Neptun: YTAJDI

Name: Yusupov Boburjon

Project description

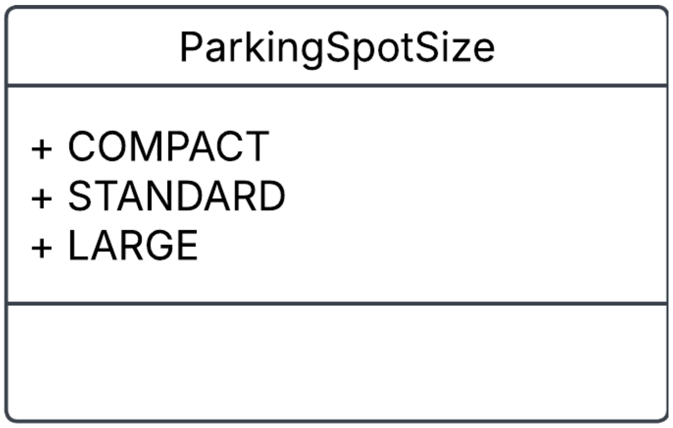
A parking lot manages a collection of parking spots, each available for short-term or long-term parking. Every parking spot has a spot number, a vehicle size limit (compact, standard, or large), an availability status indicating whether it is occupied or vacant, and an hourly parking rate.

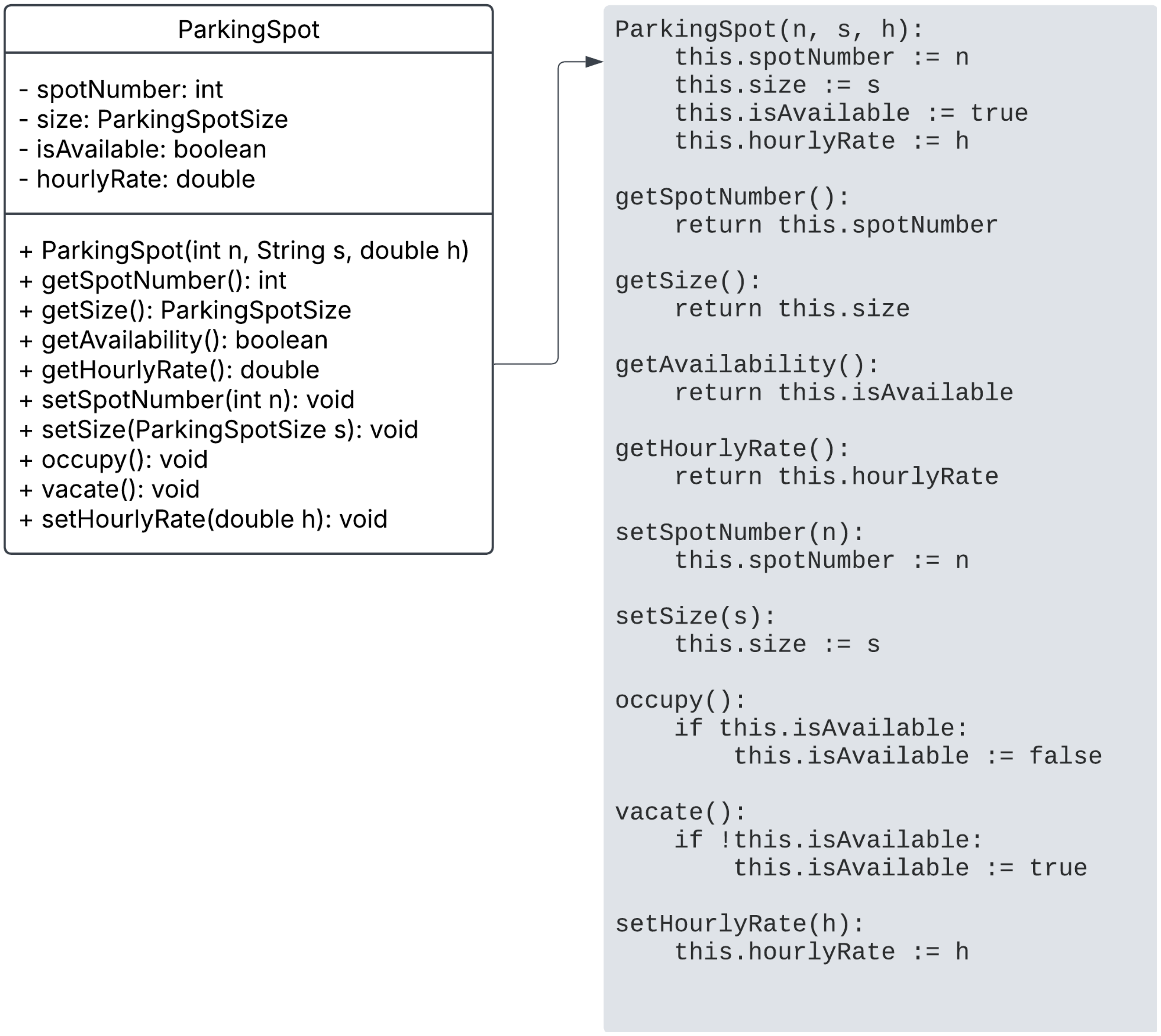
Drivers looking for parking spots may have different requirements. Some may need a large spot for a truck or SUV, while others may prioritize the cheapest available spot. Customers may also seek long-term parking arrangements for extended stays. If a vehicle is parked for more than 10 days, a 50% discount applies for every extra hour beyond the 10-day period to encourage long-term rentals.

