Car Rental System

Project D3

—

Mohy Elden Mohammed
ID: 1588
Section (19)

Introduction

Overview

The car rental system is a program that allows users to manage and perform various operations related to car rental. It provides functionality to add, edit, and delete cars, add and edit customers, rent and return cars, view available cars, view rental history, generate reports, and process payments. The system uses the CarRentalSystem class to handle these operations.

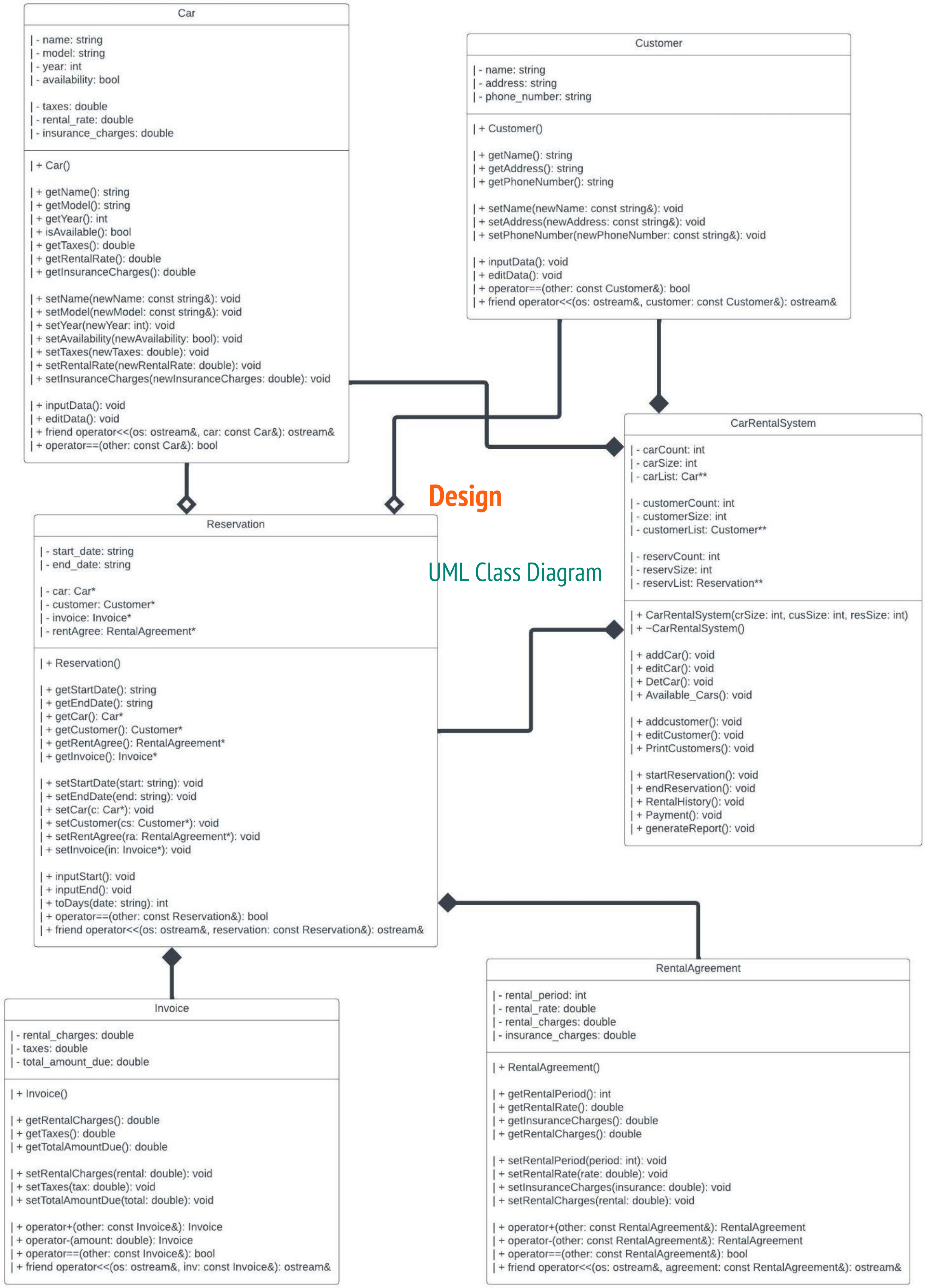
Specifications

The CarRentalSystem class is instantiated with three parameters: the maximum number of cars, the maximum number of customers, and the maximum number of reservations. The system displays a menu of options for users to choose from. The available options include:

1. Add a Car: Allows users to add a new car to the system.
2. Edit a Car: Enables users to edit the details of an existing car.
3. Delete a Car: Marks a car as not available for reservation (logical deletion).
4. Add a Customer: Allows users to add a new customer to the system.
5. Edit a Customer: Enables users to edit the details of an existing customer.
6. View All Customers: Displays a list of all customers in the system.
7. Rent a Car: Initiates the process of renting a car by reserving it for a customer.
8. Return a Car: Marks a rented car as returned, ending the reservation.
9. View Available Cars: Displays a list of cars that are available for rent.
10. View Rental History: Shows the rental history, including past reservations and returns for customers or cars.
11. Generate Reports: Generates reports cars , customers and reservations.
12. Payment of Invoice: Processes the payment for a rental invoice.

Design

UML Class Diagram



```
Car
| - name: string
| - model: string
| - year: int
| - availability: bool

| - taxes: double
| - rental_rate: double
| - insurance_charges: double

| + Car()

| + getName(): string
| + getModel(): string
| + getYear(): int
| + isAvailable(): bool
| + getTaxes(): double
| + getRentalRate(): double
| + getInsuranceCharges(): double

| + setName(newName: const string&): void
| + setModel(newModel: const string&): void
| + setYear(newYear: int): void
| + setAvailability(newAvailability: bool): void
| + setTaxes(newTaxes: double): void
| + setRentalRate(newRentalRate: double): void
| + setInsuranceCharges(newInsuranceCharges: double): void

| + inputData(): void
| + editData(): void
| + friend operator<<(os: ostream&, car: const Car&): ostream&
| + operator==(other: const Car&): bool
```

```
Customer
| - name: string
| - address: string
| - phone_number: string

| + Customer()

| + getName(): string
| + getAddress(): string
| + getPhoneNumber(): string

| + setName(newName: const string&): void
| + setAddress(newAddress: const string&): void
| + setPhoneNumber(newPhoneNumber: const string&): void

| + inputData(): void
| + editData(): void
| + operator==(other: const Customer&): bool
| + friend operator<<(os: ostream&, customer: const Customer&): ostream&
```

```
CarRentalSystem
| - carCount: int
| - carSize: int
| - carList: Car**

| - customerCount: int
| - customerSize: int
| - customerList: Customer**

| - reservCount: int
| - reservSize: int
| - reservList: Reservation**

| + CarRentalSystem(crSize: int, cusSize: int, resSize: int)
| + ~CarRentalSystem()

| + addCar(): void
| + editCar(): void
| + DetCar(): void
| + Available_Cars(): void

| + addcustomer(): void
| + editCustomer(): void
| + PrintCustomers(): void

| + startReservation(): void
| + endReservation(): void
| + RentalHistory(): void
| + Payment(): void
| + generateReport(): void
```

```
Reservation
| - start_date: string
| - end_date: string

| - car: Car*
| - customer: Customer*
| - invoice: Invoice*
| - rentAgree: RentalAgreement*

| + Reservation()

| + getStartDate(): string
| + getEndDate(): string
| + getCar(): Car*
| + getCustomer(): Customer*
| + getRentAgree(): RentalAgreement*
| + getInvoice(): Invoice*

| + setStartDate(start: string): void
| + setEndDate(end: string): void
| + setCar(c: Car*): void
| + setCustomer(cs: Customer*): void
| + setRentAgree(ra: RentalAgreement*): void
| + setInvoice(in: Invoice*): void

| + inputStart(): void
| + inputEnd(): void
| + toDays(date: string): int
| + operator==(other: const Reservation&): bool
| + friend operator<<(os: ostream&, reservation: const Reservation&): ostream&
```

```
Invoice
| - rental_charges: double
| - taxes: double
| - total_amount_due: double

| + Invoice()

| + getRentalCharges(): double
| + getTaxes(): double
| + getTotalAmountDue(): double

| + setRentalCharges(rental: double): void
| + setTaxes(tax: double): void
| + setTotalAmountDue(total: double): void

| + operator+(other: const Invoice&): Invoice
| + operator-(amount: double): Invoice
| + operator==(other: const Invoice&): bool
| + friend operator<<(os: ostream&, inv: const Invoice&): ostream&
```

```
RentalAgreement
| - rental_period: int
| - rental_rate: double
| - rental_charges: double
| - insurance_charges: double

| + RentalAgreement()

| + getRentalPeriod(): int
| + getRentalRate(): double
| + getInsuranceCharges(): double
| + getRentalCharges(): double

| + setRentalPeriod(period: int): void
| + setRentalRate(rate: double): void
| + setInsuranceCharges(insurance: double): void
| + setRentalCharges(rental: double): void

| + operator+(other: const RentalAgreement&): RentalAgreement
| + operator-(other: const RentalAgreement&): RentalAgreement
| + operator==(other: const RentalAgreement&): bool
| + friend operator<<(os: ostream&, agreement: const RentalAgreement&): ostream&
```

