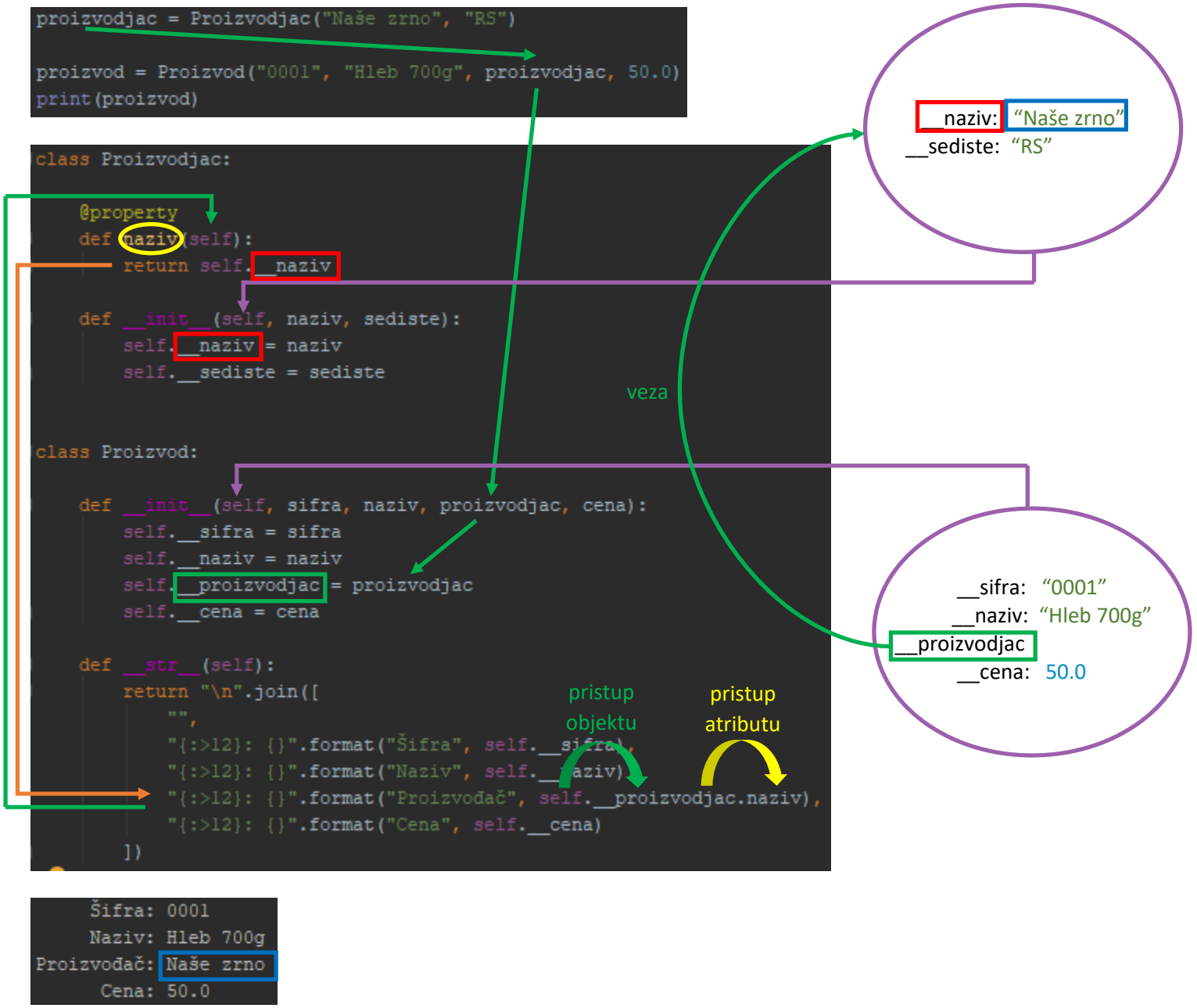
Vežba 5

Atributi objekata neke klase ne moraju biti primitivnog tipa (*int*, *float*, *string* i sl.), već mogu biti i reference na druge objekte, ili čak kolekcije referenci na druge objekte. Tada se kaže da postoje **veze** između klasa, odnosno objekata.

