## 19. Objektno programiranje

# 19.1. Razred - Lastnosti

Razred definiramo z ukazom `class`. Najbolj preprosto je narediti prazen razred, ki ne premore ničesar, razen svojega imena.

**19.2. Zgled:**

Naredimo razred, ki ga poimenujemo Oseba.

```
class Oseba:
	pass
```

Zdaj lahko generiramo objekte danega razreda.

```
janez = Oseba()
peter = Oseba()
jana = Oseba()
```

Na ta način smo kreirali tri osebe. Vsaki od njih lahko priredimo lastnosti oz. atribute. Tak atribut je npr. ime.

```
janez.ime = "Janez"
peter.ime = "Peter"
jana.ime = "Jana"
```

Osebe postavimo v seznam in izpišimo njihova imena. Priredimo jim isti priimek, npr. "Novak" in pa pravilni spol.

```
vsi = [janez,peter,jana]
for oseba in vsi:
	print(oseba.ime)
	oseba.priimek = "Novak"
	oseba.spol = "M"
jana.spol = "Ž"
```

Zdaj pa lahko izpišemo vse njihove lastnosti.

```
for oseba in vsi:
	print("Oseba", oseba.ime,oseba.priimek, " je ", "moškega" if
	oseba.spol=="M" else "ženskega", " spola.")
```

Med osebe lahko vpeljemo osnovni biološki relaciji: oče in mati. Če katerega starša ni v seznamu, osebi priredimo vrednost None.

```
for oseba in vsi:
	oseba.oce = None
	oseba.mati = None
jana.oce = peter
janez.oce = peter
```

Denimo, da je Mica Kovač mati jane, Reza Trobenta pa mati janeza.

```
mica = Oseba()
reza = Oseba()
jana.mati=mica
janez.mati = reza
mica.ime = "Mica"
mica.priimek = "Kovač"
reza.ime = "Reza"
reza.priimek = "Trobenta"
vsi.append(mica)
vsi.append(reza)
reza.spol = mica.spol = "Ž"
reza.mati = None
reza.oce = None
mica.mati = None
mica.oce = None
```

Ta način dela z objekti dopušča veliko svobode, ni pa priporočljiv, saj od zunaj neposredno dosegamo posamezne atribute objekta. Droben zatipk nam lahko povzroči velike težave. Vsako spremembo v načrtu razreda moramo popraviti pri vsakem objektu tega razreda. Zato običajno uporabljamo drugačno strategijo. Lastnosti ne dosegamo neposredno, ampak le prek posebnih metod. O metodah bomo spregovorili malo kasneje.

## 19.3. Metoda __init__ in spremenljivka self.

Ne moremo pa mimo metode __init__, ki je milo rečeno nestandardna in deluje skoraj kot "čarovnija." Na splošno se spremeljivkam, ki imajo na začetku in koncu dva podčrtaja raje izogibamo, še posebej, če ne vemo, kako delujejo. Prva izjema bo torej metoda __init__. Uporabimo jo lahko pri kreiranju posameznega objekta. Npr. v razredu Oseba bi jo lahko napisatli takole:

```
class Oseba:

       def __init__(self,ime="",priimek=""):
              self.ime = ime
              self.priimek = priimek

       def popravi(self,ime="",priimek=""):
              self.ime = ime
              self.priimek = priimek
```

Pri tem zgledu je kar nekaj posebnosti, na katere moramo biti pozorni. Za primerjavo smo dodali še metodo popravi.

### 19.4. Zgled: Grafi

```
class Graph:

    def __init__(self, edges =[]):
        self.elist = []
        self.vlist = []
        self.adic = {}
        for (u,v) in edges:
            if u < v:
                self.addedge(u,v)
            else:
                self.addedge(v,u)


    def __str__(self):
        return str(self.edges)

    def addedge(self,u,v):
        if u > v:
            u,v = v,u
        self.elist.append((u,v))
        if not u in self.vlist:
            self.vlist.append(u)
            self.adic[u] = []
        if not v in self.vlist:
            self.vlist.append(v)
            self.adic[v] = []
        self.adic[u].append(v)
        self.adic[v].append(u)
```

```
    def vertices(self):
        self.vlist.sort()
        return self.vlist

    def edges(self):
        self.elist.sort()
        return self.elist

    def nv(self):
        return len(vertices(self))

    def ne(self):
        return len(edges(self))



def K(n):
    edges = []
    for u in range(n):
        for v in range(u):
            edges.append((v,u))
    return Graph(edges)

def K(m,n):
    edges = []
    for u in range(n):
        for v in range(n,n+m):
            edges.append((v,u))
    return Graph(edges)
```

## 19.5. Lastnosti razreda in lastnosti objekta

Zavedati se moramo, da spremenljivka lahko pripada posameznemu objektu, lahko pa je skupna vsem objektom istega razreda, pripada torej celotnemu razredu. Tega dejstva mi sicer ne bomo podrobneje obravnavali.

## 19.6.Naloge:

1. Z razredi realizirajte sklad.
2. Z razredi realizirajte vrsto.
3. Z razredi realizirajne dvojiško drevo (prednikov).
4. Z razredi realizirajte usmerjeni graf. Za dano besedilo sestavite graf, ki ima na povezavi x->y utež, ki pove, kolikokrat znak y stoji tik za znakom x.