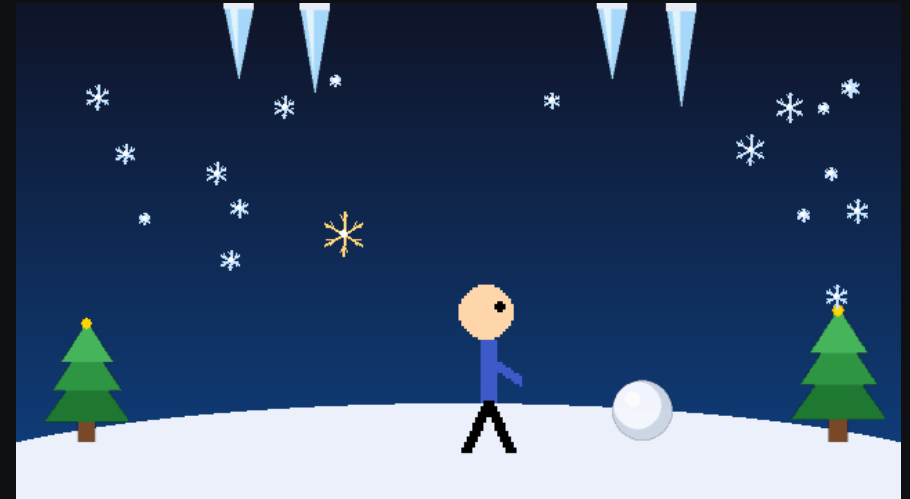


Nasleđivanje

Kako nova klasa preuzme sve od postojeće i doda svoje

Danas pravimo: porodicu objekata za igru „Zimska šetnja“: igrača, pahuljicu, ledenicu i grudvu



Svi su objekti igre: imaju položaj i umeju da se pomere. Svako dodaje svoje.

Tri pitanja za zagrevanje

1 • prošla tema: brojač

```
class Jelka:
    ukupno = 0

    def __init__(self):
        Jelka.ukupno += 1

a = Jelka()
b = Jelka()
a.ukupno = 9
print(b.ukupno)
```

2

2 • tema 02: self

```
class Lik:
    def skoci(self):
        self.y = 10

mia = Lik()
mia.skoci()
mia.y += 5
print(mia.y)
```

15

3 • mešano: copy i liste

```
import copy

a = [[1], [2]]
b = copy.copy(a)
b[0].append(9)
print(a)
```

[[1, 9], [2]]

Šest novih reči za danas

nasleđivanje

nova klasa dobija sve metode postojeće klase

roditeljska klasa

postojeća klasa od koje se nasleđuje (zove se i bazna)

izvedena klasanova klasa: `class Pahuljica(ObjekatIgre)`**super()**poziv roditeljske verzije metode, npr. `super().__init__(x, y)`**redefinisanje**

izvedena klasa piše svoju metodu sa istim imenom

isinstance

pita da li je objekat te klase ili njene izvedene klase

Pahuljica je objekat igre, plus još nešto

ObjekatIgre

roditeljska klasa

```
x, y  
pomeri(dx, dy)  
opis()
```



Pahuljica(ObjekatIgre)

```
x, y, pomeri, opis  
← nasleđeno
```

```
+ brzina  
+ padni(dt)
```



- Izvedena klasa dobija sve što ima roditeljska: attribute koje postavlja roditeljski `__init__` i sve metode.
- Zatim dodaje svoje, ili neku metodu piše iznova.
- Nasleđivanje opisuje odnos „jeste“: pahuljica jeste objekat igre. Jelka takođe jeste, a igrač takođe.

Prva izvedena klasa

Sve od roditelja, plus svoje

Izvedena klasa nasleđuje sve metode

```
class ObjekatIgre:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def pomeri(self, dx, dy):
        self.x += dx
        self.y += dy

    def opis(self):
        return f"objekat u ({self.x}, {self.y})"
```

```
class Jelka(ObjekatIgre):
    pass # sve nasleđeno
```

```
j = Jelka(40, 220)
j.pomeri(10, 0)
print(j.opis())
```

ISPIS

objekat u (50, 220)

class Jelka(ObjekatIgre): u zagradi je roditeljska klasa.