Na slici 1 se može videti primer veze asocijacije između dve klase. Ovaj konkretan primer implementira vezu više-na-jedan (*many-to-one*). Za svaki objekat klase *Proizvod* bi trebalo da bude vezan tačno jedan objekat klase *Proizvodjac*.



Slika 1. Povezivanje objekata

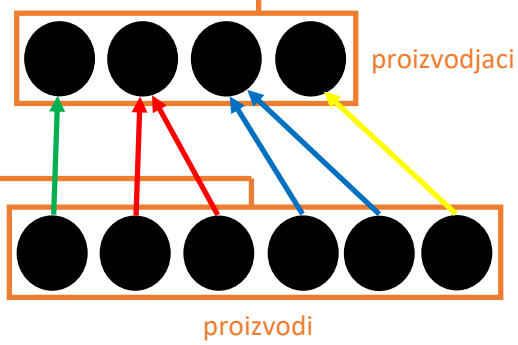
Ako se veza posmatra u suprotnom smeru, moguće je povezati isti objekat klase *Proizvodjac* za više objekata klase *Proizvod* (slika 2). Sa stanovišta proizvođača, veza se tretria kao jedan-na-više (*one-to-many*).

```python
class Proizvodjac:

    @property
    def naziv(self):
        return self.__naziv

    def __init__(self, naziv, sediste):
        self.__naziv = naziv
        self.__sediste = sediste


class Proizvod:

    format_linije = "{:5} {:20} {:12} {:9} {:>12} {:>12}"

    def __init__(self, sifra, naziv, proizvodjac, cena):
        self.__sifra = sifra
        self.__naziv = naziv
        self.__proizvodjac = proizvodjac
        self.__cena = cena

    def linija_tabele(self):
        return self.format_linije.format(self.__sifra, self.__naziv, self.__proizvodjac.naziv, self.__cena, "", "", "")

    @classmethod
    def prikazi_proizvode(cls, proizvodi):
        print()
        # zaglavlje
        print(cls.format_linije.format("Šifra", "Naziv", "Proizvođač", "Cena", "Rok trajanja", "Garancija"))
        print("{} {} {} {} {} {}".format("-"*5, "-"*20, "-"*12, "-"*9, "-"*12, "-"*12))
        # sadržaj
        for proizvod in proizvodi:
            print(proizvod.linija_tabele())
```

