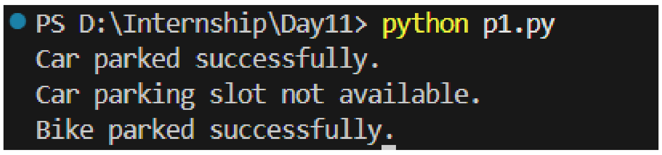
```
from abc import ABC, abstractmethod

class ParkingSpot(ABC):


    @abstractmethod
    def park_vehicle(self):
        pass

class Sensor:
    def __init__(self, availability=True):
        self.__availability = availability

    def is_available(self):
        return self.__availability

    def update_availability(self, status):
        self.__availability = status

class CarParking(ParkingSpot):

    def __init__(self):
        self.sensor = Sensor(True)

    def park_vehicle(self):
        if self.sensor.is_available():
            print("Car parked successfully.")
            self.sensor.update_availability(False)
        else:
            print("Car parking slot not available.")

class BikeParking(ParkingSpot):
    def __init__(self):
        self.sensor = Sensor(True)
    def park_vehicle(self):
        if self.sensor.is_available():
            print("Bike parked successfully.")
            self.sensor.update_availability(False)
        else:
            print("Bike parking slot not available.")

car_spot = CarParking()
bike_spot = BikeParking()

car_spot.park_vehicle()
car_spot.park_vehicle()

bike_spot.park_vehicle()
```



```
PS D:\Internship\Day11> python p1.py
Car parked successfully.
Car parking slot not available.
Bike parked successfully.
```