0nline car rental project

1.Create a module (.py file) for car rental and import the built-in module DateTime to handle the rental time and bill.

In [6]:
```
import datetime

x = datetime.datetime.now()

print(x)

```

```
2024-02-27 22:24:59.051333
```

In [ ]:

```
 ##  2.Create a class for renting the cars and define a constructor in it.
import datetime
import math
from abc import ABCMeta, abstractmethod


class Constants:
    """This class contains all the constants used in this application."""

    HOURLY_RENTAL_MODE = "H"
    DAILY_RENTAL_MODE = "D"
    WEEKLY_RENTAL_MODE = "W"
    DEFAULT_RENTAL_MODE = None
    CAR_CONDITION_OKAY = "okay"
    CAR_CONDITION_NOT_OKAY = "not okay"
    DEFAULT_MAKE_MODEL = "Hyundai i10"
    TOTAL_NUMBER_OF_CARS_AVAILABLE = 100
    HOURLY_RENT_COST = 100
    DAILY_RENT_COST = 2000
    WEEKLY_RENT_COST = 12000
    DATE_FORMAT = '%Y-%m-%d %H:%M:%S'

class Car:
    """This class acts as the base or parent class for rental cars and pro

    def __init__(self):
        self.make_model = Constants.DEFAULT_MAKE_MODEL

    def set_make_model(self, other_make_model):
        self.make_model = other_make_model

class RentalCar(Car):
    """The class provides the details of availability of the car, conditio

    def __init__(self):
        super().__init__()
        self.rental_car_id = 0
        self.car_condition = Constants.CAR_CONDITION_OKAY
        self.rental_mode = Constants.DEFAULT_RENTAL_MODE
        self.customer_rented_to = None
        self.available_now = True
        self.rented_time_stamp = None

class CarCompany(metaclass = ABCMeta):
    """This class is created to represent a business entity. This business

    __number_of_cars_available = Constants.TOTAL_NUMBER_OF_CARS_AVAILABLE
    _cars_available = []

    def __init__(self):
        for i in range(1, self.__number_of_cars_available + 1):
            rental_car = RentalCar()
            rental_car.rental_car_id = i
            self._cars_available.append(rental_car)
    @abstractmethod
    def request_cars(self, rental_mode, the_number_of_cars_rented, custome
        pass

    @abstractmethod
    def return_cars(self, rental_mode, the_number_of_cars_rented, customer
        pass




```

This class facilitates the ease of doing business (in online medium) and has the following:

1. Defines a method for displaying the available cars.

2. Define methods for renting cars on an hourly, daily and weekly basis, respectively.

3. Defines a method that validates the number of requested cars is positive and lesser than the total available cars.

4. Defines a method that stores the time of renting a car in a variable, which can later be used in the bill while returning the car.

5. Defines a method to return the cars using rental time, rental mode (hourly, daily, or weekly), and the number of cars rented.

6. Inside the return method; update the inventory stock, calculate the rental period, and generate the final bill.

In [15]:

```
    def __init__(self):
        super().__init__()

    def fetch_number_of_available_cars(self):
        number_of_available_cars = 0
        for rental_car in self._cars_available:
            if rental_car.available_now:
                number_of_available_cars += 1
        return number_of_available_cars

```

In [18]:
```
    def validate_rental_mode_for_renting_a_car(self, rental_mode):
            if rental_mode == Constants.HOURLY_RENTAL_MODE:
                return True
            elif rental_mode == Constants.DAILY_RENTAL_MODE:
                return True
            elif rental_mode == Constants.WEEKLY_RENTAL_MODE:
                return True
            else:
                print("You have entered an invalid rental mode.")
                return False

