Полиморфизам

Полиморфизам је следећи концепт: исте операције се примењују на различите објекте, без обзира на њихову класу. Нпр, нека се користи програм који пушта аудио фајлове; плејер треба да учита објекат АудиоФајл а затим и да га стартује.

Користићемо методу Старт() на објекту, која је задужена за декомпресију или извлачење аудија и усмеравање на звучну картицу и звучнике.

Само пуштање АудиоФајла се изводи са: audio_fajl.start().

Ипак, процес декомпресије и извлачења аудио фајла је различит за различите формате; фајл .wav је некомпресован док .mp3, .wma, .ogg имају различите алгоритме компресије.

Може се користити наслеђивање са полиморфизмом за поједностављење дизајна.

Сваки тип фајла се може приказати са другачијом подкласом AudioFajla, нпр, WavFajl, MP3Fajl.

Сваки од њих мора имати start() методу, али та метода ће се различито имплементирати за сваки од формата да би се осигурала тачна декомпресија.

Објекат плејера не мора знати на коју подкласу AudioFajl-а се односи; само стартује start() и полиморфно омогућава објекту да води рачуна о детаљима пуштања музике.

```
class AudioFajl:
    def __init__(self, ime_fajla):
        if not ime_fajla.endswith(self.ext):
            raise Exception("Pogresan format fajla")

        self.ime_fajla = ime_fajla

class MP3Fajl(AudioFajl):
    ext = "mp3"
    def start(self):
        print("startovano {} kao mp3".format(self.ime_fajla))

class WavFajl(AudioFajl):
    ext = "wav"
    def start(self):
        print("startovano {} kao wav".format(self.ime_fajla))

class OggFajl(AudioFajl):
    ext = "ogg"
    def start(self):
        print("startovano {} kao ogg".format(self.ime_fajla))
```

Сви аудио фајлови се проверавају да би се осигурало да добра екстензија се даје током иницијализације.

Али, види се да је __init__ метода у родитељској класи способна да приступи промењивој ext класе из различитих подкласа, што је одлика полиморфизма.

Ако се ime_fajla не завршава на коректан начин, јавља се изузетак.

Чињеница да AudioFajl не смешта референцу на промењиву ext га не спречава да му прође у подкласи.

И још, свака подкласа од AudioFajl имплементира start() на различит начин.

И то је утицај полиморфизма пошто плејер може користити исти код за стартовање фајла, без обзира на формат, тј нема везе на коју се од подкласа AudioFajl односи.

Детаљи декомпресије AudioFajla су енкапсулирани.

```
ogg = OggFajl("mojfajl.ogg")
ogg.start()
mp3 = MP3Fajl("mojfajl.mp3")
mp3.start()
nije_mp3 = MP3Fajl("mojfajl.ogg")
```

Пример даје:

```
startovano mojfajl.ogg kao ogg
startovano mojfajl.mp3 kao mp3
Traceback (most recent call last):
  File "C:\Users\nera\source\repos\PythonApplication5\PythonApplication5.py", line 27, in <module>
```

```
   nije_mp3 = MP3Fajl("mojfajl.ogg")
 File "C:\Users\nera\source\repos\PythonApplication5\PythonApplication5.py", line 4, in __init__
   raise Exception("Pogresan format fajla")
Exception: Pogresan format fajla
```

Види се како је AudioFajl.__init__ способан да провери тип фајла без да зна на коју се подкласу заправо односи.
У Пајтону је могуће користити било који објекат који омогућава тражено понашање без да се приморава да мора бити подкласа.
Динамичка природа Пајтона поједностављује проблеме полиморфизма.
Следећи пример не проширује AudioFajl, али може да се користи са истим интерфејсом:

```
class FlacFajl:
    def __init__(self, ime_fajla):
        if not ime_fajla.endswith(".flac"):
            raise Exception("Pogresan format fajla")

        self.ime_fajla = ime_fajla

    def start(self):
        print("startovano {} kao flac".format(self.ime_fajla))
```

Плејер може користити овај објекат исто једноставно као и онај који ради са AudioFajl.
Полиморфизам је један од најважнијих разлога за коришћење наслеђивања у многим ОО контекстима.
Пошто било који објекат који снабдева добар интерфејс се може користити са другим деловима кода у Пајтону, то умањује потребу за полиморфном основном надкласом.
Наслеђивање и даље може бити корисно за дељење кода али ако је све доступно као јавни интерфејс, довољно је само привезати потребан код.
Овиме се смањује потреба за наслеђивањем што смањује потребу за вишеструким наслеђивањем; често, када вишеструко наслеђивање се чини да је добро решење, можемо само прилепити код и тако наизглед користити вишеструке надкласе.