pristup objektu

pristup atributu

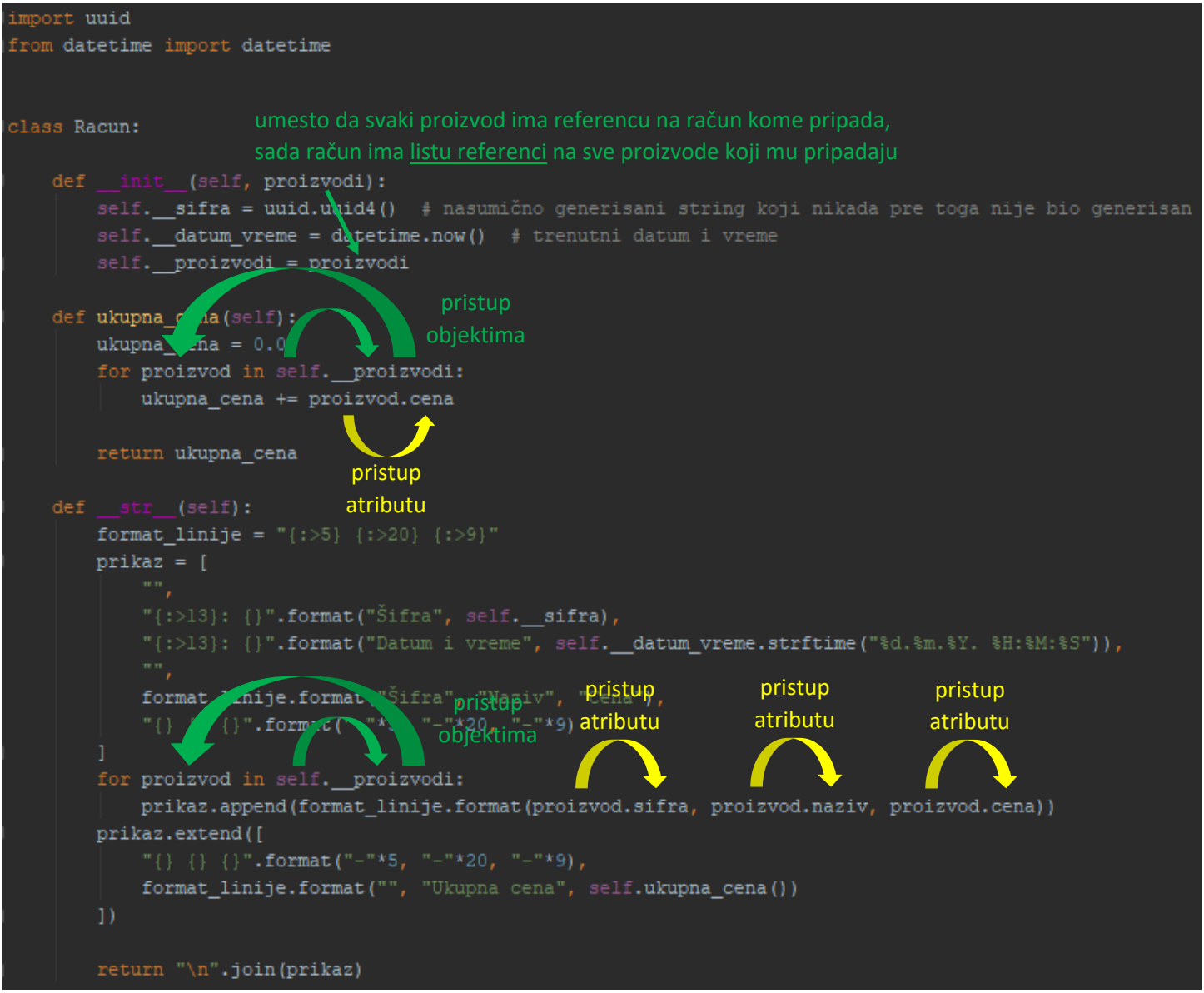
```python
proizvodjaci = [
    Proizvodjac("Naše zrno", "RS"),
    Proizvodjac("Imlek", "RS"),
    Proizvodjac("Samsung", "KR"),
    Proizvodjac("Apple", "US"),
]

proizvodi = [
    PrehrambeniProizvod("0001", "Hleb 700g", proizvodjaci[0], 50.0, 7),
    PrehrambeniProizvod("0002", "Mleko 1l", proizvodjaci[1], 80.0, 180),
    PrehrambeniProizvod("0003", "Jogurt 1.5l", proizvodjaci[1], 130.0, 14),
    TehnickiProizvod("0004", "Samsung HDTV 42", proizvodjaci[2], 50000.0, 24),
    TehnickiProizvod("0005", "Samsung Galaxy S10", proizvodjaci[2], 95000.0, 24),
    TehnickiProizvod("0006", "iPhone XS", proizvodjaci[3], 110000.0, 24)
]
Proizvod.prikazi_proizvode(proizvodi)
```



Šifra	Naziv	Proizvođač	Cena	Rok trajanja	Garancija
0001	Hleb 700g	Naše zrno	50.0	7 dana	
0002	Mleko 1l	Imlek	80.0	180 dana	
0003	Jogurt 1.5l	Imlek	130.0	14 dana	
0004	Samsung HDTV 42	Samsung	50000.0		24 meseca
0005	Samsung Galaxy S10	Samsung	95000.0		24 meseca
0006	iPhone XS	Apple	110000.0		24 meseca

