Programovací paradigma = **obecný model programování** na kterém je jazyk založen

Typy paradigma:

- **Procedurální:**
  - Procedura = **podprogram** (funkce), může se opakovaně volat
  - Program se skládá z procedur kde každá vykoná část algoritmu
  - **Píšeme přesný postup** jakým se program řídí (program jede odshora dolů, co přečte to udělá)
  - Procedury se mohou volat navzájem
  - Čistě procedurální jazyk = **Basic, C**

- **Objektové:**
  - Objekt obsahuje **atributy + metody** (data + funkce)
  - Program se skládá z objektů které dokážeme číst a měnit
  - Metody mění pouze data stejného objektu
  - Objekty mezi sebou komunikují posíláním zpráv -> říkají si co mají dělat
  - Čistě objektový jazyk = **SmallTalk, Ruby**

- **Grafické** -> místo psaní kódu spojujeme bloky programu (Scratch, Node-RED)
- **Ezoterické** -> v praxi se nepoužívají, pouze jako žert (Brainfuck, OstraJAVA)

Různé jazyky **podporují jedno nebo více** -> multiparadigmatický programovací jazyk (C++, C#, Python, Java)

Rozdíly:

Využití

- **POP** -> psaní přesně zadaných funkcí (algoritmy, knihovny)
- **OOP** -> ideální pro systémy, které budeme rozšiřovat ale zatím nevíme jak (aplikace, hry)

Rozsah

- **POP** -> s velkým rozsahem nepřehledné, přidání něčeho nového ovlivní celý program (**špagetový kód**)
- **OOP** -> snadné přidání nových věcí, programy zabírají většinou více místa

Znovupoužitelnost

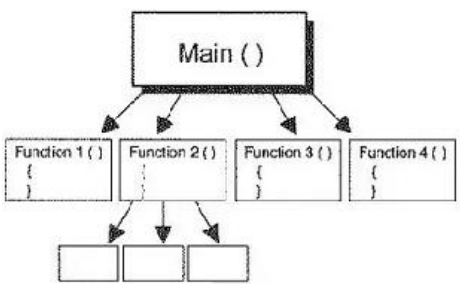
- **POP** -> jelikož píšeme sekvenci, kód musíme upravovat abychom ho mohli použít jinde
- **OOP** -> pokud umíme napsat obecný objekt, můžeme ho používat kdekoliv

Přístup a ochrana dat

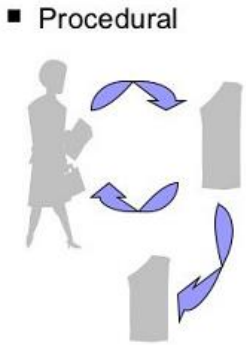
- **POP** -> s daty se pracuje přímo v kódu, přístup není omezen
- **OOP** -> data jsou zapouzdřena v objektu, přístup můžeme omezit (private, public, protected)

Návrh

- !!! V praxi se žádný program **nevymyslí čistě jednou metodou**, návrhy kombinujeme
- **POP** -> rozdělujeme si jednu velkou úlohu na několik menších (**top-bottom approach**)
- **OOP** -> vymýšlíme objekty které nakonec pospojujeme pro vyřešení problému (**bottom-up approach**)

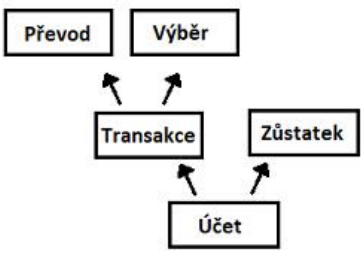


Top-bottom



Withdraw, deposit, transfer

Customer, money, account



Bottom-up

Návrh objektů:

Třída -> **šablona objektu**, definuje atributy/metody společné pro objekty, neznáme konkrétní hodnoty
Objekt = instance -> **konkrétní výskyt třídy**, nezávislé na ostatních, zadáváme hodnoty do šablony
Atribut -> uložená data objektu, hodnoty si pamatuje, můžou být různé u různých objektů
Metoda -> funkce které objekt vykonává, při volání ji provede příslušný objekt

```
1
2  class printMachine():                    # TŘÍDA
3      def __init__(self, pozdrav):
4          self.ulozenyPozdrav = pozdrav    # ATRIBUT
5
6      def printPozdrav(self):              # METODA
7          print(self.ulozenyPozdrav)
8
9  objektCau = printMachine("cau") # OBJEKT 1
10 objektCus = printMachine("cus") # OBJEKT 2
11
12 objektCau.printPozdrav() # VOLÁNÍ METODY OBJEKTU 1
13
```

