QUESTION :

1. Design and implement a console-based Ride-Hailing system to register drivers/riders, request rides, assign drivers, and compute fares using OOP in Java.

Requirements:

1. Create at least 4 classes:

o User – base with id, name, phone.

o Rider – extends User, preferredPayment, rating.

o Driver – extends User, vehicleNo, vehicleType, availability, rating.

o RideService – request matching, fare calculation, trip history.

2. Each class must include:

o ≥4 instance/static variables.

o A constructor to initialize values.

o ≥5 methods (getters/setters, requestRide(), assignDriver(), completeRide(), fare()).

3. Demonstrate OOPS Concepts:

o Inheritance → Rider & Driver from User.

o Method Overloading → fare() by distance-only or distance+time+surcharge.

o Method Overriding → different availability()/display() per role.

o Polymorphism → store users as User and resolve behavior at runtime.

o Encapsulation → protect ratings and availability.

4. Write a Main class (RideAppMain) to test:

o Register riders/drivers, request rides.

o Auto-assign nearest available driver, complete rides.

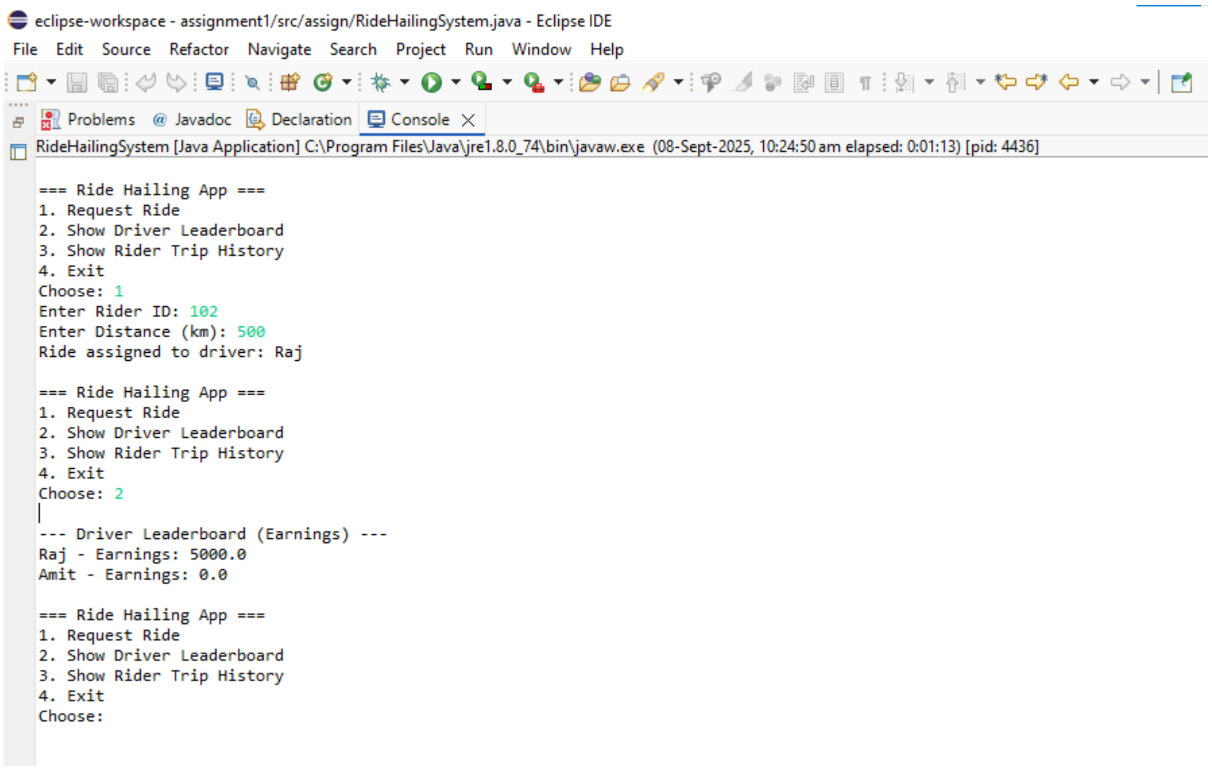
o Print daily earnings, driver leaderboards, rider histories.

