self.__godine}")

    @property
    def ime(self):
        return self.__ime

    @property
    def godine(self):
        return self.__godine

    @ime.setter
    def ime(self, novo_ime):
        self.__ime = novo_ime

    @godine.setter
    def godine(self, nove_godine):
        self.__godine = nove_godine

    # Staticka metoda
    @staticmethod
    def prikazi_broj_osoba():
        print(f"Trenutno ima {Osoba.broj_osoba} osoba")

# Kreiranje nekoliko instanci klase Osoba
osoba1 = Osoba("Ana", 30)
osoba2 = Osoba("Pera", 25)
osoba3 = Osoba("Mika", 34)
osoba4 = Osoba("Jana", 22)
```

```
osoba5 = Osoba("Marko", 40)

# Pozivanje staticke metode
Osoba.prikazi_broj_osoba()    # ispisuje "Trenutno ima 2 osoba"
```

Zadatak 3. Realizovati klasu KompleksanBroj koja ima realnu i imaginarnu vrednost, reimplementirati magične metode za prikaz broja, za sabiranje, oduzimanje, deljenje i množenje, kao i računanje modula kompleksnog broja.

```
class KompleksanBroj:
    def __init__(self, realni_deo, imaginarni_deo):
        self.realni_deo = realni_deo
        self.imaginarni_deo = imaginarni_deo

    def __str__(self):
        return f"{self.realni_deo} + {self.imaginarni_deo}j"

    def __add__(self, other):
        return KompleksanBroj(self.realni_deo + other.realni_deo,
                              self.imaginarni_deo + other.imaginarni_deo)

    def __sub__(self, other):
        return KompleksanBroj(self.realni_deo - other.realni_deo,
                              self.imaginarni_deo - other.imaginarni_deo)

    def __mul__(self, other):
        return KompleksanBroj(self.realni_deo * other.realni_deo -
                              self.imaginarni_deo * other.imaginarni_deo,
                              self.realni_deo * other.imaginarni_deo +
                              self.imaginarni_deo * other.realni_deo)

    def __truediv__(self, other):
        try:
            realni_deo = (self.realni_deo * other.realni_deo +
                          self.imaginarni_deo * other.imaginarni_deo) /
                         (other.realni_deo**2 + other.imaginarni_deo**2)

            imaginarni_deo = (self.imaginarni_deo * other.realni_deo -
                              self.realni_deo * other.imaginarni_deo) /
                              (other.realni_deo**2 + other.imaginarni_deo**2)
            return KompleksanBroj(realni_deo, imaginarni_deo)
        except ZeroDivisionError:
            print("Greska: Pokusaj deljenja sa nulom!")

    def __abs__(self):
        return (self.realni_deo**2 + self.imaginarni_deo**2) ** 0.5


```

```python
# kreiranje dva kompleksna broja
z1 = KompleksanBroj(2, 3)
z2 = KompleksanBroj(4, -1)

# ispis dva kompleksna broja
print(z1)  # 2 + 3j
print(z2)  # 4 - 1j

# sabiranje dva kompleksna broja
z3 = z1 + z2
print(z3)  # 6 + 2j

# oduzimanje dva kompleksna broja
z4 = z1 - z2
print(z4)  # -2 + 4j

# mnozenje dva kompleksna broja
z5 = z1 * z2
print(z5)  # 11 + 10j

# deljenje dva kompleksna broja
z6 = z1 / z2
print(z6)  # 0.44 + 0.78j

# racunanje modula kompleksnog broja
z7 = KompleksanBroj(-2, 5)
abs_z7 = abs(z7)
print(abs_z7)  # 5.385164807134504
```

Zadatak 4. Realizovati klasu Motor i klasu Automobil. Klasa Motor sadrži atribut *snaga* koji definiše snagu motora i metoru *ubrzaj* koja menja brzinu na zadatu. Klasa automobil ima privatne atribute *marka*, *model* i *motor*. Privatnim statičkim atributom *broj_proizvedenih* beleži se broj proizvedenih automobila. Generisati magičnu metodu za ispis podataka o automobilu, kao i svihih svojstava za prikaz elementa.

```python
class Motor:
    def __init__(self, snaga, brzina=0):
        self.snaga = snaga
        self.brzina = brzina

    def ubrzaj(self, brzina):
        print(f"Motor menja brzinu sa {self.brzina} km/h na {brzina} km/h")
        self.brzina = brzina

class Automobil:
    __broj_proizvedenih = 0

    def __init__(self, marka, model, snaga_motora):
```

```
14          self.__marka = marka
15          self.__model = model
16          self.__motor = Motor(snaga_motora)
17          Automobil.__broj_proizvedenih += 1
18
19      def __str__(self):
20          return f"{self.__marka} {self.__model}"
21
22
23      def ubrzaj(self, brzina):
24          self.__motor.ubrzaj(brzina)
25
26      @staticmethod
27      def broj_proizvedenih_automobila():
28          return Automobil.__broj_proizvedenih
29
30      @property
31      def marka(self):
32          return self.__marka
33
34      @property
35      def model(self):
36          return self.__model
37
38      @property
39      def snaga_motora(self):
40          return self.__motor.snaga
41
42  # Primer upotrebe
43  a1 = Automobil("Audi", "A4", 150)
44  a2 = Automobil("BMW", "320d", 184)
45
46  print(a1)
47  print(a2)
48
49  a1.ubrzaj(50)
50  a1.ubrzaj(80)
51  a2.ubrzaj(70)
52  a2.ubrzaj(90)
53
54  print(Automobil.broj_proizvedenih_automobila())
55
56  print(a1.snaga_motora)
```