4 pilíře OOP:

Abstrakce

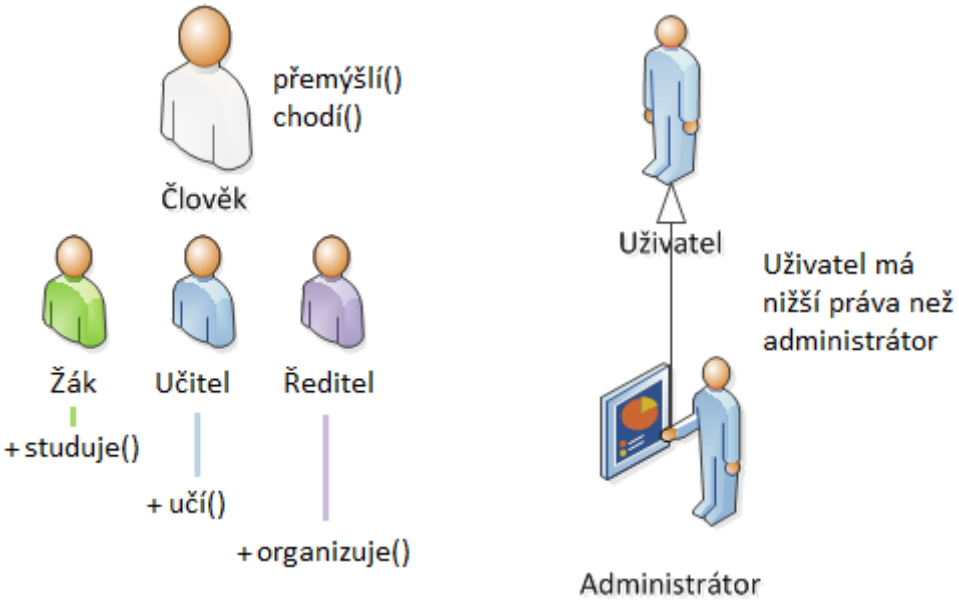
- Skrytí nepodstatných detailů ve skupině objektů
- **Příklad:** dveře otevíráme pořád stejně, je jedno jestli jsou dřevěné nebo laminované

Zapouzdření

- Sdružení funkcí a dat do jedné entity, k obsahu objektu se nedostane nikdo jiný než vlastník
- Každý objekt jde spravovat bez toho, aby ovlivnil ostatní objekty nebo funkčnost programu

Dědičnost

- Vytvoření podtřídy která používá atributy/metody vyšší třídy + ještě může používat své vlastní (**Člověk**)
- Vyšší třída může skrýt některé atributy/metody aby je mohla používat sama (**Uživatel**)



Polymorfismus

- Stejný název metody, kterou má jednotlivá nižší třída, může vykonávat různé věci

```python

class __Lod:
    def __init__(self, barva):
        self.barva = barva; # Atribut stejný pro všechny třídy

    def vypisInfoLod(self): # Metoda stejná pro všechny třídy
        print(f"Barva = {self.barva}")

class Ledoborec(__Lod):     # Třída která dědí z vyšší třídy
    def jakPlave(self):     # Polymofismus metody jakPlave()
        print("Borti ledy")

class Kanoe(__Lod):         # Třída která dědí z vyšší třídy
    def jakPlave(self):     # Polymofismus metody jakPlave()
        print("Musi veslovat")

ledoborec = Ledoborec("cervena")    # Objekt třídy Ledoborec
kanoe = Kanoe("zelena")             # Objekt třídy Kanoe

ledoborec.vypisInfoLod()    # Volá obecnou metodu
ledoborec.jakPlave()        # Volá metodu konkrétní pro Ledoborec

kanoe.vypisInfoLod()        # Volá obecnou metodu
kanoe.jakPlave()            # Volá metodu konkrétní pro Kanoe


```