```

In [19]:
```
    def rent_cars_based_on_rental_mode(self, number_of_cars_requested, ren
        self.request_cars(rental_mode, number_of_cars_requested, customer_
        if rental_mode == Constants.HOURLY_RENTAL_MODE:
            print("Your request is fulfilled. You will have to pay Rs. {0}
                Constants.HOURLY_RENT_COST))
        elif rental_mode == Constants.DAILY_RENTAL_MODE:
            print("Your request is fulfilled. You will have to pay Rs. ",
                  " per day per car.\n\n")
        elif rental_mode == Constants.WEEKLY_RENTAL_MODE:
            print("Your request is fulfilled. You will have to pay Rs. " +
        else:
            print("You have entered an invalid rental mode.\n\n")

```

In [20]:
```
    def validate_number_of_requested_cars(self, number_of_cars_requested):
        if number_of_cars_requested <= 0:
            print("You have requested an invalid number of cars. Please en
            return False
        else:
            number_of_cars_available_for_renting = self.fetch_number_of_av
            if number_of_cars_requested > number_of_cars_available_for_ren
                print("You have requested an invalid number of cars. Pleas
                return False
        return True

```

In [21]:
```
    def request_cars(self, rental_mode, the_number_of_cars_rented, custome
        time_stamp_now = datetime.datetime.now()
        total_cars_rented_to_customer = 0
        for rental_car in self._cars_available:
            if rental_car.available_now:
                rental_car.available_now = False
                rental_car.rental_mode = rental_mode
                rental_car.customer_rented_to = customer_name
                self.store_timestamp_for_renting_the_car(rental_car, time_
                total_cars_rented_to_customer += 1
                if total_cars_rented_to_customer == the_number_of_cars_ren
                    break


```

In [ ]:
```
def store_timestamp_for_renting_the_car(self, rental_car: RentalCar, time_
        rental_car.rented_time_stamp = time_stamp_now

    def fetch_records_of_existing_customer(self, customer_name):
        records_of_customer = []
        for rental_car in self._cars_available:
            if rental_car.customer_rented_to == customer_name:
                records_of_customer.append(rental_car)
        return records_of_customer

```

In [22]:
```
    def return_cars(self, rental_mode, the_number_of_cars_rented, customer_
        total_cars_returned_by_customer = 0
        time_stamp_at_the_time_of_renting = None
        for rental_car in self._cars_available:
            if rental_car.available_now is False and rental_car.customer_r
                time_stamp_at_the_time_of_renting = rental_car.rented_time_
                rental_car.available_now = True
                rental_car.rental_mode = Constants.DEFAULT_RENTAL_MODE
                rental_car.customer_rented_to = None
                total_cars_returned_by_customer += 1
                if total_cars_returned_by_customer == the_number_of_cars_r
                    break
        return time_stamp_at_the_time_of_renting

```

In [ ]:
```
    def return_the_rented_cars_and_generate_invoice(self, rental_time, ren
        time_stamp_at_the_time_of_renting = self.return_cars(rental_mode, 
        rental_period = self.calculate_rental_period(rental_time, time_sta
        total_bill_amount = self.calculate_total_bill_amount(rental_mode, 
        self.display_bill(customer_name, the_number_of_cars_rented, time_s

```

In [ ]:
```
    def calculate_rental_period(self, rental_time, time_stamp_at_the_time_
        total_rental_period = 0
        difference_between_two_dates = rental_time - time_stamp_at_the_tim
        total_duration_seconds = difference_between_two_dates.seconds
        if rental_mode == Constants.HOURLY_RENTAL_MODE:
            total_days = difference_between_two_dates.days
            total_duration_seconds += total_days * 24 * 60 * 60
            total_rental_period = math.ceil(total_duration_seconds / 3600.
        elif rental_mode == Constants.DAILY_RENTAL_MODE:
            total_rental_period += difference_between_two_dates.days
            total_rental_period += math.ceil(total_duration_seconds / 8640
        elif rental_mode == Constants.WEEKLY_RENTAL_MODE:
            total_days = difference_between_two_dates.days
            total_duration_seconds += total_days * (24 * 60 * 60)
            total_rental_period += math.ceil(total_duration_seconds / (7 *
        else:
            print("You have entered an invalid rental mode.")
        return int(total_rental_period)


