Python Presentation

Md.Nakib Nasrullah

ID : 202110125

Course code : CSE 2312

Course Title : Object-Oriented Programing Sessional.

Teacher Name :Shamimul Islam Shamim.

(Lecturer, Department of CSE, AUB)

Topic

- Vehicle parking management system.

INTRODUCTION

What's Parking Management System ?

- *It help people find parking spots quickly.*
- *Parking management system for managing the records of the incoming and outgoing vehicles.*
- Vehicles in an parking house.
- It will determine the cost of per vehicle according to their type.
- It fixes parking related complications.

Objective

- In other words we can say that this project has the following objectives : -

- No paperwork requirement.
- Reduce time consumption.
- Easy operations for operator of the system.
- Maintain records in short time in period.

PLATFORM

- This project is developed using the tools, which are most suitable for development of an Application.

1. Visual Studio code.
2. Local server like as : Chrome Browser.

- My project is developed in Python programing language.

Python

- Python is an object oriented programing language.
- Almost everything is Python is an object, with its properties and method.
- Python is a dynamic, high-level, free open source, and interpreted programing language.
- High-Level programing language so easy to learn and human readable.

Code component

- Oop concept
- Class
- Inheritance
- Subclass
- Method
- Attributes

Class

Vehicle class : It represents a base class representing a generic vehicle.

- It has an _init_ method to initialize the registration number (reg_num) attribute.The get_vehicle_type method returns "Generic Vehicle", which can be overridden by subclass.

- The get_fee method returns the parking fee, which can be overridden by subclasses Car, Motorcycle, and Truck.

```
class Vehicle:
    def __init__(self, reg_num):
        self.reg_num = reg_num

    def get_vehicle_type(self):
        return "Generic Vehicle"

    def get_fee(self):
        return 0


class Car(Vehicle):
    def get_vehicle_type(self):
        return "Car"

    def get_fee(self):
        return 500
```

Subclass

subclasses of Vehicle.
Each subclass overrides the get_v
ehicle_type method to return the specific vehicle type ("Car", "Motorcycle", "Truck").
Each subclass overrides the get_fee method to return the parking fee specific to that vehicle type.

```
19
20  class Motorcycle(Vehicle):
21      def get_vehicle_type(self):
22          return "Motorcycle"
23
24      def get_fee(self):
25          return 200
26
27
28  class Truck(Vehicle):
29      def get_vehicle_type(self):
30          return "Truck"
31
32      def get_fee(self):
33          return 1500
34
```

ParkingLot class:

It represents a parking lot. It has attributes for capacity, available spaces, occupied spaces, and a dictionary of parking slots.

```
36  class ParkingLot:
37      def __init__(self, capacity):
38          self.capacity = capacity
39          self.available_spaces = capacity
40          self.occupied_spaces = 0
41          self.parking_slots = {}
42
43      def park(self, vehicle):
44          if self.available_spaces > 0:
45              slot = self.find_empty_slot()
46              self.parking_slots[slot] = vehicle
47              self.available_spaces -= 1
48              self.occupied_spaces += 1
49              print(f"Vehicle {vehicle.reg_num} ({vehicle
50              self.charge_fee(vehicle)
51          else:
52              print("Parking lot is full!")
53
54      def find_empty_slot(self):
55          for i in range(1, self.capacity + 1):
56              if i not in self.parking_slots:
57                  return i
```

Method

Park method : The park method parks a vehicle in the parking lot, if there's space available.
The find_empty_slot method finds an empty slot in the parking lot.

Level method : The leave method removes a vehicle from a given slot in the parking lot.

```
def park(self, vehicle):
    if self.available_spaces > 0:
        slot = self.find_empty_slot()
        self.parking_slots[slot] = vehicle
        self.available_spaces -= 1
        self.occupied_spaces += 1
        print(f"Vehicle {vehicle.reg_num} ({vehic
        self.charge_fee(vehicle)
    else:
        print("Parking lot is full!")
```

```
def leave(self, slot):
    if slot in self.parking_slots:
        vehicle = self.parking_slots[slot]
        del self.parking_slots[slot]
        self.available_spaces += 1
        self.occupied_spaces -= 1
        print(f"Vehicle {vehicle.reg_num} left from slot {slot}")
    else:
        print("Slot is already empty!")
```

- display_status method : The display_status method prints the current status of the parking lot, including available and occupied spaces and details of parked vehicles.