Underlying Data Structure

The Car Rental system class uses an array to store Cars, Customers, and Reservations. There is no option to delete an object except in one scenario when making a car not available for reservation (logical deletion), and it restricts the user from adding more clients than the size of the array.

to solve these problems, a better solution for inserting would be to use a linked list. Although accessing elements in a linked list takes O(n) time, a hash table provides efficient access time takes O(1) time, making it a better option.

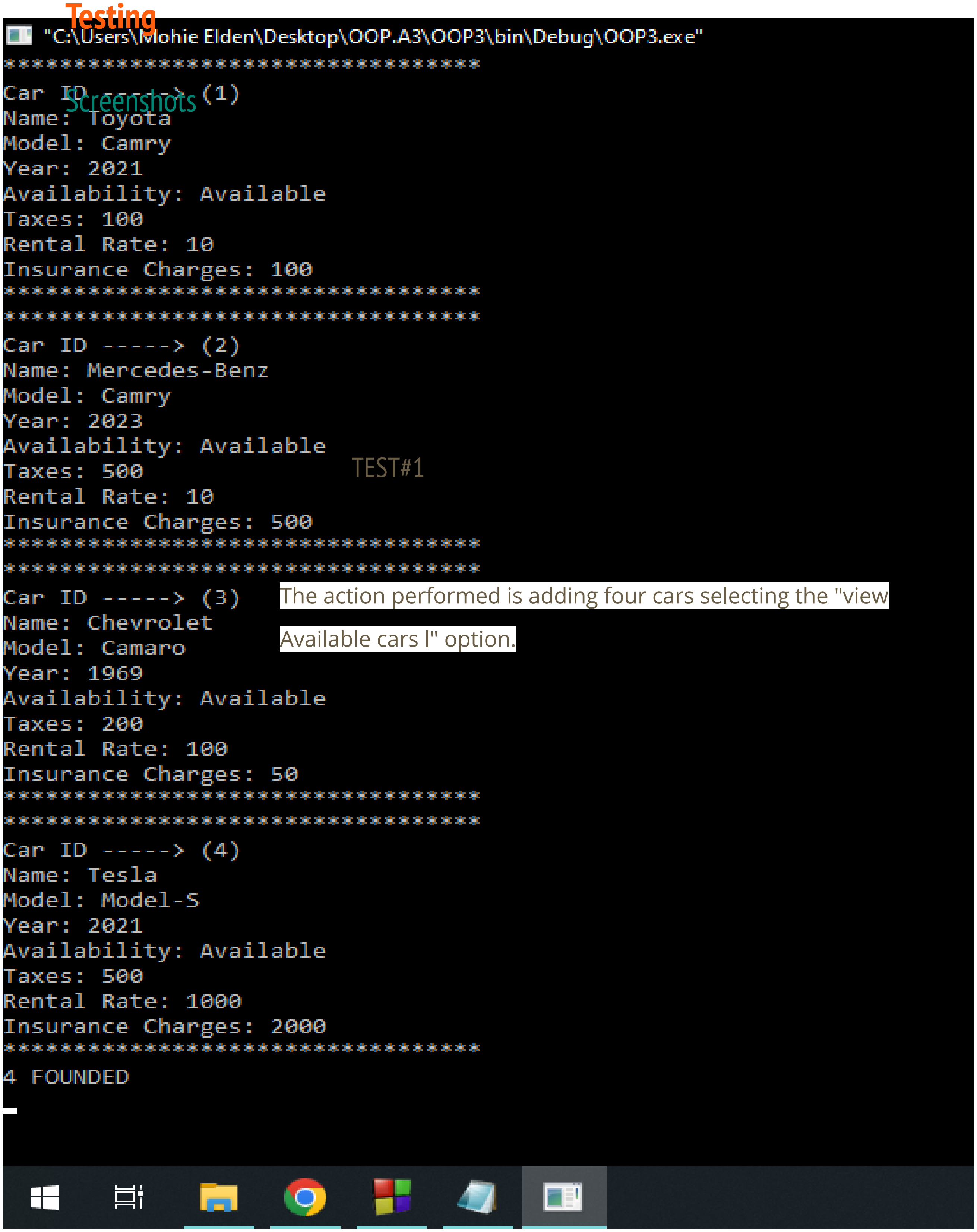
Describe Functionality

1. CarRentalSystem(int crSize, int cusSize, int resSize): This is the constructor of the CarRentalSystem class. It initializes the member variables carCount, carSize, carList, customerCount, customerSize, customerList, reservCount, reservSize, and reservList based on the provided sizes.
2. ~CarRentalSystem(): This is the destructor of the CarRentalSystem class. It deallocates the memory allocated for carList, customerList, and reservList arrays using the delete operator.
3. void addCar(): This function allows adding a new car to the system. It checks if there is enough space in the carList array, creates a new Car object, takes input for the car's data, and adds it to the carList.
4. void editCar(): This function allows editing the details of a specific car in the system. It prompts the user to enter the car ID, retrieves the corresponding car object from the carList, and calls the editData() function on that car object to modify its data.
5. void DetCar(): This function marks a specific car as unavailable. It prompts the user to enter the car ID, retrieves the corresponding car object from the carList, and sets its availability status to false.
6. void Available_Cars(): This function displays the list of available cars in the system. It iterates through the carList, checks the availability status of each car, and prints the details of the available cars.
7. void addcustomer(): This function allows adding a new customer to the system. It checks if there is enough space in the customerList array, creates a new Customer object, takes input for the customer's data, and adds it to the customerList.
8. void editCustomer(): This function allows editing the details of a specific customer in the system. It prompts the user to enter the customer ID, retrieves the corresponding customer object from the customerList, and calls the editData() function on that customer object to modify its data.
9. void PrintCustomers(): This function displays the list of all customers in the system. It iterates through the customerList and prints the details of each customer.

10. void startReservation(): This function starts a new reservation. It prompts the user to enter the car ID, customer ID, start date, and end date. It checks the availability of the chosen car, calculates the rental charges, insurance charges, and total amount due, and creates instances of Reservation, RentalAgreement, and Invoice classes. It sets the corresponding attributes for the reservation and adds it to the reservList.
11. void endReservation(): This function ends an existing reservation. It prompts the user to enter the reservation ID, input the end date, and calculates the actual rental period. It checks if the actual period exceeds the reserved period and applies additional charges if necessary. It updates the rental charges and total amount due in the corresponding Invoice and RentalAgreement objects. It also sets the availability of the car associated with the reservation to true.
12. void RentalHistory(): This function allows generating a rental history report. It prompts the user to choose between generating the report for a specific car or customer. Based on the choice, it prompts for the car ID or customer ID and displays the reservation details associated with
13. void generateReport() function provides a comprehensive overview of the cars, customers, and reservations in the car rental system. It ensures that the report is properly formatted and includes appropriate messages when there is no data available for a specific category.
14. void Payment() function allows the user to make a payment for a reservation by entering the reservation ID and the amount they wish to pay. It handles scenarios where the invoice has already been paid, calculates the remaining amount or overpayment, and updates the invoice accordingly.