To ensure efficient operations, the parking lot tracks active parking reservations, monitors occupancy rates, and calculates total revenue from parking fees. The system allows customers to search for available spots based on size and price restrictions, helping them quickly find a suitable space.

The parking lot also manages overstay penalties for vehicles that exceed their reserved time. If a driver does not remove their vehicle before the rental period expires, a penalty fee equal to twice the hourly rate per extra hour applies until the car is removed.

Diagram



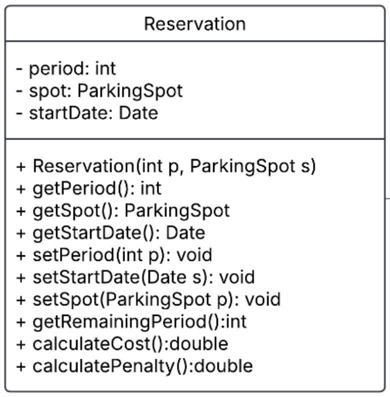


ParkingSpot

- spotNumber: int
- size: ParkingSpotSize
- isAvailable: boolean
- hourlyRate: double

+ ParkingSpot(int n, String s, double h)
+ getSpotNumber(): int
+ getSize(): ParkingSpotSize
+ getAvailability(): boolean
+ getHourlyRate(): double
+ setSpotNumber(int n): void
+ setSize(ParkingSpotSize s): void
+ occupy(): void
+ vacate(): void
+ setHourlyRate(double h): void

```
ParkingSpot(n, s, h):
    this.spotNumber := n
    this.size := s
    this.isAvailable := true
    this.hourlyRate := h

getSpotNumber():
    return this.spotNumber

getSize():
    return this.size

getAvailability():
    return this.isAvailable

getHourlyRate():
    return this.hourlyRate

setSpotNumber(n):
    this.spotNumber := n

setSize(s):
    this.size := s

occupy():
    if this.isAvailable:
        this.isAvailable := false

vacate():
    if !this.isAvailable:
        this.isAvailable := true

setHourlyRate(h):
    this.hourlyRate := h
```