`super().__init__` postavlja roditeljske attribute

```
class Pahuljica(ObjekatIgre):  
    def __init__(self, x, y, brzina):  
        super().__init__(x, y)    # roditelj: x i y  
        self.brzina = brzina      # pa svoje  
  
    def padni(self, dt):  
        self.y += self.brzina * dt  
  
p = Pahuljica(100, 0, 60)  
p.padni(0.5)  
print(p.opis(), p.brzina)
```

ISPIS

objekat u (100, 30.0) 60

`super()`

roditeljska klasa, ovde
ObjekatIgre

`super().__init__(x, y)`

pokreće roditeljski `__init__` za ovaj
objekat

`self.brzina`

posle toga pahuljica dodaje
svoje

Šta ispisuje ovaj program?

```
class ObjekatIgre:  
    def zvuk(self):  
        return "tišina"  
  
class Prepreka(ObjekatIgre):  
    pass  
  
class Ledenica(Prepreka):  
    def zvuk(self):  
        return "krc"  
  
print(Prepreka().zvuk(), Ledenica().zvuk())
```

Prepreka nema svoju metodu zvuk, a Ledenica ima.

A

tišina tišina

B

krc krc

C

tačno

tišina krc

D

AttributeError

Čudovište: zaboravljen `super().__init__`

```
class ObjekatIgre:
    def __init__(self, x, y):
        self.x = x
        self.y = y

class Igrac(ObjekatIgre):
    def __init__(self, x, y, ime):
        self.ime = ime # bez super()

mia = Igrac(300, 290, "Mia")
print(mia.ime)
print(mia.x)
```

Kako izgleda

`mia.ime` radi, a `mia.x` ruši program: `AttributeError: 'Igrac' object has no attribute 'x'`

Zašto se pojavljuje

Igrac ima svoj `__init__`, pa se roditeljski ne pokreće sam. Niko nije postavio `x` i `y`.

Kako se pobeđuje

U prvom redu svog `__init__` pozovite `super().__init__(x, y)`.

Redefinisanje

Ista metoda, nova verzija

Redefinisana metoda može da pozove roditeljsku

```
class Igrac(ObjekatIgre):  
    def __init__(self, x, y, ime):  
        super().__init__(x, y)  
        self.ime = ime
```

```
    def opis(self):                                # redefinisan  
        return self.ime + ", " + super().opis()
```

```
mia = Igrac(300, 290, "Mia")  
print(mia.opis())  
print(ObjekatIgre(5, 5).opis())
```

ISPIS

```
Mia, objekat u (300, 290)  
objekat u (5, 5)
```

- Igrac ima svoj opis: za igrače važi taj.
- `super().opis()` je roditeljska verzija, pa ne pišemo isto dva puta.
- `ObjekatIgre` i dalje ima stari opis.

Kvadrat je pravougaonik sa jednakim stranicama

```
class Pravougaonik:
    def __init__(self, sirina, visina):
        self.sirina = sirina
        self.visina = visina

    def povrsina(self):
        return self.sirina * self.visina

class Kvadrat(Pravougaonik):
    def __init__(self, stranica):
        super().__init__(stranica, stranica)

k = Kvadrat(5)
print(k.povrsina(), k.sirina == k.visina)
```

ISPIS

25 True

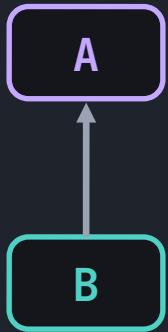
- Kvadrat ne piše površinu: nasleđuje je.
- Njegov `__init__` prima jednu stranicu i roditelju šalje dve iste.
- Zadatak iz starih beleški; cela verzija sa centrom i `__str__` je u primerima.