Testing

Screenshots

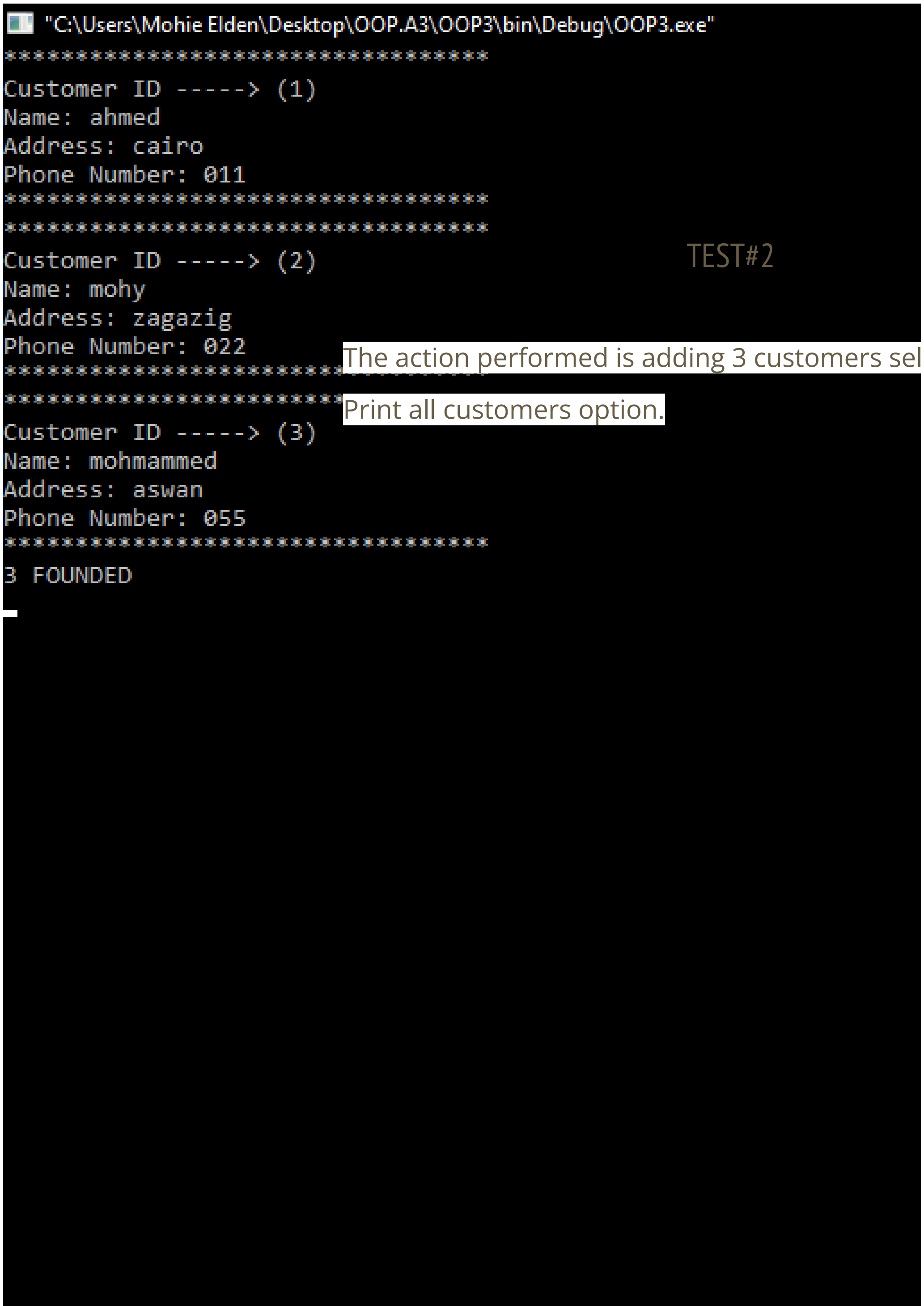


"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
**********************************
Car ID -----> (1)
Name: Toyota
Model: Camry
Year: 2021
Availability: Available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100
**********************************
**********************************
Car ID -----> (2)
Name: Mercedes-Benz
Model: Camry
Year: 2023
Availability: Available
Taxes: 500
Rental Rate: 10
Insurance Charges: 500
**********************************
**********************************
Car ID -----> (3)
Name: Chevrolet
Model: Camaro
Year: 1969
Availability: Available
Taxes: 200
Rental Rate: 100
Insurance Charges: 50
**********************************
**********************************
Car ID -----> (4)
Name: Tesla
Model: Model-S
Year: 2021
Availability: Available
Taxes: 500
Rental Rate: 1000
Insurance Charges: 2000
**********************************
4 FOUNDED
```

TEST#1

The action performed is adding four cars selecting the "view Available cars l" option.

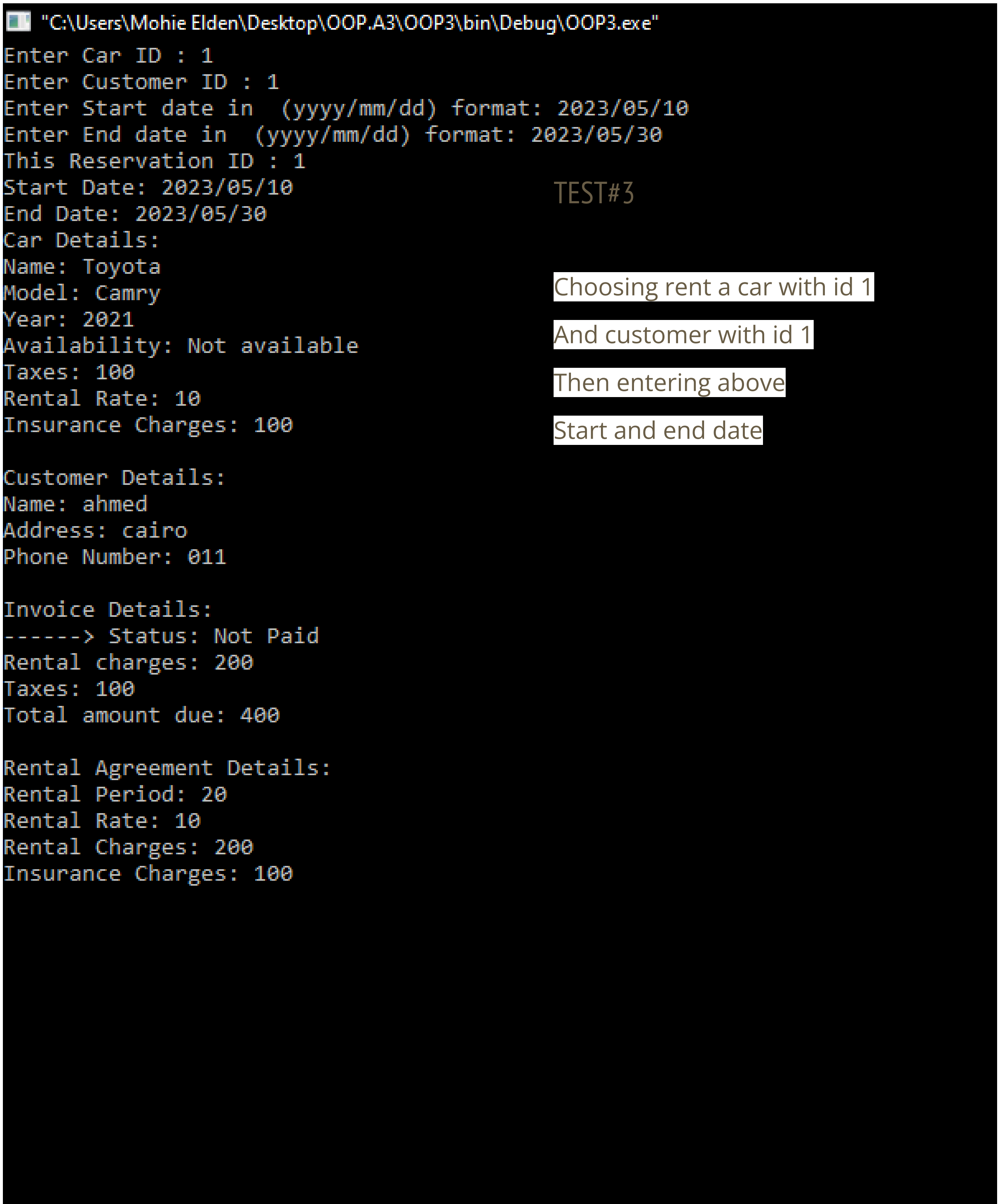


"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
**********************************
Customer ID -----> (1)
Name: ahmed
Address: cairo
Phone Number: 011
**********************************
**********************************
Customer ID -----> (2)
Name: mohy
Address: zagazig
Phone Number: 022
**********************************
**********************************
Customer ID -----> (3)
Name: mohmammed
Address: aswan
Phone Number: 055
**********************************
3 FOUNDED
```

TEST#2

The action performed is adding 3 customers selecting the Print all customers option.