- charge_fee method : The charge_fee method calculates and prints the parking fee for a vehicle.

```
def display_status(self):
    print(f"Available Spaces: {self.available_spaces}")
    print(f"Occupied Spaces: {self.occupied_spaces}")
    print("Parking Slots:")
    for slot, vehicle in self.parking_slots.items():
        print(f"Slot {slot}: {vehicle.reg_num} ({vehicle.get_vehicle_t

def charge_fee(self, vehicle):
    fee = vehicle.get_fee()
    print(f"Charged Tk {fee} for parking {vehicle.get_vehicle_type()}.
```

if _name_ == "_main_": block:

In the if _name_ == "_main_":block, a ParkingLot object is created with a capacity of 100.
Instances of Car, Motorcycle, and Truck are created with registration numbers and parked in the parking lot.
The status of the parking lot is displayed after parking each vehicle and after one vehicle leaves.

```python
if __name__ == "__main__":
    parking_lot = ParkingLot(100)

    car1 = Car("Dhaka Metro-1123")
    car2 = Car("Dhaka Metro-1207")
    bike = Motorcycle("Khulna-789")
    truck1 = Truck("Dhaka Metro-7101")
    truck2 =Truck("Khulna Kho-1201")

    parking_lot.park(car1)
    parking_lot.park(car2)
    parking_lot.park(bike)
    parking_lot.display_status()
    parking_lot.leave(2)
    parking_lot.display_status()
    parking_lot.park(truck1)
    parking_lot.park(truck2)
    parking_lot.display_status()
```

Main code

```
class Vehicle:
    def __init__(self, reg_num):
        self.reg_num = reg_num
    def get_vehicle_type(self):
        return "Generic Vehicle"
    def get_fee(self):
        return 0
class Car(Vehicle):
    def get_vehicle_type(self):
        return "Car"
    def get_fee(self):
        return 500
class Motorcycle(Vehicle):
    def get_vehicle_type(self):
        return "Motorcycle"
    def get_fee(self):
        return 200
```

```
class Truck(Vehicle):
    def get_vehicle_type(self):
        return "Truck"
    def get_fee(self):
        return 1500
class ParkingLot:
    def __init__(self, capacity):
        self.capacity = capacity
        self.available_spaces = capacity
        self.occupied_spaces = 0
        self.parking_slots = {}

    def park(self, vehicle):
        if self.available_spaces > 0:
            slot = self.find_empty_slot()
            self.parking_slots[slot] = vehicle
            self.available_spaces -= 1
            self.occupied_spaces += 1
            print(f"Vehicle {vehicle.reg_num}
({vehicle.get_vehicle_type()}) parked at slot
{slot}")
            self.charge_fee(vehicle)
        else:
            print("Parking lot is full!")
```

```
def find_empty_slot(self):
        for i in range(1, self.capacity + 1):
            if i not in self.parking_slots:
                return i
    def leave(self, slot):
        if slot in self.parking_slots:
            vehicle = self.parking_slots[slot]
            del self.parking_slots[slot]
            self.available_spaces += 1
            self.occupied_spaces -= 1
            print(f"Vehicle {vehicle.reg_num} left
from slot {slot}")
        else:
            print("Slot is already empty!")
    def display_status(self):
        print(f"Available Spaces:
{self.available_spaces}")
        print(f"Occupied Spaces:
{self.occupied_spaces}")
        print("Parking Slots:")
        for slot, vehicle in
self.parking_slots.items():
            print(f"Slot {slot}: {vehicle.reg_num}
({vehicle.get_vehicle_type()})")
```

```
def charge_fee(self, vehicle):
        fee = vehicle.get_fee()
        print(f"Charged Tk {fee} for parking
{vehicle.get_vehicle_type() }.")
if __name__ == "__main__":
    parking_lot = ParkingLot(100)  # Capacity of
100
    car1 = Car("Dhaka Metro-1123")
    car2 = Car("Dhaka Metro-1207")
    bike = Motorcycle("Khulna-789")
    truck1 = Truck("Dhaka Metro-7101")
    truck2 =Truck("Khulna Kho-1201")
    parking_lot.park(car1)
    parking_lot.park(car2)
    parking_lot.park(bike)
    parking_lot.display_status()
    parking_lot.leave(2)
    parking_lot.display_status()
    parking_lot.park(truck1)
    parking_lot.park(truck2)
    parking_lot.display_status()
```

Conclusion

Overall, the code provides a flexible and extensible vehicle parking system that can handle different types of vehicles with different parking fees. It demonstrates the use of classes, inheritance, and object-oriented principles in Python.