

Algoritmi i strukture podataka

Objektno orijentisano programiranje

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2019

Ciljevi

- Robusnost
- Adaptivnost
- Ponovna iskoristivost

Principi objektno-orientisanog dizajna

- Modularnost
- Apstrakcija
- Enkapsulacija

Modularnost

- Podela sistema u nezavisne funkcionalne celine
- Benefiti:
 - Olakšava testiranje komponenti
 - Omogućava ponovnu iskoristivost
 - Smanjuje dupliciranje koda
- Python:
- Skup srodnih funkcija i klasa
- Definiše se unutar jednog py fajla
- Upotreba:

```
import re from os
import * from math import pi, sqrt
```

Apstrakcija

- Izdvajanje najvažnijih karakteristika sistema
- Kreira apstraktne tipove podatka (ATP)
- ATP specifikuju **šta** operacije rade, ali ne i **kako**
- Skup operacija koje ATP definiše naziva se **interfejs**
- Python:
 - Duck typing
 - Apstraktne bazne klase

Enkapsulacija

- Detalji implementacije ostaju sakriveni
- Omogućava nezavisnost prilikom implementacije komponenti
- Python:
 - članovi klase čije ime počinje donjom crtom su privatni i nisu namenjeni upotrebi izvan klase (npr. `_item`)

Terminologija

- Objekat je **instanca** klase
- Metode klase definišu **ponašanje**
- Atributi klase definišu **stanje**

Identifikator *self*

- Svaka klasa može imati više instanci
- Svaki objekat ima svoje stanje
- ***self*** predstavlja instancu za koju je metoda pozvana

Primer klase

```
class Student(object):
    def __init__(self, ime, prezime):
        self._ime = ime
        self._prezime = prezime
    def get_ime(self):
        return self._ime
    def get_prezime(self):
        return self._prezime
    def predstavi_se(self):
        return 'Ja sam ' + self._ime + ' ' + self._prezime
```

Konstruktori

- Kreiraju instance klase
- Poziv se svodi na poziv metode `__init__`
- Primer:

```
s = Student('Jelena', 'Jovin')
```

Preklapanje operatora

- Omogućava izmenu semantike postojećih operatora
- Postiže se redefinisanjem specijalnih metoda

Iteratori

- Omogućavaju pristup elementima kolekcije
- Redefiniše se metoda `_next_`
- Metoda vraća sledeći element u kolekciji ili izaziva **StopIteration** izuzetak

Nasleđivanje

- Omogućava modularnu i hijerarhijsku organizaciju
- Nova klasa definiše se na osnovu postojeće
- Polazna klasa se naziva **bazna, predak ili superklasa**
- Izvedena klasa naziva se **potklasa ili klasa potomak**

Nasleđivanje

- Opšti format:

```
class NazivKlase(A, B, ...):  
    # telo klase ...
```

- new-style vs old-style

```
class A:  
    # old-style
```

-

```
class B(object):  
    # new-style
```

- Pristup roditeljskoj klasi pomoću **super**

Zadatak 1

- Napisati klasu **Point** koja predstavlja tačke u ravni.
- Klasa treba da sadrži:
 - metode za pristup koordinatama
 - metodu **_str_**
 - metodu za računanje rastojanja od druge tačke

Zadatak 2

- Napisati klasu **Rectangle** koja reprezentuje pravougaonik.
- Klasa sadrži metode za izračunavanje obima i površine pravougaonika.
- Na osnovu napisane klase izvesti klasu **Square**.