"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Enter Car ID : 1
Enter Customer ID : 1
Enter Start date in  (yyyy/mm/dd) format: 2023/05/10
Enter End date in  (yyyy/mm/dd) format: 2023/05/30
This Reservation ID : 1
Start Date: 2023/05/10
End Date: 2023/05/30
Car Details:
Name: Toyota
Model: Camry
Year: 2021
Availability: Not available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100

Customer Details:
Name: ahmed
Address: cairo
Phone Number: 011

Invoice Details:
------> Status: Not Paid
Rental charges: 200
Taxes: 100
Total amount due: 400

Rental Agreement Details:
Rental Period: 20
Rental Rate: 10
Rental Charges: 200
Insurance Charges: 100
```

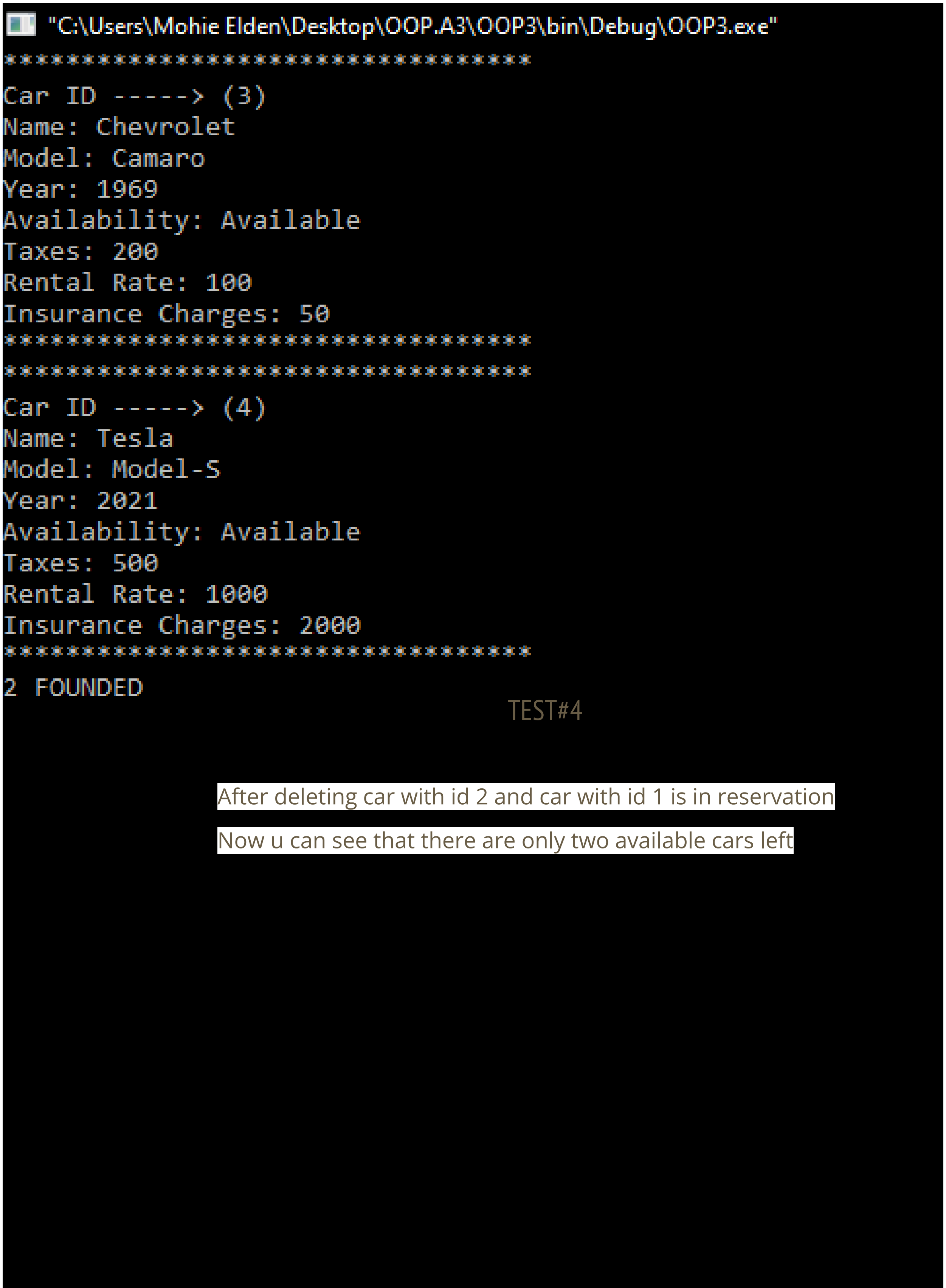
TEST#3

Choosing rent a car with id 1

And customer with id 1

Then entering above

Start and end date



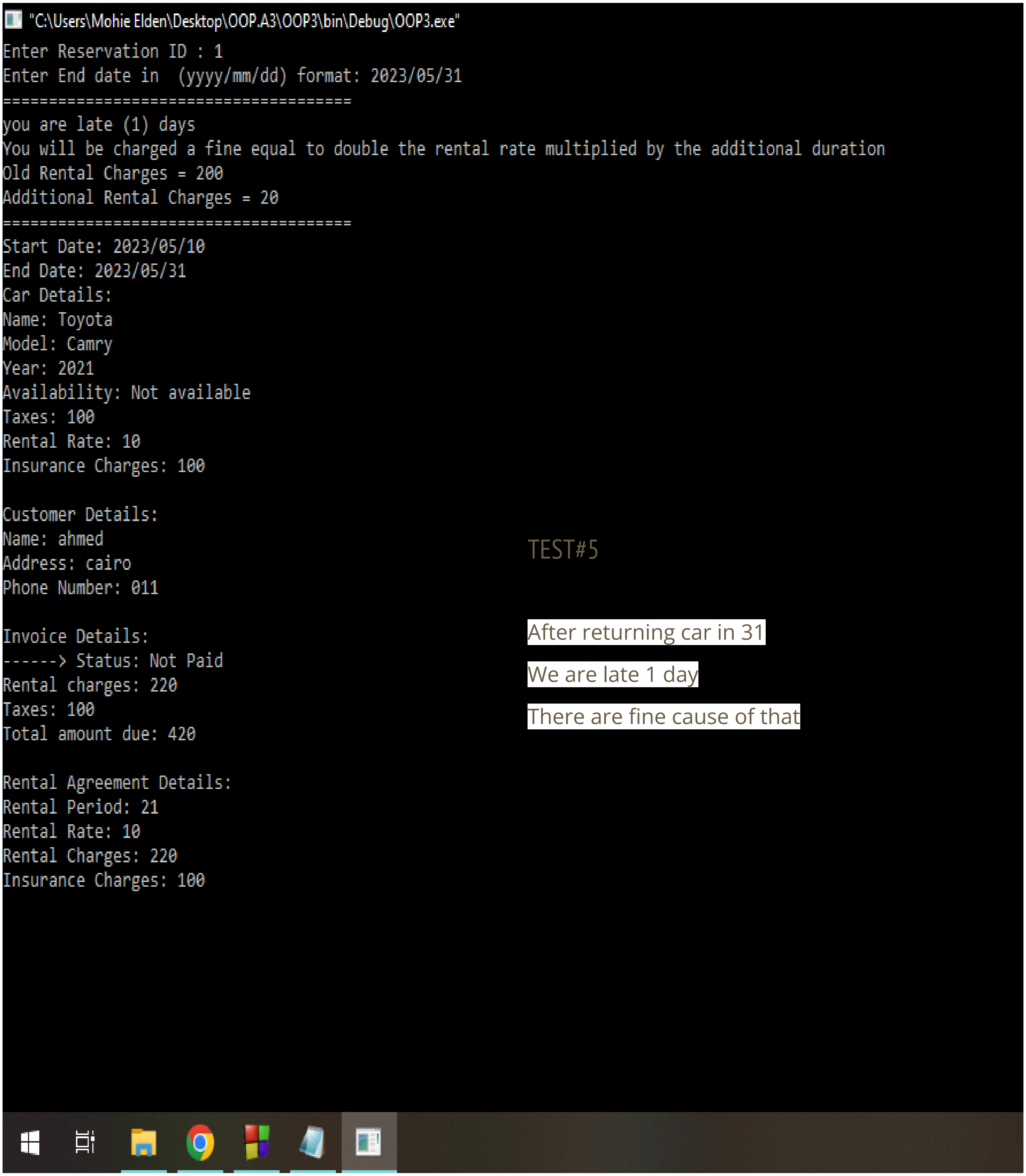
"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
**********************************
Car ID -----> (3)
Name: Chevrolet
Model: Camaro
Year: 1969
Availability: Available
Taxes: 200
Rental Rate: 100
Insurance Charges: 50
**********************************
**********************************
Car ID -----> (4)
Name: Tesla
Model: Model-S
Year: 2021
Availability: Available
Taxes: 500
Rental Rate: 1000
Insurance Charges: 2000
**********************************
2 FOUNDED
```