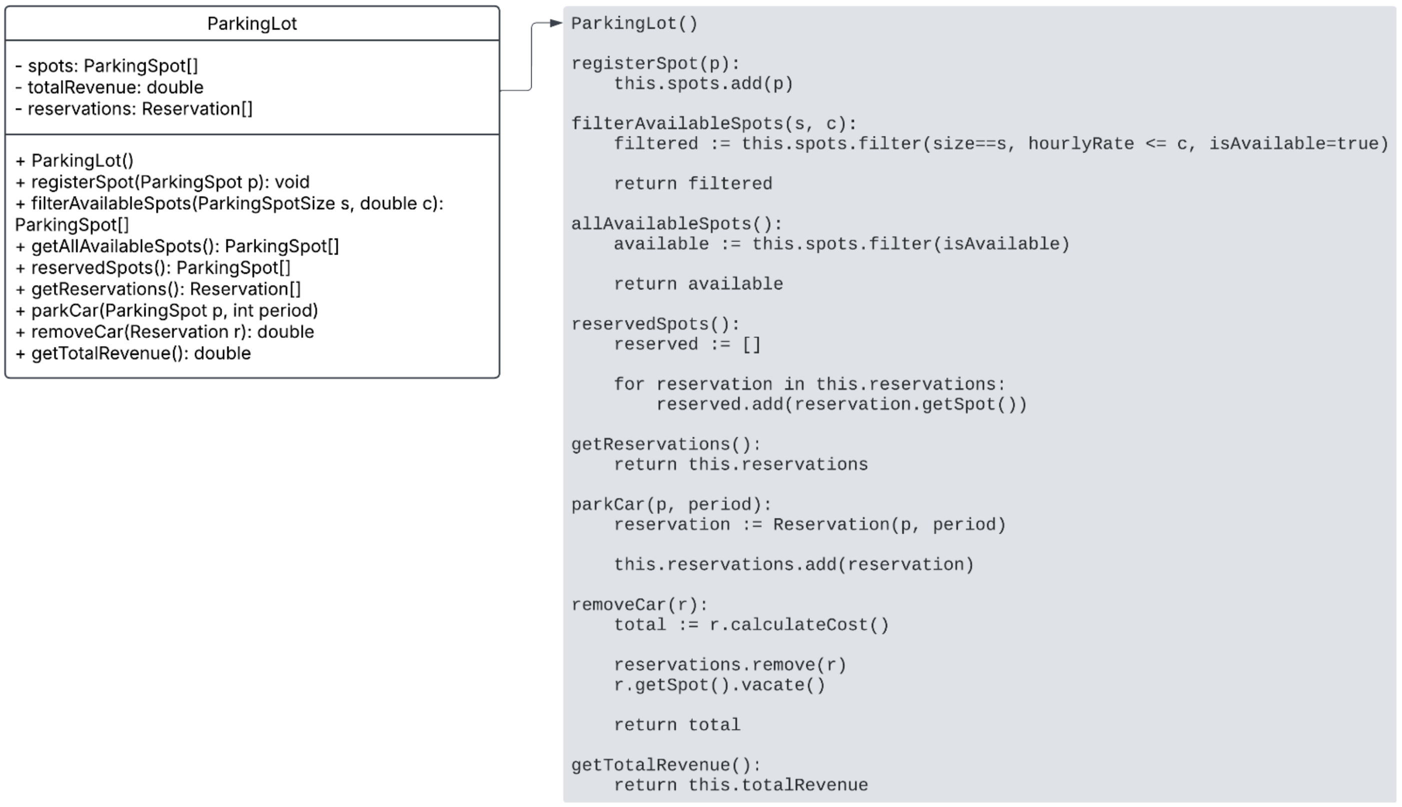


Reservation

- period: int
- spot: ParkingSpot
- startDate: Date

+ Reservation(int p, ParkingSpot s)
+ getPeriod(): int
+ getSpot(): ParkingSpot
+ getStartDate(): Date
+ setPeriod(int p): void
+ setStartDate(Date s): void
+ setSpot(ParkingSpot p): void
+ getRemainingPeriod():int
+ calculateCost():double
+ calculatePenalty():double

```
Reservation(p, s):
    this.period := p
    this.spot := s
    this.startDate := new Date()
    s.occupy()

getPeriod():
    return this.period

getSpot():
    return this.spot

getStartDate():
    return this.startDate

setPeriod(p):
    this.period := p

setStartDate(s):
    this.startDate := s

setSpot(p):
    this.spot = p

getRemainingPeriod():
    today := new Date()
    completed := (this.today.getTime() - this.startDate.getTime()) / 1000 / 60 / 60

    return completed

calculatePenalty():
    diff = System.currentTimeMillis() - this.startDate.getTime()
    diffHour := diff / 1000 / 60 / 60

    if (diffhour > period):
        return (diffHour - this.period) * this.spot.getHourlyCost() * 2

    else:
        return 0

calculateCost():
    penalty := calculatePenalty()

    if (period / 24) > 10:
        discount := (this.period - (10 * 24)) * this.spot.getHourlyRate() / 2
        actual := 10 * this.spot.getHourlyRate()

        return actual + discount

    else:
        return this.period * this.spot.getHourlyRate()
```



```
ParkingLot
- spots: ParkingSpot[]
- totalRevenue: double
- reservations: Reservation[]

+ ParkingLot()
+ registerSpot(ParkingSpot p): void
+ filterAvailableSpots(ParkingSpotSize s, double c):
ParkingSpot[]
+ getAllAvailableSpots(): ParkingSpot[]
+ reservedSpots(): ParkingSpot[]
+ getReservations(): Reservation[]
+ parkCar(ParkingSpot p, int period)
+ removeCar(Reservation r): double
+ getTotalRevenue(): double
```

```
ParkingLot()

registerSpot(p):
    this.spots.add(p)

filterAvailableSpots(s, c):
    filtered := this.spots.filter(size==s, hourlyRate <= c, isAvailable=true)

    return filtered

allAvailableSpots():
    available := this.spots.filter(isAvailable)

    return available

reservedSpots():
    reserved := []

    for reservation in this.reservations:
        reserved.add(reservation.getSpot())

getReservations():
    return this.reservations

parkCar(p, period):
    reservation := Reservation(p, period)

    this.reservations.add(reservation)

removeCar(r):
    total := r.calculateCost()

    reservations.remove(r)
    r.getSpot().vacate()

    return total

getTotalRevenue():
    return this.totalRevenue
```