```python
#Hlanhla Hlungwane
#30 April 2025
#Define a class for an Online Car Rental Platform
#Object Oriented Programming on Python
#Perfomed on Jupyter Notebook

import datetime  # Importing the built-in datetime module

class CarRental:
 """
 A class to manage car rental operations.
 """

 def __init__(self, inventory):
     """
     Constructor to initialize the car rental system.
     :inventory: Number of cars available for rent.
     """
     self.inventory = inventory  # Total number of cars available
     self.rental_time = None     # Stores the time when cars are rented
     self.rental_mode = None     # Stores the rental mode (hourly, daily, weekly)
     self.rented_cars = 0        # Stores the number of cars rented

 def display_available_cars(self):
     """
     Displays the number of available cars in the inventory.
     """
     print(f"Available cars for rent: {self.inventory}")

 def rent_car_hourly(self, num_of_cars):
     """
     Rents cars on an hourly basis.
     :num_of_cars: Number of cars requested for rent.
     """
     if num_of_cars <= 0:
         print("Number of cars must be more than zero.")
         return None
     elif num_of_cars > self.inventory:
         print("Not enough cars available for hourly rental.")
         return None
     else:
         self.rental_time = datetime.datetime.now()  # Store the current time
         self.rental_mode = "hourly"                 # Set rental mode
         self.rented_cars = num_of_cars              # Store the number of rented cars
         self.inventory -= num_of_cars               # Update inventory
         print(f"{num_of_cars} car(s) rented on an hourly basis at {self.rental_time}.")
         return self.rental_time

 def rent_car_daily(self, num_of_cars):
     """
     Rents cars on a daily basis.
     :num_of_cars: Number of cars requested for rent.
     """
     if num_of_cars <= 0:
         print("Number of cars must be greater than zero.")
         return None
     elif num_of_cars > self.inventory:
         print("Not enough cars available for daily rental.")
         return None
     else:
         self.rental_time = datetime.datetime.now()  # Store the current time
         self.rental_mode = "daily"                  # Set rental mode
         self.rented_cars = num_of_cars              # Store the number of rented cars
         self.inventory -= num_of_cars               # Update inventory
         print(f"{num_of_cars} car(s) rented on a daily basis at {self.rental_time}.")
         return self.rental_time
```

```python
def rent_car_weekly(self, num_of_cars):
    """
    Rents cars on a weekly basis.
    :num_of_cars: Number of cars requested for rent.
    """
    if num_of_cars <= 0:
        print("Number of cars must be greater than zero.")
        return None
    elif num_of_cars > self.inventory:
        print("Not enough cars available for weekly rental.")
        return None
    else:
        self.rental_time = datetime.datetime.now()  # Store the current time
        self.rental_mode = "weekly"                 # Set rental mode
        self.rented_cars = num_of_cars              # Store the number of rented cars
        self.inventory -= num_of_cars               # Update inventory
        print(f"SUCCESS!!! {num_of_cars} car(s) rented on a weekly basis at {self.rental_time}.")
        return self.rental_time

def return_car(self):
    """
    Returns the rented cars, calculates the rental period, and generates the final bill.
    """
    if self.rental_time is None:
        print("No cars have been rented.")
        return None

    # Calculate the rental period
    return_time = datetime.datetime.now()
    rental_period = return_time - self.rental_time

    # Determine cost based on rental mode
    if self.rental_mode == "hourly":
        bill = rental_period.seconds / 3600 * 250 * self.rented_cars  # R250 per hour per car
    elif self.rental_mode == "daily":
        bill = rental_period.days * 1200 * self.rented_cars  # R1200 per day per car
    elif self.rental_mode == "weekly":
        bill = (rental_period.days / 7) * 4500 * self.rented_cars  # R4500 per week per car
    else:
        print("Invalid Selection")
        return None

    # Update inventory
    self.inventory += self.rented_cars
    self.rental_time = None
    self.rental_mode = None
    self.rented_cars = 0

    print(f"Cars returned successfully. Total bill: R{bill:.2f}")
    return bill

class Customer:
 """
 A class to represent customers interacting with the car rental system.
 """

 def __init__(self):
     """
     Constructor to initialize customer details.
     """
     self.rented_cars = 0  # Tracks the number of cars rented by the customer

 def request_car(self):
     """
     Allows the customer to request cars for rent.
     :return: Number of cars requested.
     """
```

```
    try:
        num_of_cars = int(input("How many cars do you want to rent? "))
        if num_of_cars <= 0:
            print("Number of cars must be more than zero.")
            return None
        else:
            self.rented_cars = num_of_cars
            return num_of_cars
    except ValueError:
        print("Invalid input. Please enter a positive number")
        return None

def return_car(self):
    """
    Allows the customer to return rented cars.
    :return: Confirmation that cars are being returned.
    """
    if self.rented_cars == 0:
        print("You haven't rented any car")
        return None
    else:
        print(f"You have returned {self.rented_cars} car(s).")
        return self.rented_cars
```