TEST#4

After deleting car with id 2 and car with id 1 is in reservation

Now u can see that there are only two available cars left



"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

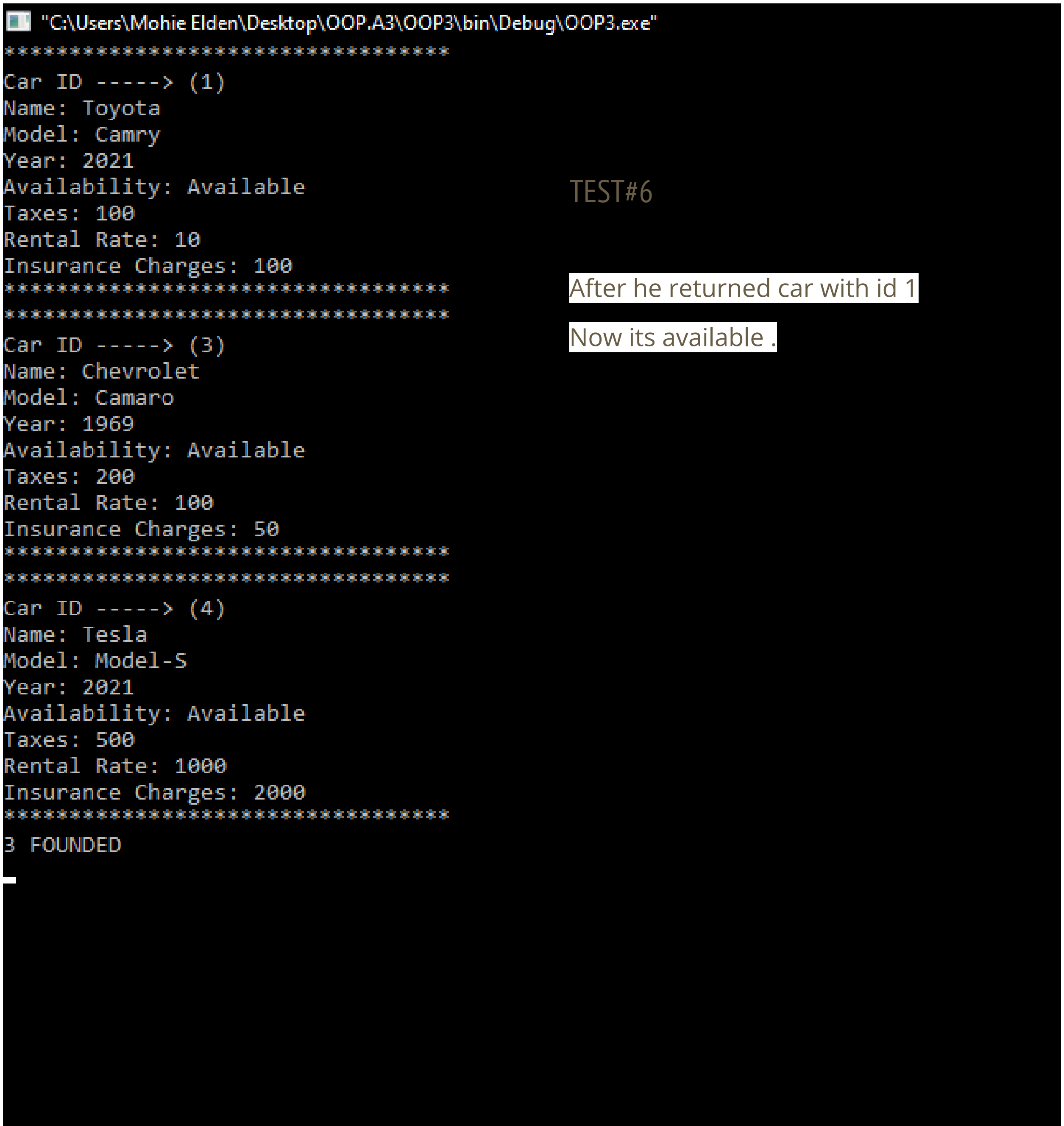
```
Enter Reservation ID : 1
Enter End date in  (yyyy/mm/dd) format: 2023/05/31
======================================
you are late (1) days
You will be charged a fine equal to double the rental rate multiplied by the additional duration
Old Rental Charges = 200
Additional Rental Charges = 20
======================================
Start Date: 2023/05/10
End Date: 2023/05/31
Car Details:
Name: Toyota
Model: Camry
Year: 2021
Availability: Not available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100

Customer Details:
Name: ahmed
Address: cairo
Phone Number: 011

Invoice Details:
------> Status: Not Paid
Rental charges: 220
Taxes: 100
Total amount due: 420

Rental Agreement Details:
Rental Period: 21
Rental Rate: 10
Rental Charges: 220
Insurance Charges: 100
```

