from datetime import datetime, timedelta

class CarRental:

    def \_\_init\_\_(self, stock=0):

        """Constructor to initialize the car rental stock."""

        self.stock = stock

    def display\_stock(self):

        """Displays the cars available for rent."""

        print(f"We have currently {self.stock} cars available to rent.")

        return self.stock

    def rent\_car\_hourly(self, num\_of\_cars):

        """Rents cars on an hourly basis."""

        if num\_of\_cars <= 0:

            print("Number of cars should be positive.")

            return None

        elif num\_of\_cars > self.stock:

            print(f"Sorry, we have currently {self.stock} cars available to rent.")

            return None

        else:

            now = datetime.now()

            print(f"You have rented {num\_of\_cars} car(s) on hourly basis today at {now.hour} hours.")

            self.stock -= num\_of\_cars

            return now

    def rent\_car\_daily(self, num\_of\_cars):

        """Rents cars on a daily basis."""

        if num\_of\_cars <= 0:

            print("Number of cars should be positive.")

            return None

        elif num\_of\_cars > self.stock:

            print(f"Sorry, we have currently {self.stock} cars available to rent.")

            return None

        else:

            now = datetime.now()

            print(f"You have rented {num\_of\_cars} car(s) on daily basis today at {now.hour} hours.")

            self.stock -= num\_of\_cars

            return now

    def rent\_car\_weekly(self, num\_of\_cars):

        """Rents cars on a weekly basis."""

        if num\_of\_cars <= 0:

            print("Number of cars should be positive.")

            return None

        elif num\_of\_cars > self.stock:

            print(f"Sorry, we have currently {self.stock} cars available to rent.")

            return None

        else:

            now = datetime.now()

            print(f"You have rented {num\_of\_cars} car(s) on weekly basis today at {now.hour} hours.")

            self.stock -= num\_of\_cars

            return now

    def return\_car(self, request):

        """Accepts a rented car and generates a bill."""

        rental\_time, rental\_basis, num\_of\_cars = request

        bill = 0

        if rental\_time and rental\_basis and num\_of\_cars:

            self.stock += num\_of\_cars

            now = datetime.now()

            rental\_period = now - rental\_time

            if rental\_basis == 'hourly':

                bill = round(rental\_period.seconds / 3600) \* 5 \* num\_of\_cars

            elif rental\_basis == 'daily':

                bill = round(rental\_period.days) \* 20 \* num\_of\_cars

            elif rental\_basis == 'weekly':

                bill = round(rental\_period.days / 7) \* 60 \* num\_of\_cars

            if (3 <= num\_of\_cars <= 5):

                print("You are eligible for a 30% discount.")

                bill = bill \* 0.7

            print(f"Thank you for returning your car(s). Your bill is ${bill}.")

            return bill

        else:

            print("Are you sure you rented a car with us?")

            return None

class Customer:

    def \_\_init\_\_(self):

        """Constructor to initialize the customer details."""

        self.cars = 0

        self.rental\_basis = None

        self.rental\_time = None

    def request\_car(self):

        """Takes a request from the customer for the number of cars."""

        cars = input("How many cars would you like to rent?")

        try:

            cars = int(cars)

        except ValueError:

            print("Number of cars should be a positive integer.")

            return -1

        if cars < 1:

            print("Invalid input. Number of cars should be greater than zero.")

            return -1

        else:

            self.cars = cars

            return self.cars

    def return\_car(self):

        """Allows the customer to return cars."""

        if self.rental\_basis and self.rental\_time and self.cars:

            return self.rental\_time, self.rental\_basis, self.cars

        else:

            return 0, 0, 0