```

In [ ]:
```
def calculate_total_bill_amount(self, rental_mode, rental_period, number_o
        total_bill_amount = 0.0
        if rental_mode == Constants.HOURLY_RENTAL_MODE:
            total_bill_amount = Constants.HOURLY_RENT_COST * rental_period
        elif rental_mode == Constants.DAILY_RENTAL_MODE:
            total_bill_amount = Constants.DAILY_RENT_COST * rental_period
        elif rental_mode == Constants.WEEKLY_RENTAL_MODE:
            total_bill_amount = Constants.WEEKLY_RENT_COST * rental_period
        else:
            print("You have entered an invalid rental mode.")
        return float(total_bill_amount)


```

In [ ]:
```
def display_bill(self, customer_name, the_number_of_cars_rented, time_stam
        print("-----------------------------------------------------------
        print("INVOICE\n")
        print("Car Company")
        print("An online car renting service provider.\n\n")
        print("Customer Name : " + customer_name)
        print("Date: " + str.split(str(datetime.datetime.now()), ".")[0])
        print("\nNumber of cars rented : " + str(the_number_of_cars_rented
        print("Rented on (date and time) : " + str.split(str(time_stamp_at
        print("Returned on (date and time) : " + str(rental_time))
        if rental_mode == "H":
            print("Rental mode : Hourly")
            print("Charges per hour : Rs. "+str(Constants.HOURLY_RENT_COST
        elif rental_mode == "D":
            print("Rental mode : Daily")
            print("Charges per day : Rs. "+str(Constants.DAILY_RENT_COST))
        elif rental_mode == "W":
            print("Rental mode : Weekly")
            print("Charges per week : Rs. "+str(Constants.WEEKLY_RENT_COST
        print("Rental period : " + str(rental_period))
        print("\n\nTotal bill amount : Rs. " + str('%.2f' % total_bill_amo
        print("-----------------------------------------------------------

    def display_welcome_screen(self):
        print("""Welcome to Car Company. We are an online car renting serv
        print("Total cars available now : " + str(self.fetch_number_of_ava
        customer_name = input("\nPlease enter your name to continue : ")
        while not self.validate_customer_name(customer_name):
            customer_name = str.rstrip(str.lstrip(input("\nPlease enter yo
        return customer_name

    def validate_customer_name(self, customer_name):
        if customer_name is None:
            return False
        if len(customer_name) == 0:
            return False
        else:
            return True

    def greet_customer_and_ask_for_input(self, customer_name):
        print("Hello ", customer_name)
        print("Total cars available now : " + str(self.fetch_number_of_ava
        print("What would you like to do \n\t1. Book cars\n\t2. Return car
        user_input = int(input("Please type 1 or 2 or 3 to proceed "))
        if user_input == 1 or user_input == 2:
            return user_input
        else:
            print("You have entered an invalid input. You are redirected t

    def book_cars(self, number_of_cars_requested, customer_name):
        if self.validate_number_of_requested_cars(number_of_cars_requested
            print("On which basis would your like to pay for cars? Please 
            print("Please type H for hourly. You will have to pay Rs. ", s
            print("Please type D for daily. You will have to pay Rs. ", st
            print("Please type W for weekly. You will have to pay Rs. ", s
            rental_mode = str.upper(input("Please provide rental mode: "))
            if self.validate_rental_mode_for_renting_a_car(rental_mode):
                self.rent_cars_based_on_rental_mode(number_of_cars_request
            else:
                print("You have entered an invalid rental mode.")
                rental_mode = str.upper(input("Please provide rental mode:
                if self.validate_rental_mode_for_renting_a_car(rental_mode
                    self.rent_cars_based_on_rental_mode(number_of_cars_req
                                                       custome
        else:
            print("You have requested an invalid number of cars.")

    def return_cars_rented_earlier(self, records_of_customer, customer_nam
        print("Your details are: ")
        hourly_basis_cars_rented = self.fetch_number_of_cars_rented_on_hou
        daily_basis_cars_rented = self.fetch_number_of_cars_rented_on_dail
        weekly_basis_cars_rented = self.fetch_number_of_cars_rented_on_wee
        print("Press 1 to return cars rented on hourly basis. You have ren
        print("Press 2 to return cars rented on daily basis. You have rent
        print("Press 3 to return cars rented on weekly basis. You have ren
        users_choice = int(input("Enter your choice: "))
        if users_choice == 1:
            number_of_cars_returned = int(
                input("How many cars do you want to return now? (only no.
            if 0 < number_of_cars_returned <= hourly_basis_cars_rented:
                rental_time = self.get_rental_time_from_user()
                self.return_the_rented_cars_and_generate_invoice(rental_ti
            else:
                print("Invalid input provided.")
        elif users_choice == 2:
            number_of_cars_returned = int(
                input("How many cars do you want to return now? (only no.
            if 0 < number_of_cars_returned <= daily_basis_cars_rented:
                rental_time = self.get_rental_time_from_user()
                self.return_the_rented_cars_and_generate_invoice(rental_ti



            else:
                print("Invalid input provided.")
        elif users_choice == 3:
            number_of_cars_returned = int(
                input("How many cars do you want to return now? (only no.
            if 0 < number_of_cars_returned <= weekly_basis_cars_rented:
                rental_time = self.get_rental_time_from_user()
                self.return_the_rented_cars_and_generate_invoice(rental_ti



            else:
                print("Invalid input provided.")
        else:
            print("You have entered an invalid input.")

    def fetch_number_of_cars_rented_on_hourly_basis(self, records_of_custo
        number_of_cars_rented = 0
        for rental_car in records_of_customer:
            if rental_car.rental_mode == Constants.HOURLY_RENTAL_MODE:
                number_of_cars_rented += 1
        return number_of_cars_rented

    def fetch_number_of_cars_rented_on_daily_basis(self, records_of_custom
        number_of_cars_rented = 0
        for rental_car in records_of_customer:
            if rental_car.rental_mode == Constants.DAILY_RENTAL_MODE:
                number_of_cars_rented += 1
        return number_of_cars_rented

    def fetch_number_of_cars_rented_on_weekly_basis(self, records_of_custo
        number_of_cars_rented = 0
        for rental_car in records_of_customer:
            if rental_car.rental_mode == Constants.WEEKLY_RENTAL_MODE:
                number_of_cars_rented += 1
        return number_of_cars_rented

    def get_rental_time_from_user(self):
        print(
            "Enter the date and time (in YYYY-MM-DD HH:MM:SS format) when
        print("For example: " + str.split(str(datetime.datetime.now()), ".
        rental_time_str = str(input("Enter date and time : "))
        rental_time = datetime.datetime.strptime(rental_time_str, Constant
        return rental_time


