OSNOVE OBJEKTNO USMJERENOG PROGRAMIRANJA

OBJEKTI I KLASE

- što vjernije opisati i modelirati stvarni svijet iz kojeg problem potječe
- objekti u stvarnom životu obično predstavljaju nešto konkretno i opipljivo
- Primjer:
  - izdavanje računa u trgovini
    - broji svaki prodani proizvod i dojavljuje kada treba obnoviti zalihe
    - proizvodi su zapravo objekti nad kojima treba obaviti neke aktivnosti
  - škola – učenici
  - banke – korisnici, računi...
- kod programiranja objekata postoje atributi koji opisuju objekt određenim podatcima i metode koje djeluju na te atribute
- objekti koji se neznatno razlikuju mogu se svrstati u tzv. klase
- klasa je nacrt za kreiranje objekta

KORACI KOD RJEŠAVANJA PROBLEMA:

- Objektno usmjerena analiza – što želimo, koji su objekti i kako oni međusobno djeluju
- Objektno usmjeren dizajn – osmisliti izgled i strukturu programskog rješenje, odrediti objekte njihove atribute, metode i svrstati ih u klase
- Objektno usmjereno programiranje – konkretno programsko rješenje

„PROJEKT” - KASNIJE

SVI TIPOVI PODATAKA SU KLASE

- I za stringove i liste tj. za **sve tipove** podataka možemo reći da su klase – imaju atribute, metode...
- operatori se mogu napisati s pomoću metoda
  - operator + ostvaruje metodom __add__()
  - operator * metodom __mul__()

DEFINIRANJE KLASE

```
class ImeKlase:
    definicija klase
```

- naziv klase piše se velikim početnim slovom (ako je više riječi - bez razmaka, velikim početnim slovom svake riječi – npr. PrvaKlasa)
- definicije njenih metoda i atributa
- najvažnija metoda je **konstruktor** (kreira objekt)
  - **ime __init__(self, parametri)**
  - postavljanje atributa (svojstava) na neku vrijednost
  - atributima možemo pristupiti iz svih metoda
  - **self.ime_atributa**
- **u svim metodama mora se pojaviti parametar self** kao prvi parametar je on predstavlja pojedini object
- metode definiramo na isti način kao i funkcije

PRIMJER 1. - KREIRANJE OSNOVNE KLASE

Kreirajte klasu kvadrat koja će kao atribut primiti duljinu stranice kvadrata te imati metode za račununanje opsega i površine kvadrata.

Samostalno dodajte metodu koja računa duljinu dijagonale kvadrata.

Klasu i metode pozivajte u glavnom programu.

RJEŠENJE 1. PRIMJERA

```
from math import*

class Kvadrat:
    def __init__(self, a=0): #ima parametar a s inicaijalnom vrijednosti 0
        self.stranica = a

    def opseg(self):
        return 4*self.stranica

    def povrsina(self):
        return self.stranica ** 2

    def dijagonala(self):
        return self.stranica*sqrt(2)

def main():
    a = Kvadrat(int(input()))
    print(a.opseg())
    print(a.povrsina())
    print(a.dijagonala())

main()
```

RJEŠENJE 1. PRIMJERA

- metode za opseg, površinu, dijagonalu nemaju parametar a (stranicu kvadrata)
  - razlikuje od strukturnog programiranja jer pri pozivu funkcije najčešće moramo staviti parametre
  - parametri su spremljeni unutar klase
- metodama, ali i **atributima** možemo pristupiti izvan klase tako da iza imena objekta stavimo točku npr. a.stranica = 4 (promijenit će stranicu s 5 na 4)

PRIMJER 2. - KLASA OSOBA

Kreirajte klasu osoba koja će kao atribut primiti ime, prezime i broj godina neke osobe te imati metodu pomoću koje se osoba pozdravlja i metodu pomoću koje se osoba ukratko predstavlja

npr.

Pozdrav!

Ja sam Ivo Ivić, imam 18 godina.

