OBJEKTNO ORJENTISANO PROGRAMIRANJE

Šta je OO programiranje?

Objektno orijentisano programiranje je metodologija programiranja kojom se modelira realan svet kao skup objekata i odnosa među objektima. Osnovni element u ovoj metodologiji su objekti koji se koriste pri izvršavanju programa.

Pomislite o objektima koji nas okužuju – automobili, ptice, drveće, ljudi itd. Ima ih bezbroj svuda oko nas. Uobičajeno je da sve objekte koji nas okružuju klasifikujemo pa otuda i jedan od osnovnih pojmova kod OO metodologije – pojam klase (Class).

Klasa (Class)

Klasa je sastavljena od osobina objekta koje se izražavaju podacima koje ih opisuju (atribut, properti) i akcija koje objekti mogu sprovoditi (metoda).

Uzmimo na primer automobil: podaci (atributi) koji karakterišu automobile su brzina, boja, broj sedišta, itd., a aktivnosti (metode) koje ovaj objekat može da obavlja su ubrzavanje, kočenje, promena stepena brzine (menjačem) itd.

Ili, uzmimo za primer objekte kao što su studenti. Atributi studenta su ime, prezime, broj indeksa, godina rođenja, godina studija, itd.

Aktivnosti koje studenti izvršavaju su upis semestra, prijava ispita, polaganje ispita, itd.

Ili, recimo posmatrajmo automat za kafu. Kao atribute automata mozemo da uzmemo boju, listu artikala koji se nude, iznos novca u kasi, trenutnu uplatu, itd. Metode kod automata mogu biti izbor artikla, ubacivanje novca, preuzimanje artikla, preuzimanje kusura, odustajanje od kupovine, itd.

Klasa opisuje sve objekte datog tipa, na taj način što definiše koji će podaci biti korišćeni za opis svakog pojedinačnog objekta. Zato je klasa apstraktna struktura bez pojedinačnih vrednosti za podatke kojima se opisuju objekti. Podaci u obliku vrednosti atributa pojavljuju se tek kada se kreira poseban objekat iz već definisane klase.

Objekti (Object)

Objekat je jedna konkretizacija klase, odnosno objekat je jedan konkretan primerak sa konkretnim vrednostima atributima, koji ga razlikuju od drugih objekata iz iste klase. Na primer u klasi student, koju smo ranije opisali, možemo kreirati objekat “Jovana Mirković” koji prestavlja jedan poseban slučaj objekta (konkretizaciju klase) iz klase “Student”.

Razmena poruka (Message Passing)

Objekti ne postoje izolovano. Oni interaguju sa drugim objektima. U OO programiranju ova interakcija se ostvaruje preko poruka. Prenos poruka je proces u kome jedan objekat (pošiljalac-sender) šalje podatke drugom objektu (primaocu- receiver) ili traži od drugog objekta da aktivira neki svoj metod. Tako svaka poruka ima pošiljaoca i primaoca.

Ponašanje objekata (Behavior)

Ponašanje se odnosi na to kako objekti reaguju na poruke, to jest kako menjaju sopstveno stanje (atribute) i/ili aktiviraju svoje metode.

Pogledajmo sada kako se OOP koristi u Python-u.

Deklarisanje nove klase

Kada želimo da deklarišemo (definišemo, opišemo) novu klasu, u Pythonu to činimo uz pomoć sledeće sintakse:

```
class ImeKlase:
    # atributi, podaci kojima se opisuju objekti iz klase
    # metode, kojima se opisuju akcije (funkcije) koje objekti izvršavaj
```

Ovde je class ključna reč. Ime klase može biti bilo koje ime koje želimo da damo klasi, ali u skaldu sa pravilima imenovanje klase.

Konvencija za davanje imena varijablama, funkcijama, klasama ...

U Python-u se preporučuje upotreba:

“`UpperCamelCase`” imenovanja za klase,

“`CAPITALIZED_WITH_UNDERSCORES`” za konstante, a

“`lowercase_separated_by_underscores`” za ostala iemna.

Primer definisanja klase:

```
class Student:
    def __init__(self,ime = "",prezime = "",broj_indeksa = ""): #konstruktor
        self.ime = ime
        self.prezime = prezime
        self.broj_indeksa = broj_indeksa
    def prikaz(self):
        print ( "%-10s %s %s" % (self.broj_indeksa, self.prezime, self.ime) )
```

Kreiranje objekata

Kada želimo da kreiramo objekat iz neke klase, to radimo na sličan način kao kada definišemo novu varijablu.

Na primer, naredbom:

```
s1 = Student("Novak","Djoković", "123/15")
```

se kreira jedan novi objekat klase Student, sa zadatim vrednostima atributa.

