Autopůjčovna v Pythonu

Semestrální projekt z Objektového modelování, ČZU

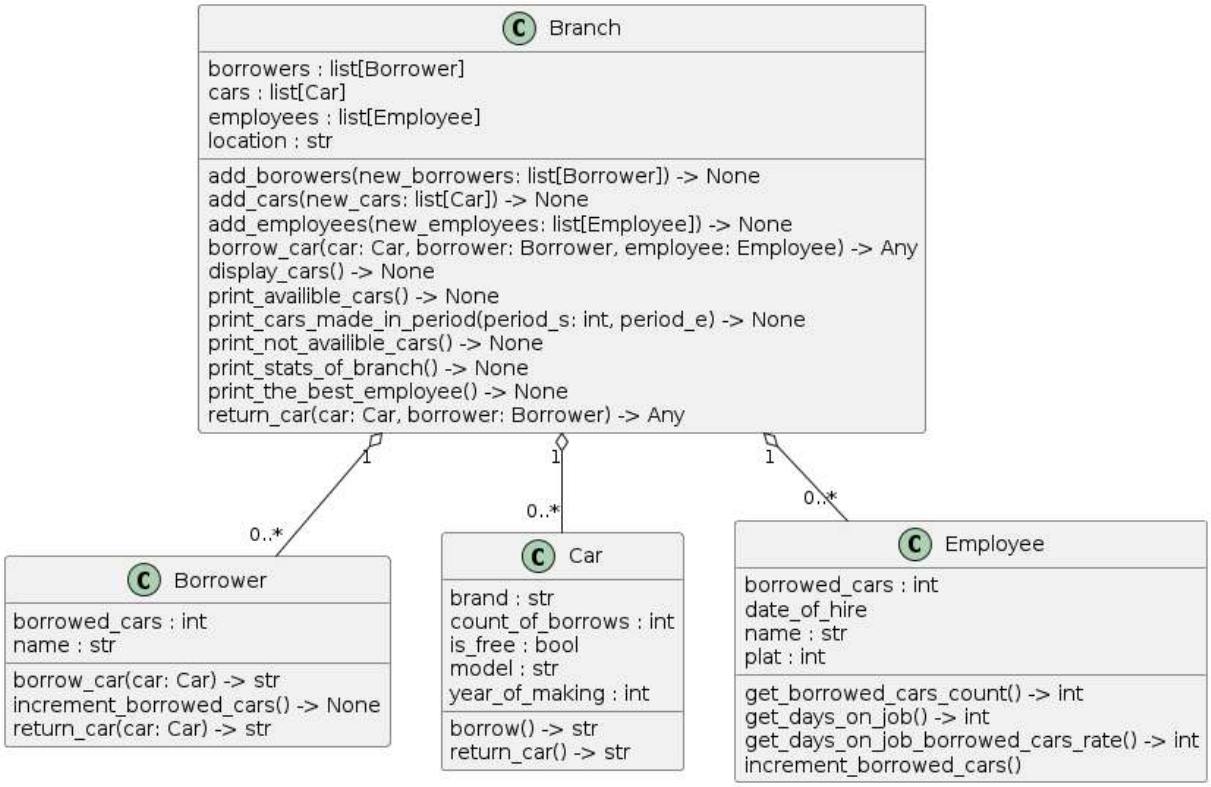
Autor: **Jakub Matoušek**

Popis projektu

V projektu jsem se snažil vytvořit jednoduchý systém pro správu autopůjčovny v programovacím jazyce Python. Systém umožňuje správu vozidel, zákazníků a zaměstnanců půjčovny. Program je rozdělen do 4 tříd, kde každá plní svou klíčovou funkci. Částečná dokumentace tříd a metod se nazchází přímo v kódu. Místy ji zde rozšiřuji. Systém je schopný mimo jiné generování reportů o nejlepším zaměstnanci pobočky a vypsání pouze dostupných / nedostupných aut. Generování reportů je možné díky faktu, že si každá z agregovaných instancí (Car, Borrower, Employee) v třídě „Branch“ ukládá počet uskutečněných výpujček. Systém si hlídá, aby nebylo možné například půjčit již vypůjčené auto, či vypůjčit auto, které není vedené na pobočce, nebo připsat zprostředkování výpůjčky zaměstnanci, kterého pobočka nezná.

Celý systém je navržen s ohledem na objektově orientované programování, což umožňuje snadnou rozšiřitelnost a údržbu kódu.

