تمرین OOP

تمرین OOP

در این پیاده‌سازی، یک **سیستم مدیریت پارکینگ هوشمند HamidParking** طراحی می‌کنیم که از اصول OOP (کپسوله‌سازی، وراثت، چندریختی، انتزاع، ترکیب) بهره می‌برد. هدف آن مدیریت وسایل نقلیه، جای پارک، رزرو، قیمت‌گذاری داینامیک، پرداخت و گزارش‌گیری است. در ادامه برای هر کلاس اصلی، متدها همراه با پارامترها، مقدار بازگشتی و شرح منطق داخلی آمده است.

۱. کلاس `Vehicle` و زیرکلاس‌ها

نمایش خودروی ورودی و تعیین اندازه جای پارک مورد نیاز

متد	پارامترها	بازگشتی	شرح منطق داخلی
`__init__(self, size: str, plate: str, entry_time: datetime = None)`	شماره پلاک، زمان ورود اختیاری	—	ذخیره پلاک؛ اگر زمان ورود داده نشده، زمان فعلی ثبت می‌شود.
`required_spot_size(self) -> str`	—	`str`	بازمی‌گرداند که این نوع خودرو به چه سایزی از جای پارک (Small/Medium/Large) نیاز دارد.
`can_fit_in_spot(self, spot: ParkingSpot) -> bool`	شیء `ParkingSpot`	Bool	بررسی می‌کند آیا اندازه خودرو با اندازه جای پارک مطابقت دارد و جای پارک آزاد است.
`calculate_parking_duration(self, exit_time: datetime) -> timedelta`	زمان خروج	`timedelta`	تفاضل زمان خروج و زمان ورود را محاسبه می‌کند.

زیرکلاس‌ها (`Car` , `Motorcycle` , `Truck`) متد `required_spot_size` را به‌صورت اختصاصی پیاده‌سازی می‌کنند.

۲. کلاس `ParkingSpot`

نمایش یک جای پارک منفرد

متد	پارامترها	بازگشتی	شرح منطق داخلی
`__init__(self, floor: int, number: int, size: str)`	طبقه، شماره، اندازه	—	مقداردهی اولیه ویژگی‌ها و وضعیت «خالی».
`is_available(self) -> bool`	—	Bool	برمی‌گرداند آیا جای پارک در حال حاضر آزاد است.
`assign_vehicle(self, vehicle: Vehicle) -> bool`	شیء `Vehicle`	Bool	اگر اندازه و وضعیت اجازه دهد، خودرو را اختصاص می‌دهد و وضعیت را «پر» می‌کند.
`remove_vehicle(self) -> None`	—	—	خودرو را آزاد می‌کند و وضعیت را «خالی» می‌کند.

۳. کلاس `Floor`

مدیریت مجموعه‌ای از جای پارک‌ها در یک طبقه

متد	پارامترها	بازگشتی	شرح منطق داخلی
`__init__(self, level: int, spots: List[ParkingSpot])`	شماره طبقه، فهرست جای پارک‌ها	—	ذخیره اطلاعات طبقه و جای پارک‌ها.
`find_available_spot(self, vehicle: Vehicle) -> Optional[ParkingSpot]`	شیء `Vehicle`	`ParkingSpot` یا None	در میان جای پارک‌ها، اولین جای مناسب و آزاد را پیدا و برمی‌گرداند.
`occupy_spot(self, spot: ParkingSpot, vehicle: Vehicle) -> bool`	جای پارک، خودرو	Bool	متد `assign_vehicle` را روی جای پارک فراخوانی می‌کند.
`vacate_spot(self, spot: ParkingSpot) -> None`	جای پارک	—	متد `remove_vehicle` را روی جای پارک اجرا

متد	پارامترها	بازگشتی	شرح منطق داخلی
			می‌کند.