RJEŠENJE 2. PRIMJERA

```
class Osoba:
    def __init__(self, ime = "", prezime = "", godine = 0):
        self.ime = ime
        self.prezime = prezime
        self.godine = godine


    def pozdrav(self):
        print("Dobar dan!")

    def pred(self):
        print("Moje ime je ", self.ime, self.prezime, ".Imam", self.godine, "godina")

def main():
    a = Osoba("Marko", "Ivić", 10)
    a.pozdrav()
    a.pred()

main()
```

METODE S DODATNIM PARAMETRIMA

- metode mogu imati parametre koji nisu atributi klase
- vrijednosti tih parametara šaljemo direktno metodama i vidljivi su samo u toj metodi

PRIMJER 3. - DODATNI PARAMETRI

U klasi osoba definirajte metodu ručak koja će primati jedan string u kojem je navedeno što je određena osoba ručala i to ispisati.

npr.

Ivo: Za ručak bilo je juhe od rajčice i čevapi.

RJEŠENJE 3. PRIMJERA

```
    def rucak(self, s):
        print(self.ime, ":Za ručak bilo je", s)

def main():
    a = Osoba(input(), input(), int(input()))
    a.pozdrav()
    a.pred()
    a.rucak(input())

main()
```

OBJEKTI KAO PARAMETRI METODA I SPECIJALNE METODE

- metode mogu imati parametre koji su i sami objekti te iste klase
- za određene metode pomoću standardnih specijaliziranih metoda možemo definirati drugačiji poziv samih metoda
- __str__ →tekstualni prikaz objekta kakav će se ispisivati pozivom tog objekta unutar funkcije print()
- __repr__ → prikaz objekta kakav će se ispisati u interaktivnom sučelju

naziv metode	opis
`__add__(self, b)`	zbrajanje (+)
`__sub__(self, b)`	oduzimanje (-)
`__mul__(self, b)`	množenje (*)
`__floordiv__(self, b)`	cjelobrojno dijeljenje (//)
`__truediv__(self, b)`	dijeljenje (/)
`__mod__(self, b)`	ostatak cjelobrojnog dijeljenja (%)
`__pow__(self, n)`	potenciranje (**)
`__lt__(self, b)`	manje (<)
`__le__(self, b)`	manje ili jednako (<=)
`__gt__(self, b)`	veće (>)
`__ge__(self, b)`	veće ili jednako (>=)
`__eq__(self, b)`	jednako (==)

PRIMJER 4.

Kreirajte klasu razlomak koja će kreirati razlomak te definirajte metodu za ispis razlomka, metodu za zbrajanje dva razlomka i metodu za umnožak dvaju razlomka. Na kraju ćemo kreirati metodu skrati koja će neki razlomak skratiti do kraja.

RJEŠENJE 4. PRIMJERA

```
class Razlomak:
    def __init__(self, b = 0, n = 1):
        self.b = b
        self.n = n

    def __add__(self, r):
        if self.n == r.n:
            b = self.b+r.b
            n = self.n
        else:
            b = self.b*r.n+r.b*self.n
            n = self.n*r.n
        zb = Razlomak(b, n)
        return zb

    def __mul__(self, r):
            b = self.b*r.b
            n = self.n*r.n
            um = Razlomak(b, n)
            return um
```

RJEŠENJE 4. PRIMJERA

```
    def __str__(self):
        if self.n == 1:
            return str(self.b)
        else:
            return str(self.b) + "/" + str(self.n)
    def __repr__(self):
        return self.__str__()

def main():
    a = Razlomak(2, 3)
    print(a)
    b = Razlomak(5, 3)
    print(a+b)
    print(a*b)

main()
```

ZADACI ZA VJEŽBU

ZADATAK 1.

Kreirajte klasu pas koja će kao atribute imati vrstu psa, boju, broj godina te ime psa. Kreirajte dvije metode koje opsiju neke pseće radnje npr. jedna neka bude pas laje, a drugu sami osmislite. Kreirajte neke objekte u glavnom programu i pozovite izrađene metode.

ZADATAK 2.

Kreirajte klasu pravokutnik koja će kao atribut primiti duljine stranica pravokutnika te imati metode za račununanje opsega, površine i duljine dijagonale tog pravokutnika.

Klasu i metode pozivajte u glavnom programu.

ZADATAK 3.

Kreirajte samostalno klasu za stringove u kojoj će se nalaziti barem 5 metoda za stringove koje često koristite, ali ih kreirajte na svoj način bez upotrebe gotovih metoda. Npr. imamo metodu s.upper() koja vraća string s velikim slovima kreirajte metodu s.velika() koja će raditi istu stvar, ali bez upotrebe metode upper.