UML CLASS DIAGRAM



Třída – Branch (pobočka)

```python
class Branch:
    """
    Třída reprezentující pobočku půjčovny aut. Ukládá auta, zákazníky a zaměstnance.
    Při inicializaci nemá žádné auta, zákazníky ani zaměstnance. Název pobočky je povinný parametr.
    Přes třídu lze uskutečnit půjčení auta, vrácení auta, přidání aut, zákazníků a zaměstnanců.
    """

    def __init__(self, location: str) -> 'Branch':
        self.location = location
        self.cars: list[Car] = []
        self.borrowers: list[Borrower] = []
        self.employees:list[Employee] = []

    def __str__(self) -> str:
        """Vrátí název pobočky."""

        return f"Branch {self.location}"

    def add_cars(self, new_cars: list[Car]) -> None:
        """Přidá auta do pobočky. Nemá návratovou hodnotu."""

        self.cars = self.cars + new_cars

    def add_borowers(self, new_borrowers: list[Borrower]) -> None:
        """Přidá zákazníky do pobočky. Nemá návratovou hodnotu."""

        self.borrowers = self.borrowers + new_borrowers

    def borrow_car(self, car: Car, borrower: Borrower, employee: Employee = None) -> Any:
        """Půjčí auto zpět z pobočky.
           Vrací False pokud auto není v pobočce, jinak vrátí objekt vráceného auta.
           Může přijmout i zaměstnance, který půjčení zprostředkuje."""

        if car in self.cars:
            if employee not in self.employees:
                return "We dont know this employee at this branch."
            employee.increment_borrowed_cars()
            return borrower.borrow_car(car)
        print(f"{car.brand} {car.model} {car.year_of_making} is not available at this branch.")
        return False
```

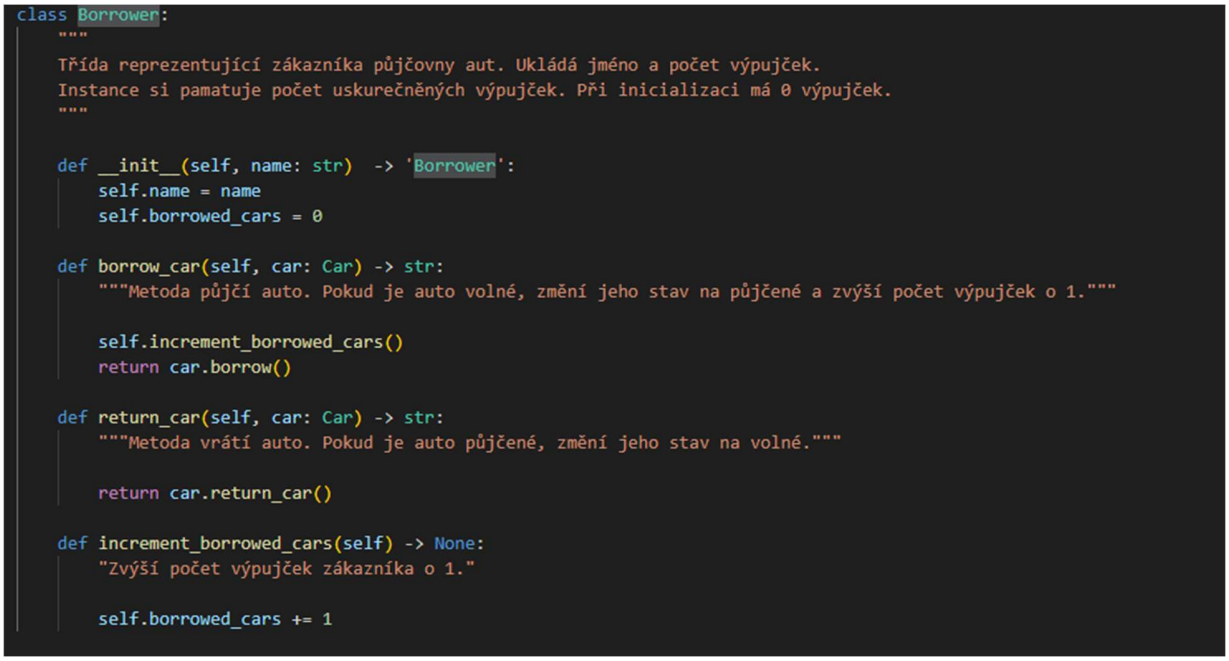
```python
def return_car(self, car: Car, borrower: Borrower) -> Any:
    """Vrátí auto zpět do pobočky.
       Vrací False pokud auto není v pobočce,
       jinak vrátí objekt vráceného auta"""

    if car in self.cars:
        return borrower.return_car(car)
    print(f"{car.brand} {car.model} {car.year_of_making} is not available at this branch.")
    return False

def add_employees(self, new_employees: list[Employee]) -> None:
    "Přidá zaměstnance do pobočky. Nemá návratovou hodnotu."

    self.employees += new_employees

def display_cars(self) -> None:
    "Vypíše všechna auta pobočky. Nemá návratovou hodnotu."

    print(f"Cars available at {self.location}:")
    for car in self.cars:
        print(car)

def print_availible_cars(self) -> None:
    "Vypíše nedostupná auta. Nemá návratovou hodnotu."

    print(f"Cars available at {self.location}:")
    for car in self.cars:
        if car.is_free:
            print(car)

def print_not_availible_cars(self) -> None:
    "Vypíše volná auta. Nemá návratovou hodnotu."

    print(f"Cars not available at {self.location}:")
    for car in self.cars:
        if not car.is_free:
            print(car)

def print_cars_made_in_period(self, period_s: int, period_e=None) -> None:
    "Metoda vypíše auta vyrobená v zadaném období let (včetně)"

    if period_e is None:
        period_e = datetime.datetime.now().year
    if period_s > period_e:
        period_s, period_e = period_e, period_s
    print(f"Cars made between {period_s} and {period_e}:")
```

```python
def print_stats_of_branch(self) -> None:
    "Metoda vypíše statistiky pobočky - počet zákazníků a počet vypůjčení aut."

    borrowings = 0
    for car in self.cars:
        borrowings += car.count_of_borrows
    print(f"{self.__str__()} has {len(self.borrowers)} customers. \nBranch has borrowed a car {borrowings}x times")

def print_the_best_employee(self) -> None:
    "Metoda vypíše zaměstnance s nejlepším poměrem dní v práci / výpujček."

    if len(self.employees) == 0:
        print("No employees in the branch.")
        return
    best_emp_rate = self.employees[0].get_days_on_job_borrowed_cars_rate()
    best_emp = None
    for e in self.employees:
        rate  = e.get_days_on_job_borrowed_cars_rate()
        if rate and rate < best_emp_rate:
            best_emp_rate = rate
            best_emp = e
    print(f"""Best employee is {best_emp.__str__()}
            |-> with rate of {best_emp_rate} days_on_job/borrowed_car
            |-> has borrowed {best_emp.get_borrowed_cars()} cars
            """)
```