۴. کلاس `Reservation`

ثبت و پیگیری رزروها

متد	پارامترها	بازشدی	شرح منطق داخلی
`__init__(self, customer: Customer, vehicle: Vehicle, spot: ParkingSpot, start: datetime, end: datetime)`	مشتری، خودرو، جای پارک، زمان شروع، زمان پایان	—	ایجاد شیء رزرو و علامت‌گذاری جای پارک به عنوان رزرو شده.
`cancel(self) -> None`	—	—	لغو رزرو و آزادسازی جای پارک.
`check_in(self, current_time: datetime) -> bool`	زمان فعلی	Bool	اگر زمان فعلی ≥ زمان شروع باشد، ورود را ثبت می‌کند.
`check_out(self, current_time: datetime) -> timedelta`	زمان فعلی	مدت زمان پارک	خروج را ثبت، جای پارک آزاد و مدت پارک را بازمی‌گرداند.

۵. رابط و Factory برای قیمت‌گذاری

انتزاع نرخ‌ها و تولید استراتژی مناسب

```
from abc import ABC, abstractmethod

class PricingStrategy(ABC):
    @abstractmethod
    def calculate_fee(self, reservation: Reservation) -> float:
        pass

class FixedRateStrategy(PricingStrategy):
```

```
    def __init__(self, rate_per_hour: float): ...
    def calculate_fee(self, reservation): ...

class PeakHourStrategy(PricingStrategy):
    def __init__(self, base_rate: float, peak_multiplier: float, peak_hou
    def calculate_fee(self, reservation): ...

class DemandBasedStrategy(PricingStrategy):
    def __init__(self, base_rate: float, parking_lot: ParkingLot): ...
    def calculate_fee(self, reservation): ...
```

متد Factory	پارامترها	بازگشتی	شرح منطق داخلی
`get_pricing_strategy(type: str) -> PricingStrategy`	نوع استراتژی (“fixed”/“peak”/“demand”)	شیء `PricingStrategy`	بر اساس نوع درخواست، نمونه مناسب را می‌سازد.

۶. رابط و کلاس‌های پرداخت

انتزاع روش‌های پرداخت

```
class PaymentMethod(ABC):
    @abstractmethod
    def process(self, amount: float) -> bool:
        pass

class CreditCardPayment(PaymentMethod):
    def process(self, amount): ...

class MobileWalletPayment(PaymentMethod):
    def process(self, amount): ...
```

```
class CashPayment(PaymentMethod):
    def process(self, amount): ...
```

متد	پارامترها	بازگشتی	شرح منطق داخلی
`process(self, amount: float) -> bool`	مبلغ	Bool	اجرای تراکنش متناسب با روش (مثلاً اعتبارسنجی کارت، کیف پول، یا ثبت نقدی).

۷. کلاس `ParkingLot` (Singleton)

مدیریت کلی سیستم

متد	پارامترها	بازشدی	شرح منطق داخلی
`get_instance()`	—	`ParkingLot`	اطمینان از اینکه تنها یک نمونه از پارکینگ ساخته شود.
`add_floor(self, floor: Floor) -> None`	طبقه	—	افزودن طبقه به پارکینگ.
`make_reservation(self, customer, vehicle, start, end) -> Reservation`	اطلاعات رزرو	`Reservation`	جستجوی جای مناسب، ساخت رزرو و ذخیره در لیست رزروها.
`calculate_fee(self, reservation) -> float`	رزرو	مبلغ	انتخاب استراتژی قیمت‌گذاری و محاسبه هزینه.
`process_payment(self, reservation, method: PaymentMethod) -> bool`	رزرو، روش پرداخت	Bool	محاسبه هزینه، فراخوانی `process` و ثبت وضعیت پرداخت.
`generate_report(self, format: str) -> str`	"csv"/"json"	متن خروجی	خروجی رزروها، درآمد و اشغال به فرمت خواسته‌شده.

۸. کلاس `Reporting`

تولید خروجی CSV و JSON

متد	پارامترها	بازگشتی	شرح منطق داخلی
`to_csv(self, reservations: list(Reservation)) -> str`	فهرست رزرو	متن CSV	تبدیل جزئیات رزرو به رشته CSV.
`to_json(self, reservations: List[Reservation]) -> str`	فهرست رزرو	متن JSON	سریال‌سازی رزروها به JSON.

خروجی‌های مورد انتظار

۱. فایل پایتون `hamid_parking.py` شامل تمام کلاس‌ها و رابط‌ها

۲. اسکریپت تست با `pytest` پوشش‌دهنده سناریوهای اصلی (امتیازی)

۳. فایل `README.md` با معماری، نمودار UML و راهنمای اجرا (امتیازی)