Vrste nasleđivanja

Pet oblika porodičnog stabla klasa

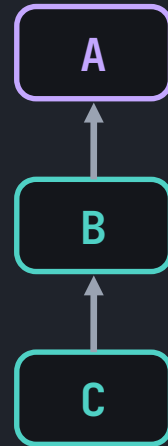
Pet vrsta nasleđivanja

jednostruko



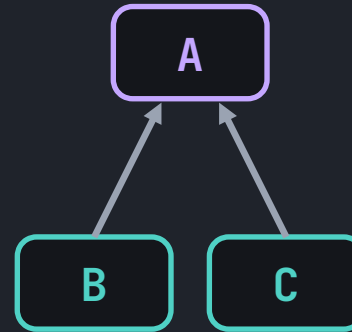
```
class B(A)
```

višenivovsko



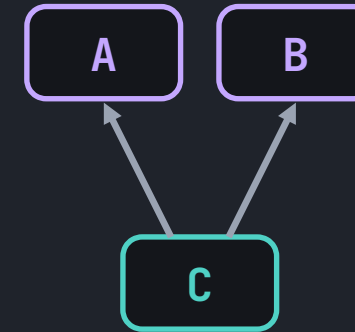
```
class B(A)  
class C(B)
```

hijerarhijsko



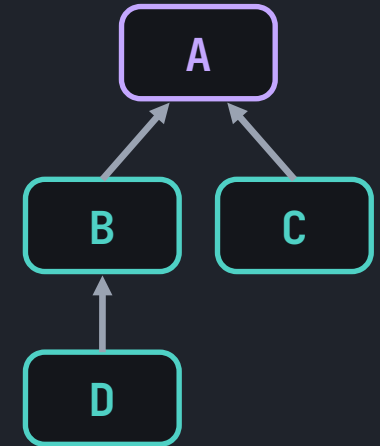
```
class B(A)  
class C(A)
```

višestruko



```
class C(A, B)
```

hibridno



spoj bar dve
vrste, npr.
hijerarhijsko
+ višenivovsko

Drvo objekata u igri „Zimska šetnja“



Tri klase istog roditelja: hijerarhijsko.

ObjekatIgre → Prepreka → Ledenica: višenivovsko.

Zajedno: hibridno nasleđivanje.

Ledenica nasleđuje kroz dva nivoa

```
class Prepreka(ObjekatIgre):
    def __init__(self, x, y, steta):
        super().__init__(x, y)
        self.steta = steta

class Ledenica(Prepreka):
    def __init__(self, x):
        super().__init__(x, 0, 1) # vrh, šteta 1
        self.otopljena = False

l = Ledenica(220)
print(l.x, l.y, l.steta, l.otopljena)
l.pomeri(0, 15)
print(l.opis())
```

ISPIS

```
220 0 1 False
objekat u (220, 15)
```

- x, y i pomeri: od ObjekatIgre.
- steta: od Prepreke.
- otopljena: samo ledenica.

Višestruko: ilustrator je i crtač i pisac

```
class Crtac:  
    def crtaj(self):  
        return "crtam"  
  
    def potpis(self):  
        return "crtač"
```

```
class Pisac:  
    def pisi(self):  
        return "pišem"  
  
    def potpis(self):  
        return "pisac"
```

```
class Ilustrator(Crtac, Pisac):  
    pass  
  
a = Ilustrator()  
print(a.crtaj(), a.pisi())  
print(a.potpis())
```

```
ISPIS  
  
crtam pišem  
crtač
```

Isto ime metode u oba roditelja: važi klasa napisana prva u zagradi.

Šta ispisuje ovaj program?

```
class Zvuk:  
    def ogласи_se(self):  
        return "dzing"  
  
class Svetlo:  
    def ogласи_se(self):  
        return "blic"  
  
class ZlatnaPahuljica(Svetlo, Zvuk):  
    pass  
  
print(ZlatnaPahuljica().ogласи_se())
```

Pogledajte redosled klasa u zagradi.

A

dzing

B

tačno

blic

C

dzing blic

D

TypeError

isinstance

Pitamo objekat: jesi li i ti iz ove porodice?

isinstance prihvata i roditeljske klase

```
l = Ledenica(220)
print(isinstance(l, Ledenica))
print(isinstance(l, Prepreka))
print(isinstance(l, ObjekatIgre))
print(isinstance(l, Igrac))
print(type(l) == Prepreka)
print(issubclass(Ledenica, ObjekatIgre))
```

ISPIS

```
True
True
True
False
False
True
```

- Ledenica jeste i Prepreka i ObjekatIgre.
- `type(l) == Prepreka` traži baš tu klasu, pa je False.
- `issubclass` pita isto za klase.

Šta ispisuje ovaj program?

```
class Lik:
    pass

class Igrac(Lik):
    pass

class Neprijatelj(Lik):
    pass

n = Neprijatelj()
print(isinstance(n, Lik), isinstance(n, Igrac))
```

Igrac i Neprijatelj imaju istog roditelja.