Израда лабораторијских вежби:
Задатак 055: Креирати класе за две државе са истим методама. Инстанцирати класе за различитим објектима и показати различито деловање истих метода у класама због дејства полиморфизма.

```
class Srbija():
    def glavni_grad(self):
        print("Beograd.")

    def jezik(self):
        print("Srpski.")

    def tip_zemlje(self):
        print("Zemlja u razvoju.")
class SAD():
    def glavni_grad(self):
        print("Vasington.")

    def jezik(self):
        print("Engleski.")

    def tip_zemlje(self):
        print("Visoko razvijena zemlja.")

srbija = Srbija()
amerika = SAD()
for zemlja in (srbija, amerika):
    zemlja.glavni_grad()
    zemlja.jezik()
    zemlja.tip_zemlje()
```

Задатак 056: Креирати надкласу Птица и подкласе Врабац и Ној. Исту методу класе Летење користити у свим класама а са различитим садржајем. Инстанцирањем различитих објеката проверити полиморфизам код класа које наслеђују.

```
class Ptica:
  def opis(self):
    print("Postoji veliki broj vrsta ptica.")

  def letenje(self):
    print("Neke ptice mogu da lete a druge ne.")

class Vrabac(Ptica):
  def letenje(self):
    print("Vrabac moze da leti.")

class Noj(Ptica):
  def letenje(self):
    print("Noj ne moze da leti.")

ptica = Ptica()
vrabac = Vrabac()
noj = Noj()
ptica.opis()
ptica.letenje()
vrabac.opis()
vrabac.letenje()
noj.opis()
noj.letenje()
```

Задатак 057: Креирати класе Медвед и Пас са истим методом класе звук али са различитим садржајем. Креирати функцију глас_животиње која приказује гласове животиње. Проверити полиморфизам инстанцираних објеката.

```
class Medved(object):
    def zvuk(self):
        print("Grrrrr")

class Pas(object):
    def zvuk(self):
        print("Vau vau")

def glas_zivotinje(tip_zivotinje):
        tip_zivotinje.zvuk()

meda = Medved()
kuca = Pas()
glas_zivotinje(meda)
glas_zivotinje(kuca)
```

Задатак 058: Креирати надкласу Особа са методама опис и кретање. Креирати подкласу Беба. Извршити metod overriding над методом кретање у подкласи Беба пошто родитељска метода не одговара.

```
class Osoba:
    def kretanje(self):
        print("Hoda uspravno.")

class Beba(Osoba):
    def kretanje(self):
        print("Puzi.")

beba = Beba()
```

```
tinejdzer = Osoba()
beba.kretanje()
tinejdzer.kretanje()
```

Задатак 059: Креирати надкласу Особа са методама опис и кретање. Креирати подкласу Дете. Извршити metod overriding над методом кретање у подкласи Дете пошто родитељска метода не одговара али оставити могућност приступа родитељској методи.

```
class Osoba:
    def kretanje(self):
        print("Ponekad hoda uspravno.")

class Dete(Osoba):
    def kretanje(self):
        super().kretanje()
        print("Ponekad puzi.")

dete = Dete()
dete.kretanje()
```

Задаци за самосталан рад:

55. Креирати самосталне функције тест_летења и тест_пливања које испитују да ли објекти класе Папагај и Пингвин умеју да лете и пливају. Испитати плиморфизам различитим исходима над овим функцијама.

56. Користити три исте методе у класама Свемир и Земља па преко полиморфизма указати на различито понашање метода са истим именом.

57. Коришћењем надкласе Спорт са методом опис и методом употреба_лопте и подкласа Фудбал и Кошарка и њиховим методама употреба_лопте, доказати дејство полиморфизма унтар објеката подкласа.

58. Доказати принцип полиморфизма у релацији Ауто, Возач и Путник.

59. Применити metod overriding над родитељском методом у релацији Тата, Мама и Беба.