Třída – Employee (zaměstnanec)

```python
class Employee:
    """
    Třída reprezentující zaměstnance půjčovny aut. Ukládá jméno, datum nástupu a plat.

    Tato třída také počítá následující statistiky o zaměstnanci:
    - Dní v práci / poměr výpujček
    - Počet dní v práci
    """
    def __init__(self, name: str, date_of_hire: datetime, plat: int) -> 'Employee':
        self.name = name
        self.date_of_hire = date_of_hire
        self.plat = plat
        self.borrowed_cars = 0

    def increment_borrowed_cars(self):
        "Zvýší počet výpujček  zaměstnance o 1."
        self.borrowed_cars += 1

    def get_days_on_job_borrowed_cars_rate(self) -> int:
        "dní v práci / poměr výpujček"
        days_on_job = self.get_get_days_on_job()
        if self.borrowed_cars == 0 or days_on_job == 0:
            return None
        return int(days_on_job / self.borrowed_cars)

    def get_days_on_job(self) -> int:
        "počet dní v práci jako integer"
        return (datetime.datetime.now() - self.date_of_hire).days

    def __str__(self) -> str:
        "Vrátí jméno, plat a datum nástupu zaměstnance jako string."
        return f"Employee {self.name} - pay: {self.plat}kč - date of hire: {self.date_of_hire}"

    def get_borrowed_cars_count(self) -> int:
        """Vrátí počet vypůjčených aut zaměstnancem."""
        return self.borrowed_cars
```