A

True True

B

tačno

True False

C

False False

D

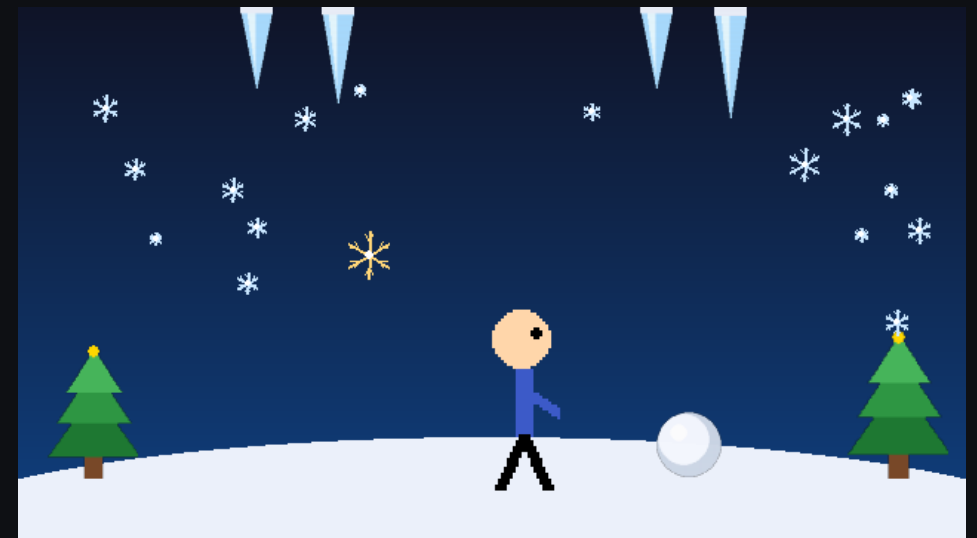
False True

Zimska šetnja: porodica objekata igre

Cilj: klase za sve što postoji u igri, sa zajedničkim roditeljem `ObjekatIgre`. Na kraju jedna petlja opiše ceo svet igre.

- 1 `ObjekatIgre`: `x`, `y`, pomeri, opis
- 2 `Igrac` i `Pahuljica`: `super().__init__` i svoje
- 3 Prepreka sa štetom, pa `Ledenica` i `Grudva`
- 4 Lista `svet` sa objektima svih vrsta
- 5 `isinstance`: ko pada, a ko nanosi štetu

SVI OBJEKTI IZ IGRE



Danas u konzoli; kad igra dobije prozor, svaki od njih dobija i metodu `crtaj`.

Igrač i pahuljica: svako dodaje svoje

```
class Igrac(ObjekatIgre):  
    def __init__(self, x, y, ime):  
        super().__init__(x, y)  
        self.ime = ime  
        self.zivoti = 3  
  
    def opis(self):  
        return f"igrač {self.ime}, životi {self.zivoti}"  
  
class Pahuljica(ObjekatIgre):  
    def __init__(self, x, y, brzina):  
        super().__init__(x, y)  
        self.brzina = brzina          # piksela u sekundi  
  
    def padni(self, dt):  
        self.pomeri(0, self.brzina * dt)
```

Igrac.opis

redefinisan: igrač se opisuje imenom i životima

Pahuljica.padni

koristi nasleđenu metodu pomeri

Pahuljica.opis

nije napisan: važi onaj iz ObjekatIgre

Prepreke: dva nivoa nasleđivanja

```
class Prepreka(ObjekatIgre):
    def __init__(self, x, y, steta):
        super().__init__(x, y)
        self.steta = steta          # koliko života oduzima

    def opis(self):
        return f"prepreka (šteta {self.steta}), " + super().opis()

class Ledenica(Prepreka):
    def __init__(self, x):
        super().__init__(x, 0, 1)

    def opis(self):
        return "ledenica: " + super().opis()

class Grudva(Prepreka):
    def __init__(self, x, y):
        super().__init__(x, y, 2)
```

Ledenica

šteta 1, svoj opis

Grudva

šteta 2, opis od Prepreke

Jedna lista, ceo svet igre

```
svet = [Igrac(300, 290, "Mia"),
        Pahuljica(100, 0, 60),
        Ledenica(220),
        Grudva(500, 120)]
for o in svet:
    if isinstance(o, Pahuljica):
        o.padni(0.5)
for o in svet:
    print(o.opis())

steta = 0
for o in svet:
    if isinstance(o, Prepreka):
        steta += o.steta
print("Ukupna šteta prepreka:", steta)
```

ISPIS

```
igrač Mia, životi 3
objekat u (100, 30.0)
ledenica: prepreka (šteta 1), objekat u (220, 0)
prepreka (šteta 2), objekat u (500, 120)
Ukupna šteta prepreka: 3
```

- Isti poziv o.opis() radi za svaki objekat, svako na svoj način.
- isinstance(o, Prepreka) je True i za ledenicu i za grudvu.