Slika 2. Povezivanje istog objekta za više drugih objekata

U prethodnom primeru objekti na “one” strani veze (proizvođači) su bili referencirani u objektima na “many” strani veze (proizvodi). Tada je klasa *Proizvod* implementirala jedan atribut kojim referencira klasu *Proizvodjac*. Moguće je vezu implementirati i na suprotan način. Tada se u klasi na “one” strani veze implementira kolekcija (tipično lista) referenci na objekte klase na “many” strani veze. Na slikama 3.1 i 3.2 se može videti takav primer.

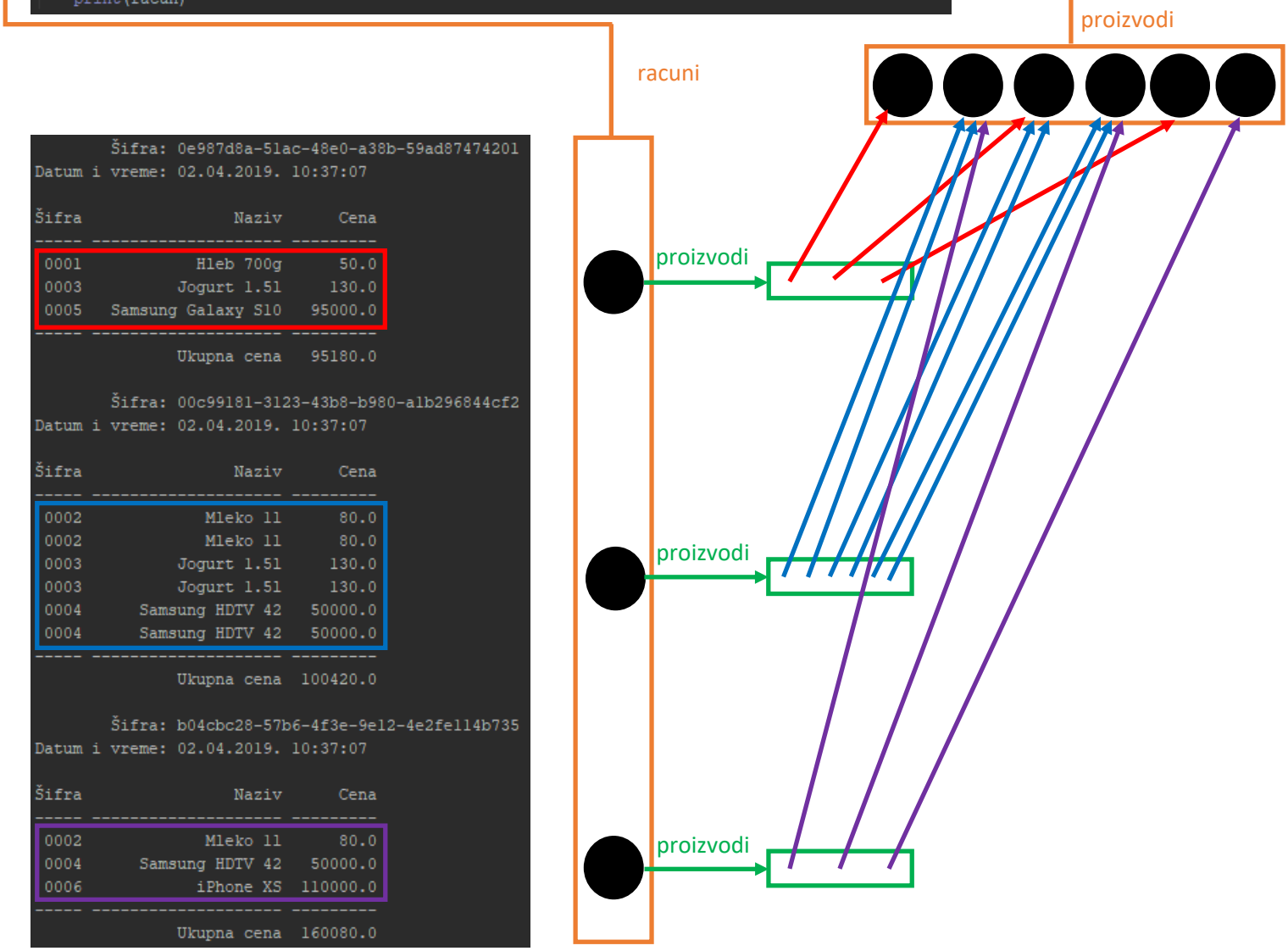



```
import uuid
from datetime import datetime


class Racun:
    def __init__(self, proizvodi):
        self.__sifra = uuid.uuid4()  # nasumično generisani string koji nikada pre toga nije bio generisan
        self.__datum_vreme = datetime.now()  # trenutni datum i vreme
        self.__proizvodi = proizvodi

    def ukupna_cena(self):
        ukupna_cena = 0.0
        for proizvod in self.__proizvodi:
            ukupna_cena += proizvod.cena

        return ukupna_cena

    def __str__(self):
        format_linije = "{:>5} {:>20} {:>9}"
        prikaz = [
            "",
            "{:>13}: {}".format("Šifra", self.__sifra),
            "{:>13}: {}".format("Datum i vreme", self.__datum_vreme.strftime("%d.%m.%Y. %H:%M:%S")),
            "",
            format_linije.format("Šifra", "Naziv", "Cena"),
            "{} {} {}".format("-"*5, "-"*20, "-"*9)
        ]
        for proizvod in self.__proizvodi:
            prikaz.append(format_linije.format(proizvod.sifra, proizvod.naziv, proizvod.cena))
        prikaz.extend([
            "{} {} {}".format("-"*5, "-"*20, "-"*9),
            format_linije.format("", "Ukupna cena", self.ukupna_cena())
        ])

        return "\n".join(prikaz)
```