Korišćenje objekata u OO programima

Za pistup atributima i metodama kreiranog objekta koristi se . (dot) operator. Na primer, ako želimo da aktiviramo metod za prikaz na ekranu podataka objekta s1 iz klase student pisaćemo:

```
s1.prikaz()
```

Atributima se može pristupati na sličan način, na primer:

```
s1.ime = "Stojan"
```

Prednosti korišćenja OO programiranja

- Realistično modeliranje realnosti: Korišćenje OO metodologije daje mogućnost boljeg i verodostojnijeg modeliranja procesa iz naše realnosti.
- Višestruko korišćenje klasa: Jedanput dizajnirana klasa može biti iskorišćena u raznim aplikacijama.
- Olakšano održavanje softvera: Korišćenjem OO metodologije značajno se olakšava održavanje softvera, jer je jednostavnija identifikacija modula koje treba menjati.

Osnovna svojstva OO metodologije

Sada ćemo se ukratko baviti nekim osnovnim svojstvima OO metodologije.

Učaurivanje, enkapsulacija (Encapsulation, data hiding)

Enkapsulacija je jedan važan koncept u OO programiranju. Spoljni svet, kako je već rečeno, pristupa samo onim atributima i metodama koje su označene kao public. Programer koji koristi neku klasu ne mora da zna kako su atributi i metode u samoj klasi kodirane. Sve što je njemu potrebno je da poznaje intefejs preko kojeg pristupa metodama i atributima klase. Naravno, potrebno je da razume i značenje atributa i metoda kojima pristupa, ali ne mora da zna kako su same metode programirane. Na taj način je obezbeđeno da se greške koje se pojave u klasi otklanjaju u samoj klasi, bez potrebe za promenom onog dela programa koji tu klasu koristi.

Apstrakcija (Abstraction)

Apstrakcija je proces kojim se objekti iz realnog sveta modeliraju uprošćenom “slikom” napravljenom u klasi pomoću atributa i metoda. Recimo klasa automobil koju smo napred pomenuli ne opisuje sve moguće karakteristike automobola, već neke osobine zanemaruje (apstrahuje) i ne prikazuje ih kroz atribute i metode. Koji atributi i metodi će biti odabrani za prikaz u klasi zavisi od namera koje imamo u pogledu budućeg korišćenja klase u raznim aplikacijama. Pravilno odabran model realnog sveta može da pomogne da se razumeju rešenja problema koje treba rešiti kroz softversku aplikaciju.

Kompozicija (Composition)

Objekti mogu da interaguju na različite načine u nekom sistemu. U nekim slučajevima klase i objekti mogu biti tako međusobno povezani da svi zajedno čine kompleksan sistem. U primeru sa automobilom točkovi, paneli, motor, menjač itd. Mogu biti posmatrani kao posebne klase. Klasa automobil, u tom slučaju, predstavlja kompoziciju ovih posebnih klasa.

Nasleđivanje (Inheritance)

Nasleđivanje omogućava da neka klasa (subklasa) bude bazirana na drugoj klasi (super klasi) i da od nje nasledi svu funkcionalnost. Kroz dodatni kod (atribute metode) sub klasa može biti specijalizovana za posebne potrebe. Na primer, od klase vozila (kao super klase) možemo kreirati subklase automobili i motorcikli. Obe ove klase nasledile bi sve metode klase vozila, ali bi takođe mogle da imaju i svoje specijalizovane metode i atribute, kao što su metoda Nagnise() i atribut Ugaonaginjanja za motorcikle.

Polimorfizam (Polymorphism)

Polimorfizam je sposobnost objekta da menja ponašanje u zavisnosti od načina na koji se koristi. Polimorfizam u najjednostavnijem obliku se pojavljuje kao niz metoda sa istim imenom ali sa različitim parametrima. Ovaj oblik polimorfizma naziva se “overloding” (overloading), a sreće se i kod operatora . U zavisnosti od parametara koje koristimo biće pozvan onaj metod koji odgovara korišćenim parametrima. Polimorfizam ima i druge oblike o kojima ovde neće biti reči.

Modularnost (Modularity)

Pored gore navedenih koncepata OO programiranje donosi i povećanje modularnosti programa. Individualne klase ili grupe povezanih klasa (biblioteke klasa) mogu se posmatrati kao moduli koji se mogu koristiti u raznim softverskim projektima. Na taj način se smanjuje potreba za ponovnim programiranjem sličnih zadataka, pa se tako redukuje ukupan napor potreban za razvoj sofvera.

Pitanja

1. Šta je klasa?
2. Šta je objekat?
3. Koje su osnovne komponente klase?
4. Šta je apstarkcija?
5. Šta je učaurivanje?
6. Šta je nasleđivanje? Primer.
7. Šta je polimorfizam? Primer.
8. Šta je overloding metoda? Primer.
9. Šta je overloding operatora? Primer.
10. Koje su prednosti korišćenja OO metoda?
11. Kako se postiže modularnost u OO programiranju?