**SOURCE CODE DOCUMENTATION**

import datetime

class Vehicle:

def \_\_init\_\_(self, vehicle\_number, vehicle\_type, vehicle\_name, owner\_name):

"""

Initialize a Vehicle object with provided details.

Parameters:

vehicle\_number (str): The vehicle number plate.

vehicle\_type (str): The type of the vehicle (Car, Motorcycle, Bicycle, etc.).

vehicle\_name (str): The name of the vehicle.

owner\_name (str): The name of the vehicle owner.

"""

self.vehicle\_number = vehicle\_number

self.vehicle\_type = vehicle\_type

self.vehicle\_name = vehicle\_name

self.owner\_name = owner\_name

self.entry\_time = None

self.exit\_time = None

self.parking\_duration = None

self.parking\_charge = None

self.payment\_status = False

def park(self):

"""

Park the vehicle and record the entry time.

"""

self.entry\_time = datetime.datetime.now()

print(f"Vehicle {self.vehicle\_number} parked at {self.entry\_time}")

def unpark(self):

"""

Unpark the vehicle and record the exit time. Calculate parking charge and update payment status.

"""

self.exit\_time = datetime.datetime.now()

self.parking\_duration = self.exit\_time - self.entry\_time

self.calculate\_parking\_charge()

print(f"Vehicle {self.vehicle\_number} unparked at {self.exit\_time}")

print(f"Parking duration: {self.parking\_duration}")

print(f"Parking charge: {self.parking\_charge} KSH")

self.payment\_status = True

def calculate\_parking\_charge(self):

"""

Calculate the parking charge based on the parking duration and vehicle type.

"""

minutes\_parked = self.parking\_duration.total\_seconds() / 60

if self.vehicle\_type == 'Car':

self.parking\_charge = minutes\_parked \* 10

elif self.vehicle\_ty**pe** == 'Motorcycle':

self.parking\_charge = minutes\_parked \* 5

else:

self.parking\_charge = minutes\_parked \* 3

def display\_details(self):

"""

Display the details of the vehicle including entry time, exit time, parking duration,

parking charge, and payment status.

"""

print(f"Vehicle Number: {self.vehicle\_number}")

print(f"Vehicle Type: {self.vehicle\_type}")

print(f"Vehicle Name: {self.vehicle\_name}")

print(f"Owner Name: {self.owner\_name}")

if self.entry\_time:

print(f"Entry Time: {self.entry\_time}")

if self.exit\_time:

print(f"Exit Time: {self.exit\_time}")

if self.parking\_duration:

print(f"Parking Duration: {self.parking\_duration}")

if self.parking\_charge:

print(f"Parking Charge: {self.parking\_charge} KSH")

print(f"Payment Status: {'Paid' if self.payment\_status else 'Pending'}")

class ParkingLot:

def \_\_init\_\_(self, capacity):

"""

Initialize the ParkingLot object with the given capacity.

Parameters:

capacity (int): The total number of parking spaces in the parking lot.

"""

self.capacity = capacity

self.available\_spaces = capacity

self.occupied\_spaces = 0

self.vehicles = {}

def park\_vehicle(self, vehicle):

"""

Park the vehicle in the parking lot if space is available.

Parameters:

vehicle (Vehicle): The Vehicle object to be parked.

"""

if self.available\_spaces > 0:

self.vehicles[vehicle.vehicle\_number] = vehicle

vehicle.park()

self.available\_spaces -= 1

self.occupied\_spaces += 1

else:

print("Parking lot is full. Cannot park vehicle.")

def unpark\_vehicle(self, vehicle\_number):

"""

Unpark the vehicle from the parking lot.

Parameters:

vehicle\_number (str): The vehicle number plate of the vehicle to be unparked.

"""

if vehicle\_number in self.vehicles:

vehicle = self.vehicles[vehicle\_number]

del self.vehicles[vehicle\_number]

vehicle.unpark()

self.available\_spaces += 1

self.occupied\_spaces -= 1

else:

print("Vehicle not found in the parking lot.")

def search\_vehicle(self, vehicle\_number):

"""

Search for a vehicle in the parking lot and display its details if found.

Parameters:

vehicle\_number (str): The vehicle number plate of the vehicle to search for.

"""

if vehicle\_number in self.vehicles:

vehicle = self.vehicles[vehicle\_number]

print("Vehicle found in the parking lot.")

vehicle.display\_details()

else:

print("Vehicle not found in the parking lot.")

def display\_parked\_vehicles(self):

"""

Display the details of all parked vehicles in the parking lot.

"""

if not self.vehicles:

print("No vehicles parked in the parking lot.")

else:

print("Parked Vehicles:")

for vehicle in self.vehicles.values():

print("---------------------")

vehicle.display\_details()

print("---------------------")

def display\_parking\_lot\_status(self):

"""

Display the current status of the parking lot including total spaces, occupied spaces,

and available spaces.

"""

print("Parking Lot Status:")

print(f"Total Spaces: {self.capacity}")

print(f"Occupied Spaces: {self.occupied\_spaces}")

print(f"Available Spaces: {self.available\_spaces}")

def main():

parking\_lot = ParkingLot(50)

while True:

print("-------------------------------")

print("Vehicle Parking Management System")

print("-------------------------------")

print("1. Park Vehicle")

print("2. Unpark Vehicle")

print("3. Search Vehicle")

print("4. Display Parked Vehicles")

print("5. Display Parking Lot Status")

print("6. Exit")

print("-------------------------------")

choice = input("Enter your choice (1-6): ")

if choice == '1':

vehicle\_number = input("Vehicle Number: ")

vehicle\_type = input("Vehicle Type (Car/Motorcycle/Bicycle): ")

vehicle\_name = input("Vehicle Name: ")

owner\_name = input("Owner Name: ")

vehicle = Vehicle(vehicle\_number, vehicle\_type, vehicle\_name, owner\_name)

parking\_lot.park\_vehicle(vehicle)

elif choice == '2':

vehicle\_number = input("Enter vehicle number: ")

parking\_lot.unpark\_vehicle(vehicle\_number)

elif choice == '3':

vehicle\_number = input("Enter vehicle number: ")

parking\_lot.search\_vehicle(vehicle\_number)

elif choice == '4':

parking\_lot.display\_parked\_vehicles()

elif choice == '5':

parking\_lot.display\_parking\_lot\_status()

elif choice == '6':

break

else:

print("Invalid choice. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**SCREENSHOTS**