Slika 3.1. Povezivanje više drugih objekata za isti objekat

```
proizvodi = [
    PrehrambeniProizvod("0001", "Hleb 700g", proizvodjaci[0], 50.0, 7),
    PrehrambeniProizvod("0002", "Mleko 1l", proizvodjaci[1], 80.0, 180),
    PrehrambeniProizvod("0003", "Jogurt 1.5l", proizvodjaci[1], 130.0, 14),
    TehnickiProizvod("0004", "Samsung HDTV 42", proizvodjaci[2], 50000.0, 24),
    TehnickiProizvod("0005", "Samsung Galaxy S10", proizvodjaci[2], 95000.0, 24),
    TehnickiProizvod("0006", "iPhone XS", proizvodjaci[3], 110000.0, 24)
]

racuni = [
    Racun([proizvodi[0], proizvodi[2], proizvodi[4]]),
    Racun([proizvodi[1], proizvodi[1], proizvodi[2], proizvodi[2], proizvodi[3], proizvodi[3]]),
    Racun([proizvodi[1], proizvodi[3], proizvodi[5]])
]
for racun in racuni:
    print(racun)
```

```
        Šifra: 0e987d8a-51ac-48e0-a38b-59ad87474201
Datum i vreme: 02.04.2019. 10:37:07

Šifra                Naziv      Cena
----- -------------------- ---------
0001            Hleb 700g      50.0
0003          Jogurt 1.5l     130.0
0005   Samsung Galaxy S10   95000.0
----- -------------------- ---------
               Ukupna cena   95180.0

        Šifra: 00c99181-3123-43b8-b980-a1b296844cf2
Datum i vreme: 02.04.2019. 10:37:07

Šifra                Naziv      Cena
----- -------------------- ---------
0002             Mleko 1l      80.0
0002             Mleko 1l      80.0
0003          Jogurt 1.5l     130.0
0003          Jogurt 1.5l     130.0
0004      Samsung HDTV 42   50000.0
0004      Samsung HDTV 42   50000.0
----- -------------------- ---------
               Ukupna cena  100420.0

        Šifra: b04cbc28-57b6-4f3e-9e12-4e2fe114b735
Datum i vreme: 02.04.2019. 10:37:07

Šifra                Naziv      Cena
----- -------------------- ---------
0002             Mleko 1l      80.0
0004      Samsung HDTV 42   50000.0
0006           iPhone XS  110000.0
----- -------------------- ---------
               Ukupna cena  160080.0
```



Slika 3.2. Povezivanje više drugih objekata za isti objekat

Pogledati stranu 5!