SOURCE CODE :

```
package assign;
import java.util.*;

class User {
    protected int id;
    protected String name;
    protected String phone;

    public User(int id, String name, String phone) {
        this.id = id;
        this.name = name;
        this.phone = phone;
    }

    public void display() {
        System.out.println("User ID: " + id + ", Name: " +
name + ", Phone: " + phone);
    }
}

class Rider extends User {
    private String preferredPayment;
    private double rating;
    private List<String> tripHistory;

    public Rider(int id, String name, String phone, String
preferredPayment) {
        super(id, name, phone);
        this.preferredPayment = preferredPayment;
        this.rating = 5.0;
        this.tripHistory = new ArrayList<>();
    }

    public void addTrip(String trip) {
        tripHistory.add(trip);
    }

    @Override
    public void display() {
        System.out.println("Rider: " + name + " | Payment: 
" + preferredPayment + " | Rating: " + rating);
    }

    public List<String> getTripHistory() {
        return tripHistory;
    }

    public void setRating(double rating) {
        if (rating >= 0 && rating <= 5) {
            this.rating = rating;
        }
    }

    public double getRating() {
        return rating;
    }
}

class Driver extends User {
    private String vehicleNo;
    private String vehicleType;
    private boolean available;
    private double rating;
    private double earnings;

    public Driver(int id, String name, String phone, 
String vehicleNo, String vehicleType) {
        super(id, name, phone);
        this.vehicleNo = vehicleNo;
        this.vehicleType = vehicleType;
        this.available = true;
        this.rating = 5.0;
        this.earnings = 0.0;
    }

    public boolean isAvailable() {
        return available;
    }

    public void setAvailable(boolean available) {
        this.available = available;
    }

    public void addEarnings(double fare) {
        earnings += fare;
    }

    public double getEarnings() {
        return earnings;
    }

    @Override
    public void display() {
        System.out.println("Driver: " + name + " | 
Vehicle: " + vehicleType + " (" + vehicleNo + ") | 
Available: " + available + " | Rating: " + rating + " | 
Earnings: " + earnings);
    }
}

class RideService {
    private List<Rider> riders;
    private List<Driver> drivers;

    public RideService() {
        riders = new ArrayList<>();
        drivers = new ArrayList<>();
    }

    public void registerRider(Rider rider) {
        riders.add(rider);
    }

    public void registerDriver(Driver driver) {
        drivers.add(driver);
    }

    public Driver assignDriver() {
        for (Driver d : drivers) {
            if (d.isAvailable()) {
                d.setAvailable(false);
                return d;
            }
        }
        return null;
    }

    public double fare(double distance) {
        return distance * 10;
    }

    public double fare(double distance, double time,
double surcharge) {
        return (distance * 10) + (time * 2) + surcharge;
    }

    public void requestRide(Rider rider, double distance)
{
        Driver driver = assignDriver();
        if (driver == null) {
            System.out.println("No drivers available right
now.");
            return;
        }
        System.out.println("Ride assigned to driver: " +
driver.name);
        double tripFare = fare(distance);
        driver.addEarnings(tripFare);
        rider.addTrip("Driver: " + driver.name + " | Fare:
" + tripFare);
        completeRide(driver);
    }

    public void completeRide(Driver driver) {
        driver.setAvailable(true);
    }

    public void showDriverLeaderboard() {
        System.out.println("\n--- Driver Leaderboard
(Earnings) ---");
        drivers.sort((a, b) ->
Double.compare(b.getEarnings(), a.getEarnings()));
        for (Driver d : drivers) {
```

```
            System.out.println(d.name + " - Earnings: " + 
d.getEarnings());
        }
    }

    public void showRiderHistory(Rider rider) {
        System.out.println("\n--- Trip History for " + 
rider.name + " ---");
        for (String trip : rider.getTripHistory()) {
            System.out.println(trip);
        }
    }
}

public class RideHailingSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        RideService service = new RideService();

        service.registerDriver(new Driver(1, "Raj", 
"999111222", "MH12AB1234", "Car"));
        service.registerDriver(new Driver(2, "Amit", 
"888111222", "MH14XY5678", "Bike"));

        Rider r1 = new Rider(101, "Kiran", "777111222", 
"UPI");
        Rider r2 = new Rider(102, "Sneha", "666111222", 
"Cash");
        service.registerRider(r1);
        service.registerRider(r2);

        while (true) {
            System.out.println("\n=== Ride Hailing App 
===");
            System.out.println("1. Request Ride");
            System.out.println("2. Show Driver 
Leaderboard");
            System.out.println("3. Show Rider Trip 
History");
            System.out.println("4. Exit");
            System.out.print("Choose: ");
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter Rider ID: ");
                    int riderId = sc.nextInt();
                    Rider rider = (riderId == 101) ? r1 : 
r2;

                    System.out.print("Enter Distance (km): 
");

                    double dist = sc.nextDouble();
                    service.requestRide(rider, dist);
                    break;

                case 2:
                    service.showDriverLeaderboard();
                    break;

                case 3:
                    System.out.print("Enter Rider ID: ");
                    int rid = sc.nextInt();
                    Rider r = (rid == 101) ? r1 : r2;
                    service.showRiderHistory(r);
                    break;

                case 4:
                    System.out.println("Exiting... Thank 
you!");

                    sc.close();
                    return;

                default:
                    System.out.println("Invalid option!");
            }
        }
    }
}
```

OUTPUT :


eclipse-workspace - assignment1/src/assign/RideHailingSystem.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems @ Javadoc Declaration Console

RideHailingSystem [Java Application] C:\Program Files\Java\jre1.8.0_74\bin\javaw.exe (08-Sept-2025, 10:24:50 am elapsed: 0:01:13) [pid: 4436]

```
=== Ride Hailing App ===
1. Request Ride
2. Show Driver Leaderboard
3. Show Rider Trip History
4. Exit
Choose: 1
Enter Rider ID: 102
Enter Distance (km): 500
Ride assigned to driver: Raj

=== Ride Hailing App ===
1. Request Ride
2. Show Driver Leaderboard
3. Show Rider Trip History
4. Exit
Choose: 2

--- Driver Leaderboard (Earnings) ---
Raj - Earnings: 5000.0
Amit - Earnings: 0.0

=== Ride Hailing App ===
1. Request Ride
2. Show Driver Leaderboard
3. Show Rider Trip History
4. Exit
Choose:
```