Pronađi grešku: pahuljica se ne pravi

```
class Pahuljica(ObjekatIgre):  
    def __init__(self, x, y, brzina):  
        super().__init__(self, x, y)  
        self.brzina = brzina
```

```
p = Pahuljica(100, 0, 60)
```

Program se ruši u poslednjem redu. Ko dobija argument previše?

PORUKA (JEDAN RED)

```
TypeError: ObjekatIgre.__init__() takes 3 positional arguments but 4 were given
```

super()

već zna za self

(self, x, y)

tri argumenta + objekat = 4

ispravka

super().__init__(x, y)

Izaberite nivo i počnite

LEVEL 1

Jelka

Jelka nasleđuje `ObjekatIgre`, dodaje visinu i redefiniše opis tako da počinje rečima „jelka visoka ...“.

Pomoć: `super().__init__(x, y)` i `super().opis()`.

LEVEL 2

Nameštaj

Ispitni zadatak: `Namestaj` (težina, visina), `Sto` i `Stolica` sa svojim atributima i metodom `__str__`.

Pomoć: `__str__` izvedene klase počinje sa `super().__str__()`.

LEVEL 3

Zlatna pahuljica

`ZlatnaPahuljica` nasleđuje `Pahuljicu` i klasu `Svetlo`, vredi 5 poena i ume da zasvetli.

Pomoć: `class ZlatnaPahuljica(Pahuljica, Svetlo)`.

Sledeći put: isti poziv, različito ponašanje

```
for o in svet:  
    print(o.opis())    # svako svoj
```

Petlja iz naše igre već radi nešto važno: isti poziv `o.opis()` svaki objekat izvrši na svoj način. U temi 05 to dobija ime, polimorfizam. Videćemo i kako da igraču zaštitimo broj života, da nikad ne padne ispod nule.

Rezime

izvedena klasa

```
class Pahuljica(ObjekatIgre):
```

Dobija sve metode roditeljske klase.

super()

```
super().__init__(x, y)
```

```
super().opis()
```

Roditeljska verzija; bez self u zagradi.

vrste

jednostruko · višenivovsko

hijerarhijsko · višestruko · hibridno

Hibridno je spoj bar dve vrste.

isinstance

```
isinstance(l, Prepreka)
```

True i za objekte izvedenih klasa.

Pitanje pre odlaska

```
class Lik:  
    def zvuk(self):  
        return "hm"  
  
class Igrac(Lik):  
    def zvuk(self):  
        return super().zvuk() + "!"  
  
print(Igrac().zvuk())
```

Šta ispisuje ovaj program?

Pošaljite odgovor pre nego što izađete.



Literatura

- Zvanični Python tutorijal, poglavlje 9 „Classes“, odeljak 9.5 o nasleđivanju: docs.python.org/3/tutorial
- Lutz, M. Learning Python, 5. izdanje, O'Reilly, 2013 (delovi o nasleđivanju)
- Wentworth, P., Elkner, J., Downey, A. B., Meyers, C. How to Think Like a Computer Scientist: Learning with Python 3 (besplatna knjiga na internetu)
- Nystrom, R. Game Programming Patterns, poglavlje „Component“ (besplatno na gameprogrammingpatterns.com)

Dodatna literatura i zanimljivi sajtovi

Python tutorijal: nasleđivanje

docs.python.org/3/tutorial/classes.html

Odeljak 9.5: nasleđivanje, isinstance, issubclass i višestruko nasleđivanje.

Python: funkcija super

docs.python.org/3/library/functions.html

Zvaničan opis funkcije super; na istoj stranici su isinstance i issubclass.

Real Python: nasleđivanje i kompozicija

[realpython.com · inheritance, composition](https://realpython.com/inheritance-composition)

Nasleđivanje, super(), višestruko nasleđivanje i kompozicija.

Game Programming Patterns: Component

gameprogrammingpatterns.com/component.html

Zašto drvo klasa u igri brzo postane nezgodno i šta se radi umesto toga.

Hvala!

Pitanja su uvek dobrodošla: na času, u četvrtu ili međim.

Slajdovi, primeri i kartica za podsećanje su na sajtu predmeta.

PREDAVANJA

dr Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic

VEŽBE

Milan Kocić

milan.kocic@matf.bg.ac.rs

poincare.matf.bg.ac.rs/~milan.kocic