Třída – Borrower (klient autopůjčovny)



```python
class Borrower:
    """
    Třída reprezentující zákazníka půjčovny aut. Ukládá jméno a počet výpujček.
    Instance si pamatuje počet uskurečněných výpujček. Při inicializaci má 0 výpujček.
    """

    def __init__(self, name: str)  -> 'Borrower':
        self.name = name
        self.borrowed_cars = 0

    def borrow_car(self, car: Car) -> str:
        """Metoda půjčí auto. Pokud je auto volné, změní jeho stav na půjčené a zvýší počet výpujček o 1."""

        self.increment_borrowed_cars()
        return car.borrow()

    def return_car(self, car: Car) -> str:
        """Metoda vrátí auto. Pokud je auto půjčené, změní jeho stav na volné."""

        return car.return_car()

    def increment_borrowed_cars(self) -> None:
        "Zvýší počet výpujček zákazníka o 1."

        self.borrowed_cars += 1
```

Třída – Car (auto)

```python
class Car:
    """Třída reprezentující auto. Ukládá značku, počet výpujček, model a rok výroby."""

    def __init__(self, brand: str, model: str, year:int) -> 'Car':
        self.brand = brand
        self.model = model
        self.year_of_making = year
        self.is_free = True
        self.count_of_borrows = 0

    def borrow(self) -> str:
        """Metoda půjčí auto. Pokud je auto volné, změní jeho stav na půjčené a zvýší počet výpujček o 1."""

        if self.is_free:
            self.is_free = False
            self.count_of_borrows += 1
            return f"{self.brand} {self.model} {self.year_of_making} has been borrowed."
        return f"{self.brand} {self.model} {self.year_of_making} is not available for borrowing."

    def return_car(self) -> str:
        """Metoda vrátí auto. Pokud je auto půjčené, změní jeho stav na volné."""

        if not self.is_free:
            self.is_free = True
            return f"{self.brand} {self.model} {self.year_of_making} has been returned."
        return f"{self.brand} {self.model} {self.year_of_making} is not currently borrowed."

    def __str__(self) -> str:
        """Vrátí značku, model, rok výroby a stav auta (volné/půjčené) jako string."""

        return f"{self.brand} {self.model} {self.year_of_making} - {'Free' if self.is_free else 'Borrowed'}"
```

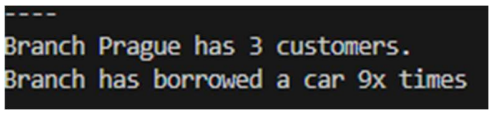
Použité knihovny

```python
import datetime # importování modulu datetime pro práci s datem a časem
from typing import Any # importování modulu Any pro možnost vrácení libovolného datoveho typu
```

Inizializační skript

```python
if __name__ == "__main__":
    # inicializační skript

    # inicializace aut
    car1 = Car("Škoda", "Fabia 1.0", 2020)
    car2 = Car("Honda", "Anhilator", 2019)
    car3 = Car("Ford", "Megasmoker", 2015)
    car4 = Car("Renault", "Turbo", 1991)

    # inicializace pobočky
    branch = Branch("Prague")
    branch.add_cars([car1, car2, car3, car4])

    # inicializace zákazníků
    b1 = Borrower("John")
    b2 = Borrower("Marie")
    b3 = Borrower("Honzik")

    branch.add_borowers([b1,b2,b3]) # přidání zákazníků do pobočky

    # inicializace zaměstnanců
    e1 = Employee("Petr Majer", datetime.datetime(2023, 1, 1), 30000)
    e2 = Employee("Honza Kvitko", datetime.datetime(2021, 1, 1), 40000)
    e3 = Employee("Jan Vyliž", datetime.datetime(2024, 1, 1), 18000)

    branch.add_employees([e1,e2,e3]) # přidání zaměstnanců do pobočky

    # půjčení a vrácení aut
    branch.borrow_car(car1, b1, e1)
    branch.return_car(car1, b1)
    branch.borrow_car(car1, b1, e2)
    branch.return_car(car1, b1)
    branch.borrow_car(car1, b1, e3)
    branch.return_car(car1, b1)
    branch.borrow_car(car1, b1, e1)
    branch.return_car(car1, b1)
    branch.borrow_car(car1, b3, e3)

    branch.borrow_car(car2, b1, e1)
    branch.return_car(car2, b1)
    branch.borrow_car(car2, b2, e3)
    branch.return_car(car2, b1)
    branch.borrow_car(car2, b1, e1)

    branch.borrow_car(car3, b2, e2)
    branch.return_car(car3, b2)

    print("----")
    branch.print_stats_of_branch() # vypíše data pobočky
    print("----")
    branch.print_the_best_employee() # vypíše nejlepšího zaměstnance
```

Výstup metody print_stats_of_branch třídy Branch



Výstup metody print_the_best_employee třídy Branch