Zadatak 1

Napisati modul *pacijent.py*. U modulu je najpre potrebno uvesti klasu *Osoba* iz modula *osoba.py* i klasu *Lekar* iz modula *zaposleni.py*.
Potrebno je definisati klasu ***Pacijent*** koja nasleđuje klasu *Osoba*, a:

1. definiše atribut *__lbo* (tipa *string*) i odgovarajući *property*
2. definiše atribut *__izabrani_lekar* (referenca na objekat klase *Lekar*) i odgovarajući *property*
3. proširuje konstruktor, tako da inicijalizuje i nove atribute
4. proširuje *__str__* metodu tako da prikazuje i nove atribute (za izabranog lekara je potrebno prikazati ime i prezime)

Test funkcija koja odgovara ovakvom programu je na slici 1.

```
lekari = [
    Lekar("lllllllllllll", "Aaa", "Aaa", 1981, "opsti"),
    Lekar("2222222222222", "Bbb", "Bbb", 1982, "opsti")
]

pacijenti = [
    Pacijent("3333333333333", "Ccc", "Ccc", 1993, "33333333333", lekari[0]),
    Pacijent("4444444444444", "Ddd", "Ddd", 1994, "44444444444", lekari[0]),
    Pacijent("5555555555555", "Eee", "Eee", 1995, "55555555555", lekari[1])
]
for pacijent in pacijenti:
    print(pacijent)
```

```
        JMBG: 3333333333333
         Ime: Ccc
     Prezime: Ccc
God. rođenja: 1993
     Starost: 26
         LBO: 33333333333
Izabr. lekar: Dr Aaa Aaa

        JMBG: 4444444444444
         Ime: Ddd
     Prezime: Ddd
God. rođenja: 1994
     Starost: 25
         LBO: 44444444444
Izabr. lekar: Dr Aaa Aaa

        JMBG: 5555555555555
         Ime: Eee
     Prezime: Eee
God. rođenja: 1995
     Starost: 24
         LBO: 55555555555
Izabr. lekar: Dr Bbb Bbb
```

Slika 1. Test funkcija modula *pacijent.py*

Potrebno je implementirati funkciju ***prikazi_po_lekarima*** klase *Pacijent*, koja u listi pacijenata prebrojava pacijente za svakog odabranog lekara. Primer rezultata ove funkcije je na slici 2.

```
class Pacijent(Osoba):

    @classmethod
    def pacijenti_po_lekarima(cls, pacijenti):
        pass
```

```
lekari = [
    Lekar("1111111111111", "Aaa", "Aaa", 1981, "opsti"),
    Lekar("2222222222222", "Bbb", "Bbb", 1982, "opsti")
]

pacijenti = [
    Pacijent("3333333333333", "Ccc", "Ccc", 1993, "33333333333", lekari[0]),
    Pacijent("4444444444444", "Ddd", "Ddd", 1994, "44444444444", lekari[0]),
    Pacijent("5555555555555", "Eee", "Eee", 1995, "55555555555", lekari[1])
]
for pacijent in pacijenti:
    print(pacijent)

Pacijent.pacijenti_po_lekarima(pacijenti)
```

```
Dr Aaa Aaa: 2
Dr Bbb Bbb: 1
```

Slika 2. Test funkcija modula *pacijent.py*

Pogledati stranu 7!