TEST#5

After returning car in 31

We are late 1 day

There are fine cause of that



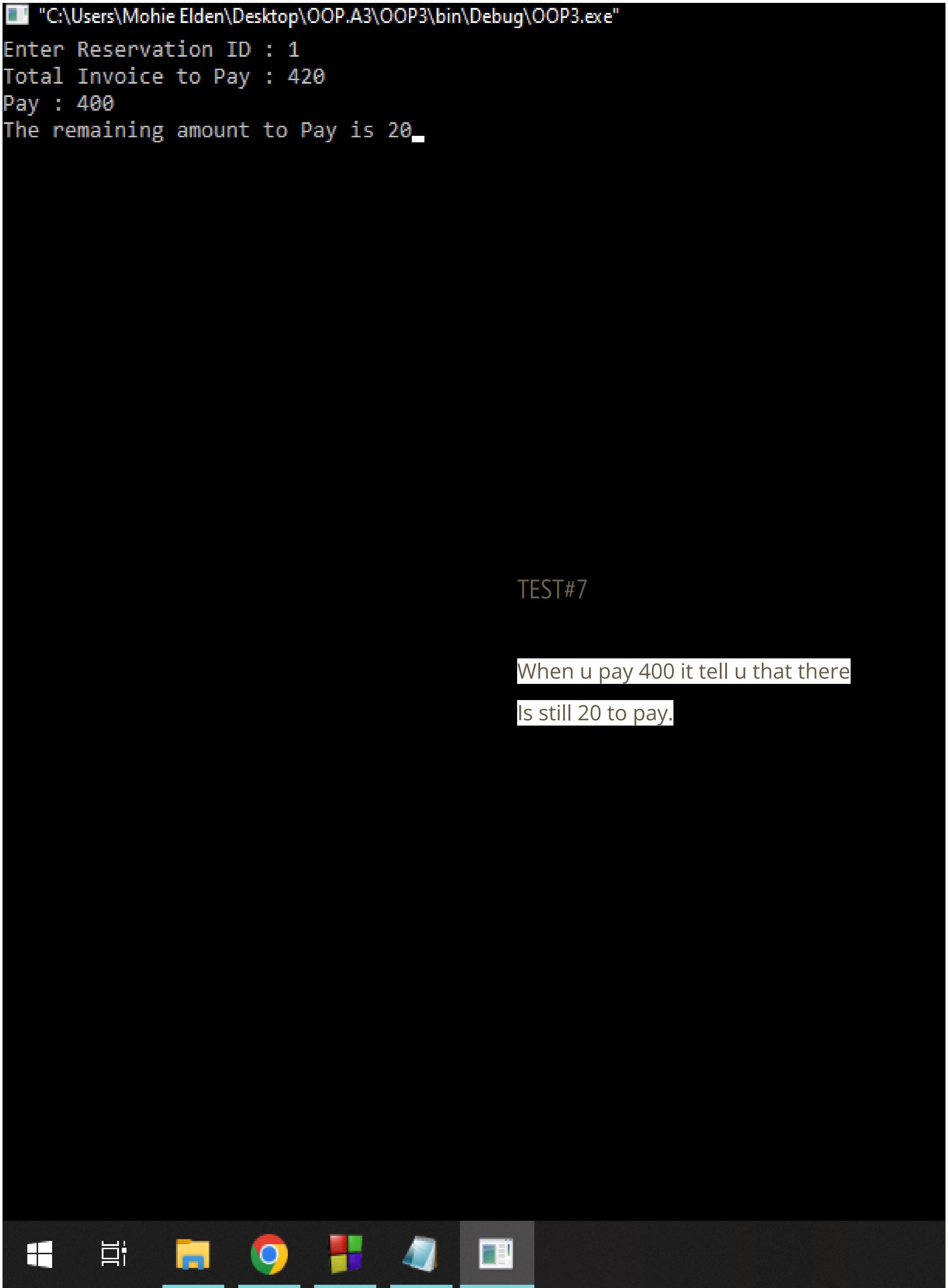
"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
**********************************
Car ID -----> (1)
Name: Toyota
Model: Camry
Year: 2021
Availability: Available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100
**********************************
**********************************
Car ID -----> (3)
Name: Chevrolet
Model: Camaro
Year: 1969
Availability: Available
Taxes: 200
Rental Rate: 100
Insurance Charges: 50
**********************************
**********************************
Car ID -----> (4)
Name: Tesla
Model: Model-S
Year: 2021
Availability: Available
Taxes: 500
Rental Rate: 1000
Insurance Charges: 2000
**********************************
3 FOUNDED
```

TEST#6

After he returned car with id 1

Now its available .

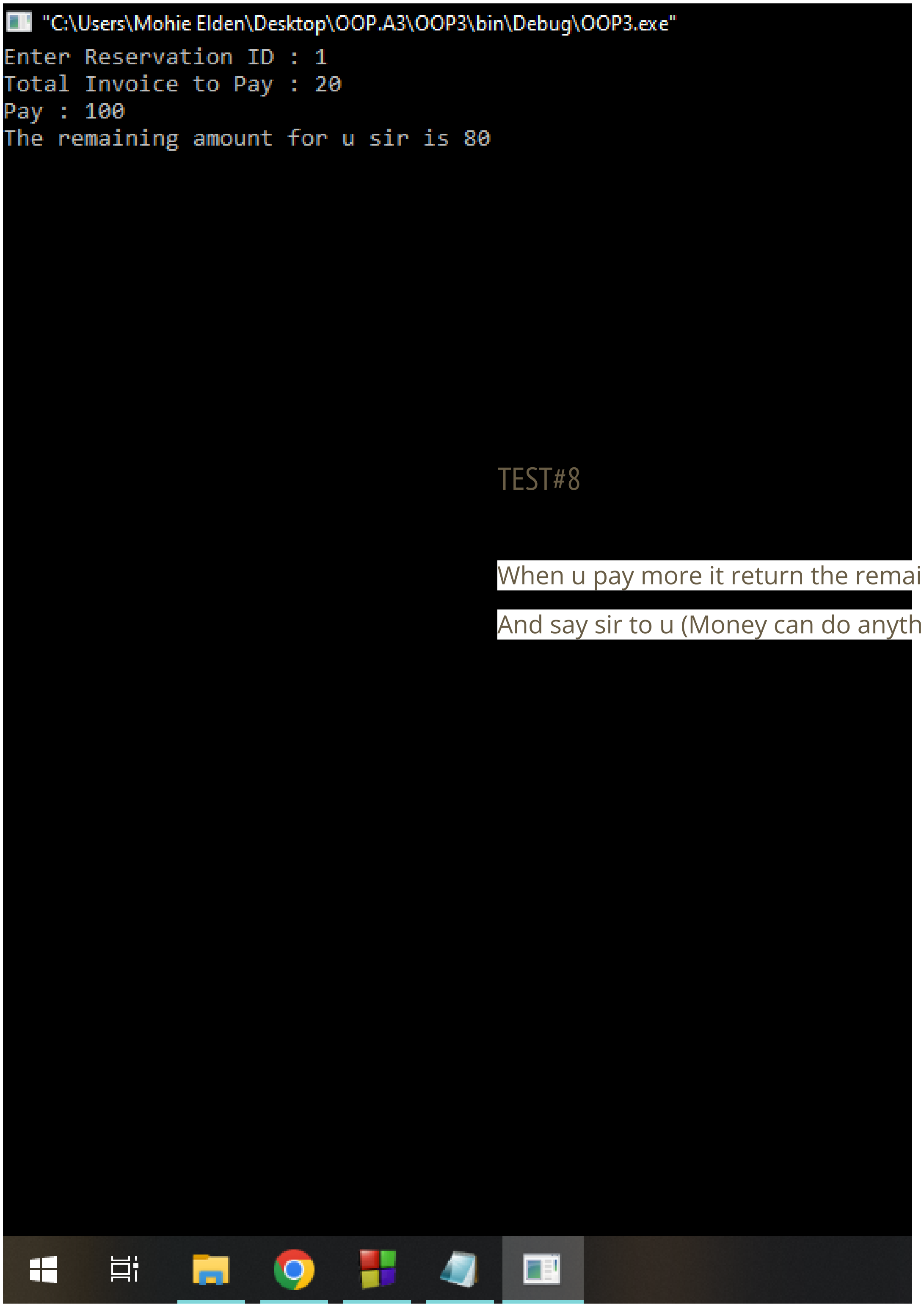

"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Enter Reservation ID : 1
Total Invoice to Pay : 420
Pay : 400
The remaining amount to Pay is 20_
```

TEST#7

When u pay 400 it tell u that there

Is still 20 to pay.

