Enkapsulácia

Dôležitá súčasť **OOP**

- **Enkapsulácia** znamená zapuzdrenie atribútov alebo metód v tele triedy pred vonkajším svetom (private variable, private function), tzn. nedostupnosť takýchto atribútov (premenných) a funkcií mimo triedy, v ktorej sú definované
- Zabraňuje tomu, aby používateľ objektu vedel dostať tento objekt do nevalidného stavu
- Schovávame implementačné detaily

Príklad auto

- Bez enkapsulácie vieme zabočiť ľavým kolesom doprava a pravým doľava – auto pokazíme
- Namiesto toho kolesá schováme pred verejnosťou (private) a spravíme verejnú metódu zaboč()
- Auto interne vie (alebo ten kto ju vytváral), že auto zabáča vždy oboma kolesami
- Používateľ nemusí vôbec vedieť ako auto zabáča, dôležité je, že zabáča

Access modifiers

- Týkajú sa atribútov (premenných) a metód
- **Private** (súkromná) – iba daná trieda vie k atribútom a metódam pristupovať
- **Protected** (chránená) – daná trieda a jej deti - potomkovia (triedy od nej dediace) vedia k vlastnostiam a k metódam pristupovať
- **Public** (verejná) – dostupné pre všetkých vrátane mimo tela triedy

Prečo nemôže byť všetko public

- Menenie vlastností môže mať veľa vedľajších efektov, ktoré vieme ošetriť/zabezpečiť v setteri
- Vďaka enkapsulácii vieme spraviť read-only alebo write-only atribúty (premenné)
- Enkapsulácia zjednodušuje testovanie

Enkapsulácia

- Pristúpiť k premenným vieme cez metódy **set()** - nastaví hodnotu premennej a **get()** – vráti hodnotu premennej
- get() a set() môžu chrániť stav objektu pred možným zneužitím, napr. ak v class premennej očakávame int, dostaneme string
- V Pythone **nie je typická** enkapsulácia zabraňujúca prístup k premenným, k tzv. **private** variables, ale vieme k premenným pristupovať pomocou enkapsulácie
- Skryté premenné a skryté metódy sa v Pythone označujú __example_var alebo __method() (použitím dvoch znakov podtržníka "_")
- Ku skrytým premenným pristupujeme metódami get() a set() alebo: viď príklad
- Pri dedení sa v potomkovi zdedia všetky skryté premenné, metódy z rodičovskej triedy a pristupuje sa k ním rovnako v zmysle dedenia: inštanciaPotomka._TriedaRodiča__názovPremennej alebo metódy

Príklad vráti:

1901 West Madison Street

Different address

United Center

Chicago

15000

```python
class Stadium:
    capacity = 15000
    __name = "United Center " # skrytie premennej

    def __init__(self, a, b):
        self.__a = a # skrytie premennej
        self.__b = b

    def __print_capacity(self): # skrytie metódy
        print(Stadium.capacity)

    def get_a(self):
        return self.__a

    def set_a(self, value):
        self.__a = value

    def print_address(self):
        print(f"Address: {self.__a} {self.__b}")

arena = Stadium("1901 West Madison Street", "Chicago")
# Výpis vráti chybu,
# nevieme pristúpiť k _name, rovnako ani k __a, __b
print(Stadium.__name)

# Prístup 1 k skrytým premenným
print(arena.get_a())
arena.set_a("Different address")
print(arena.get_a())

# Prístup 2 k skrytým premenným a k skrytej metóde
print(arena._Stadium__name)
print(arena._Stadium__b)
arena._Stadium__print_capacity()
```