Potrebno je definisati klasu ***Pregled***, koja:

1. definiše atribut *__datum_vreme* (tipa *datetime*) i odgovarajući *property*
2. definiše atribut *__lekar* (referenca na objekat klase *Lekar*) i odgovarajući *property*
3. definiše atribut *__izvestaj* (tipa *string*) i odgovarajući *property*
4. definiše konstruktor, tako da inicijalizuje i nove atribute (samo *lekar* i *izvestaj* su parametri, a *__datum_vreme* se incijalizuje na trenutno)

Potrebno je u konstruktoru klase ***Pacijent*** definisati atribut ***__pregledi*** kao praznu listu, a zatim implementirati metodu ***dodaj_pregled*** tako da dodaje prosleđeni objekat klase *Pregled* u tu listu. Zatim je potrebno dopuniti ***__str__*** metodu klase *Pacijent* tako da nakon ostalih atributa prikaže i sve preglede. Primer rezultata je na slici 3.

```
class Pacijent(Osoba):

    def __init__(self, jmbg, ime, prezime, god_rodjenja, lbo, izabrani_lekar):
        super().__init__(jmbg, ime, prezime, god_rodjenja)
        # implementacija
        # ...

        self.__pregledi = []

    def dodaj_pregled(self, pregled):
        pass
```

```
lekari = [
    Lekar("1111111111111", "Aaa", "Aaa", 1981, "opsti"),
    Lekar("2222222222222", "Bbb", "Bbb", 1982, "opsti")
]

pacijenti = [
    Pacijent("3333333333333", "Ccc", "Ccc", 1993, "33333333333", lekari[0]),
    Pacijent("4444444444444", "Ddd", "Ddd", 1994, "44444444444", lekari[0]),
    Pacijent("5555555555555", "Eee", "Eee", 1995, "55555555555", lekari[1])
]
pacijenti[0].dodaj_pregled(Pregled(lekari[0], "Izveštaj 1"))
pacijenti[0].dodaj_pregled(Pregled(lekari[1], "Izveštaj 2"))
pacijenti[1].dodaj_pregled(Pregled(lekari[1], "Izveštaj 3"))
pacijenti[2].dodaj_pregled(Pregled(lekari[0], "Izveštaj 4"))
pacijenti[2].dodaj_pregled(Pregled(lekari[0], "Izveštaj 5"))
pacijenti[2].dodaj_pregled(Pregled(lekari[1], "Izveštaj 6"))

for pacijent in pacijenti:
    print(pacijent)

Pacijent.pacijenti_po_lekarima(pacijenti)
```

```
        JMBG: 3333333333333
         Ime: Ccc
     Prezime: Ccc
God. rođenja: 1993
     Starost: 26
         LBO: 33333333333
Izabr. lekar: Dr Aaa Aaa

Datum i vreme        Lekar           Izveštaj
-------------------- --------------- --------------------------------------------------
02.04.2019. 23:34:39 Dr Aaa Aaa      Izveštaj 1
02.04.2019. 23:34:39 Dr Bbb Bbb      Izveštaj 2

        JMBG: 4444444444444
         Ime: Ddd
     Prezime: Ddd
God. rođenja: 1994
     Starost: 25
         LBO: 44444444444
Izabr. lekar: Dr Aaa Aaa

Datum i vreme        Lekar           Izveštaj
-------------------- --------------- --------------------------------------------------
02.04.2019. 23:34:39 Dr Bbb Bbb      Izveštaj 3

        JMBG: 5555555555555
         Ime: Eee
     Prezime: Eee
God. rođenja: 1995
     Starost: 24
         LBO: 55555555555
Izabr. lekar: Dr Bbb Bbb

Datum i vreme        Lekar           Izveštaj
-------------------- --------------- --------------------------------------------------
02.04.2019. 23:34:39 Dr Aaa Aaa      Izveštaj 4
02.04.2019. 23:34:39 Dr Aaa Aaa      Izveštaj 5
02.04.2019. 23:34:39 Dr Bbb Bbb      Izveštaj 6
```

Slika 3. Test funkcija modula *pacijent.py*