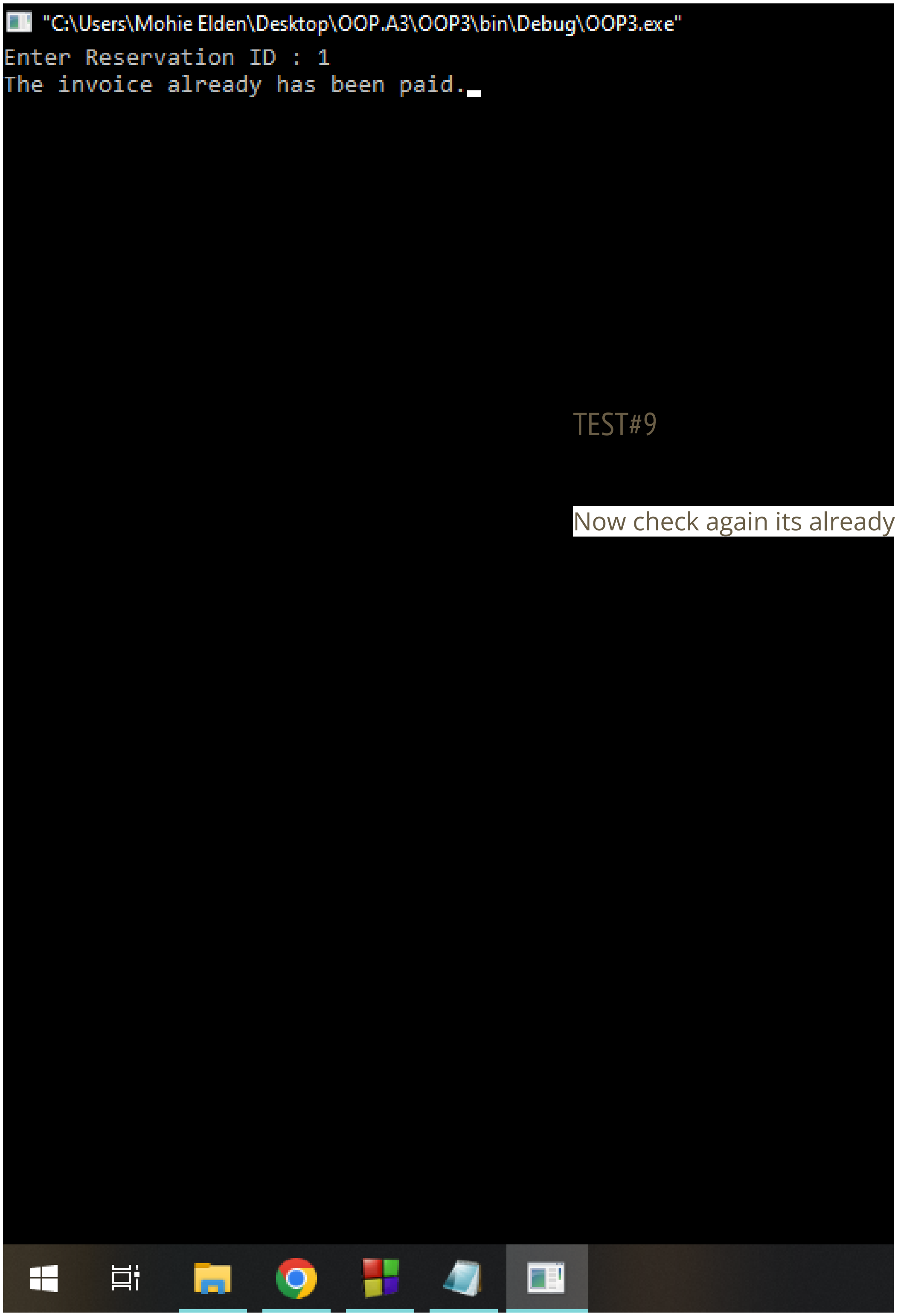

"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Enter Reservation ID : 1
Total Invoice to Pay : 20
Pay : 100
The remaining amount for u sir is 80
```

TEST#8

When u pay more it return the remain

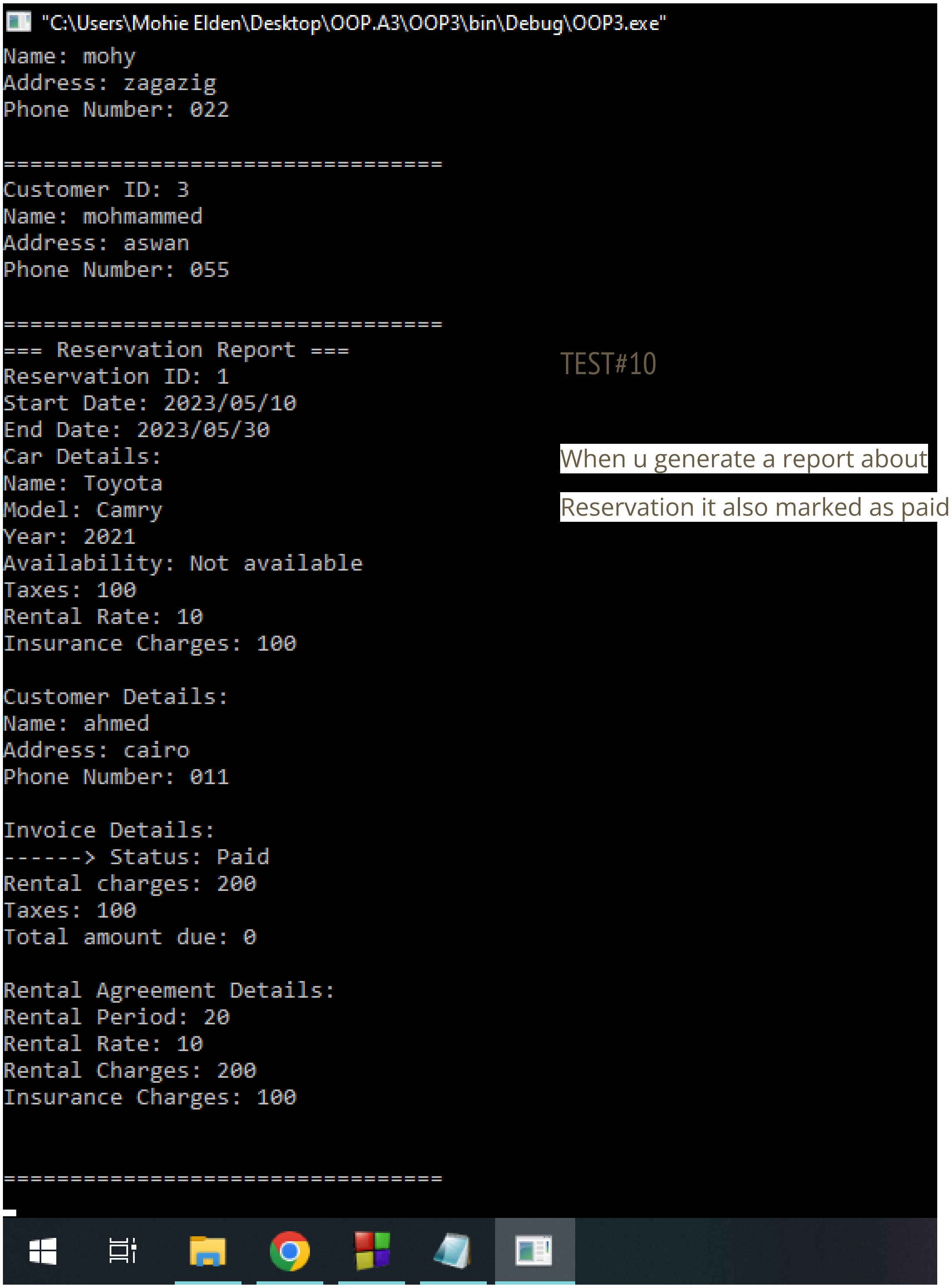
And say sir to u (Money can do anything)


"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Enter Reservation ID : 1
The invoice already has been paid._
```

TEST#9

Now check again its already paid.

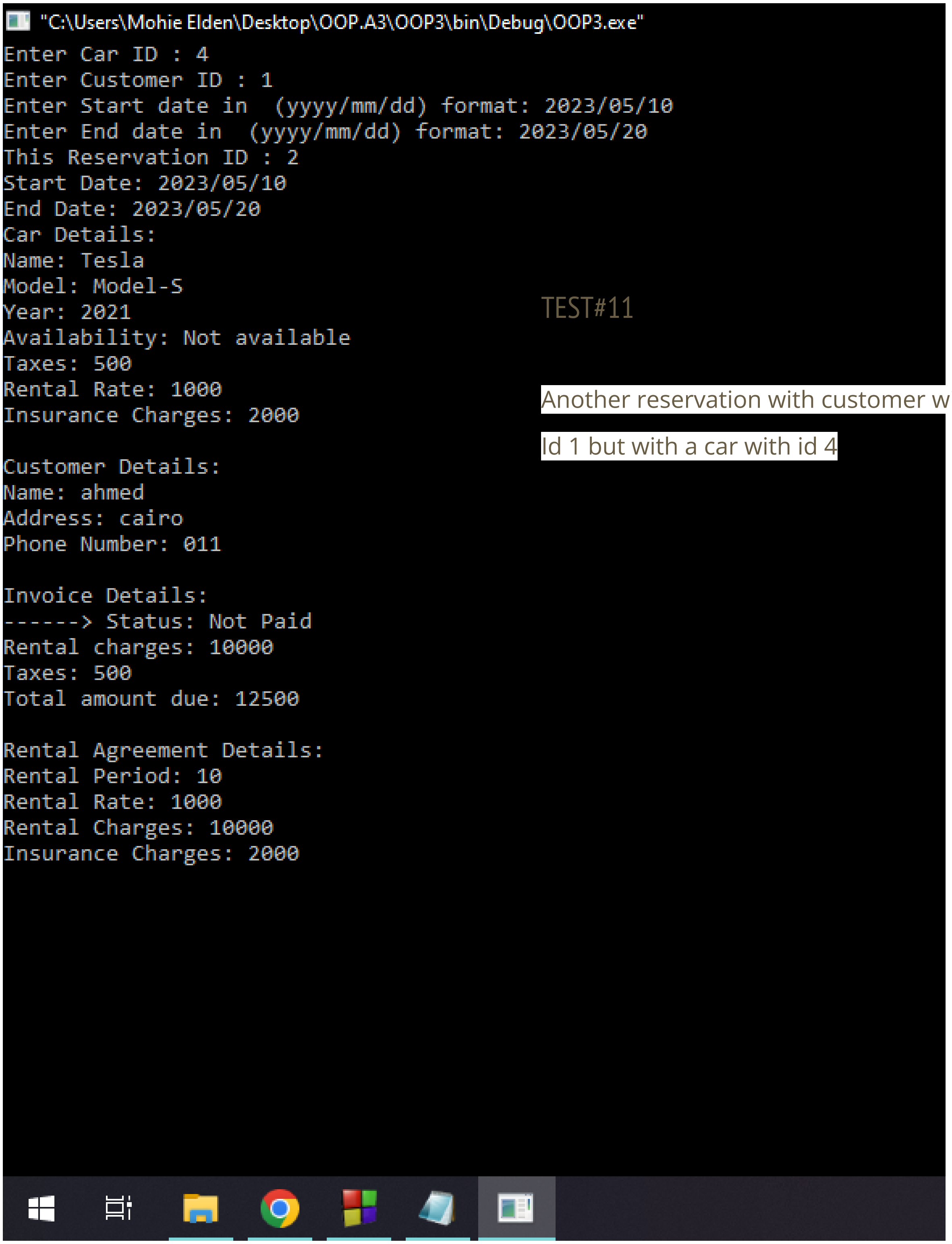

"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Name: mohy
Address: zagazig
Phone Number: 022