```

class Customer:

"""

This class facilitates the customers and provides them the following options:

1. Defines a methods for requesting the cars.

2. Defines a method for returning the cars. ¶

"""

In [ ]:

```
    def __init__(self):
        self.customer_name = ""

    def request_cars(self):
        while True:
            try:
                return int(input("How many cars do you want to book now? (
            except ValueError:
                print("you have entered an invalid input. Please provide yo

    def return_cars(self, number_of_cars_rented):
        if number_of_cars_rented == 0:
            print("Hey ", self.customer_name, "!, According to our records
        else:
            print("According to our records, you have already rented car fr


if __name__ == '__main__':
    online_car_rental_platform = CarCompanyOnlinePlatform()
    while True:
        customer_name = online_car_rental_platform.display_welcome_screen(
        visiting_customer = Customer()
        visiting_customer.customer_name = customer_name
        user_input = online_car_rental_platform.greet_customer_and_ask_for_
        if user_input == 1:
            number_of_cars_required = visiting_customer.request_cars()
            online_car_rental_platform.book_cars(number_of_cars_required, 
        elif user_input == 2:
            records_of_customer = online_car_rental_platform.fetch_records_
            visiting_customer.return_cars(len(records_of_customer))
            if len(records_of_customer) != 0:
                online_car_rental_platform.return_cars_rented_earlier(recor


```

In [ ]:

```

```