=================================
Customer ID: 3
Name: mohmammed
Address: aswan
Phone Number: 055

=================================
=== Reservation Report ===
Reservation ID: 1
Start Date: 2023/05/10
End Date: 2023/05/30
Car Details:
Name: Toyota
Model: Camry
Year: 2021
Availability: Not available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100

Customer Details:
Name: ahmed
Address: cairo
Phone Number: 011

Invoice Details:
------> Status: Paid
Rental charges: 200
Taxes: 100
Total amount due: 0

Rental Agreement Details:
Rental Period: 20
Rental Rate: 10
Rental Charges: 200
Insurance Charges: 100


=================================
```

TEST#10

When u generate a report about

Reservation it also marked as paid.

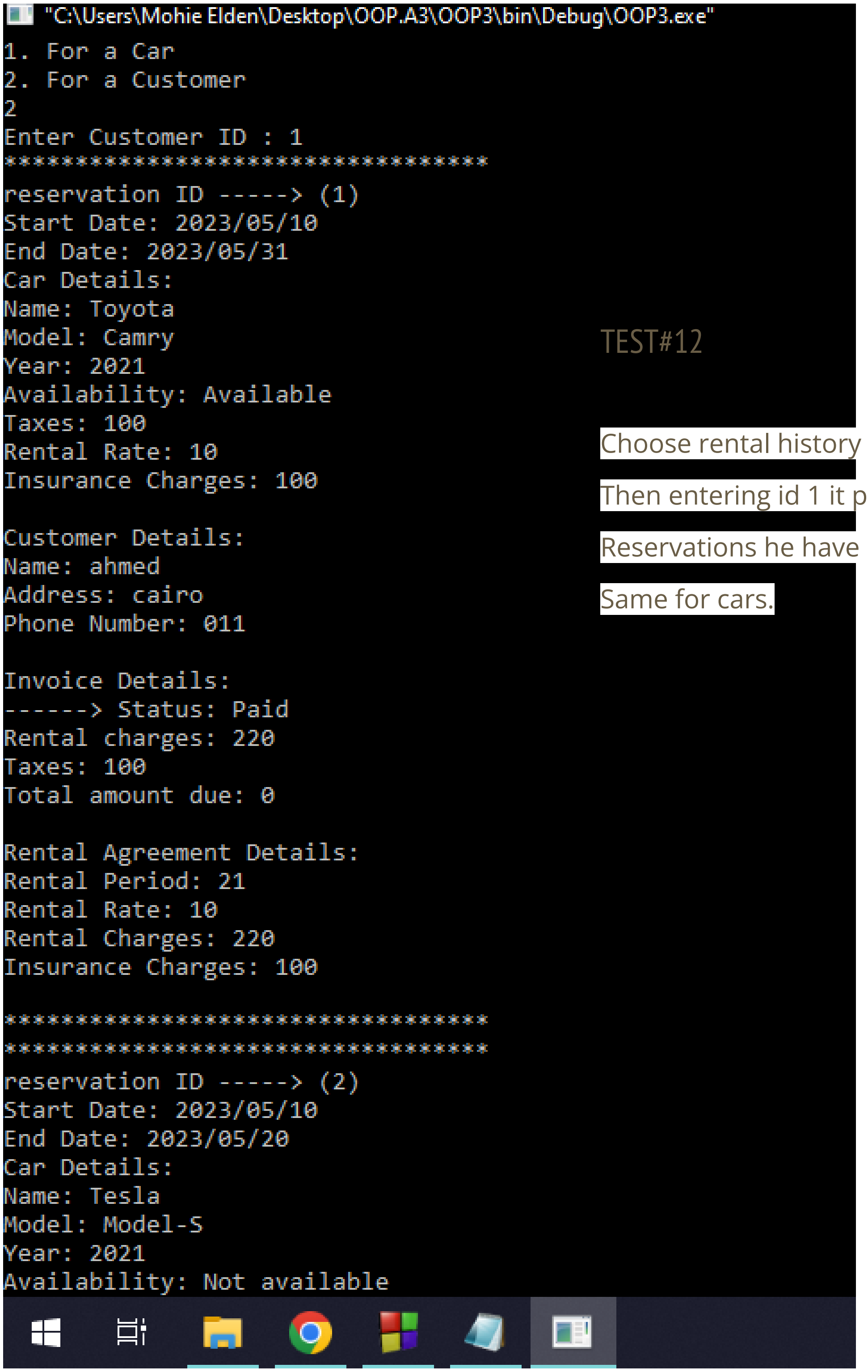


"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
Enter Car ID : 4
Enter Customer ID : 1
Enter Start date in  (yyyy/mm/dd) format: 2023/05/10
Enter End date in  (yyyy/mm/dd) format: 2023/05/20
This Reservation ID : 2
Start Date: 2023/05/10
End Date: 2023/05/20
Car Details:
Name: Tesla
Model: Model-S
Year: 2021
Availability: Not available
Taxes: 500
Rental Rate: 1000
Insurance Charges: 2000

Customer Details:
Name: ahmed
Address: cairo
Phone Number: 011

Invoice Details:
------> Status: Not Paid
Rental charges: 10000
Taxes: 500
Total amount due: 12500

Rental Agreement Details:
Rental Period: 10
Rental Rate: 1000
Rental Charges: 10000
Insurance Charges: 2000
```

TEST#11

Another reservation with customer with Id 1 but with a car with id 4



"C:\Users\Mohie Elden\Desktop\OOP.A3\OOP3\bin\Debug\OOP3.exe"

```
1. For a Car
2. For a Customer
2
Enter Customer ID : 1
**********************************
reservation ID -----> (1)
Start Date: 2023/05/10
End Date: 2023/05/31
Car Details:
Name: Toyota
Model: Camry
Year: 2021
Availability: Available
Taxes: 100
Rental Rate: 10
Insurance Charges: 100

Customer Details:
Name: ahmed
Address: cairo
Phone Number: 011

Invoice Details:
------> Status: Paid
Rental charges: 220
Taxes: 100
Total amount due: 0

Rental Agreement Details:
Rental Period: 21
Rental Rate: 10
Rental Charges: 220
Insurance Charges: 100

**********************************
**********************************
reservation ID -----> (2)
Start Date: 2023/05/10
End Date: 2023/05/20
Car Details:
Name: Tesla
Model: Model-S
Year: 2021
Availability: Not available
```

TEST#12

Choose rental history then for a customer

Then entering id 1 it print the two

Reservations he have done and its the

Same for cars.

Test Cases

Four Cars Test Cases.

1

Toyota

Camry

2021

100

10

100

1

Mercedes-Benz

Camry

2023

500

10

500

1

Chevrolet

Camaro

1969

200

100

50

1

Tesla

Model-S

2021

500

1000

2000

Three Customers Test Cases.

4

ahmed

cairo

011

4

mohy

zagazig

022

4

mohmammed

aswan

055

First Reservation.

7

1

1

2023/05/10

2023/05/30

8

1

2023/05/31

Conclusion

The system utilizes classes such as Car, Customer, Reservation, and Invoice to represent different entities and their interactions. It maintains lists to store the records of cars, customers, and reservations. The code includes methods to add new entries, display existing records, and handle payments.

Some improvements could be made to enhance the system's functionality and robustness. For example, error handling and validation could be implemented to ensure data integrity and prevent crashes due to invalid input. Additionally, the code could benefit from further modularization and separation of concerns to improve code readability and maintainability.

The End