

ACADEMIC YEAR: 2024-25

Name & Register No of the Candidate: DHARUN RAM P & 717824L210						
Course Code & Title: 23CSR306 & JAVA PROGRAMMING						
Date of Issue: 14.09.2025				Date of Submission:		
Year/Dept./Sem/Section: II / ECE / III / B						
Assignment: I						
Reference(s):						
Marks Details						
Q. No						Total (100)
COs	CO1	CO2				
Marks Obtained						

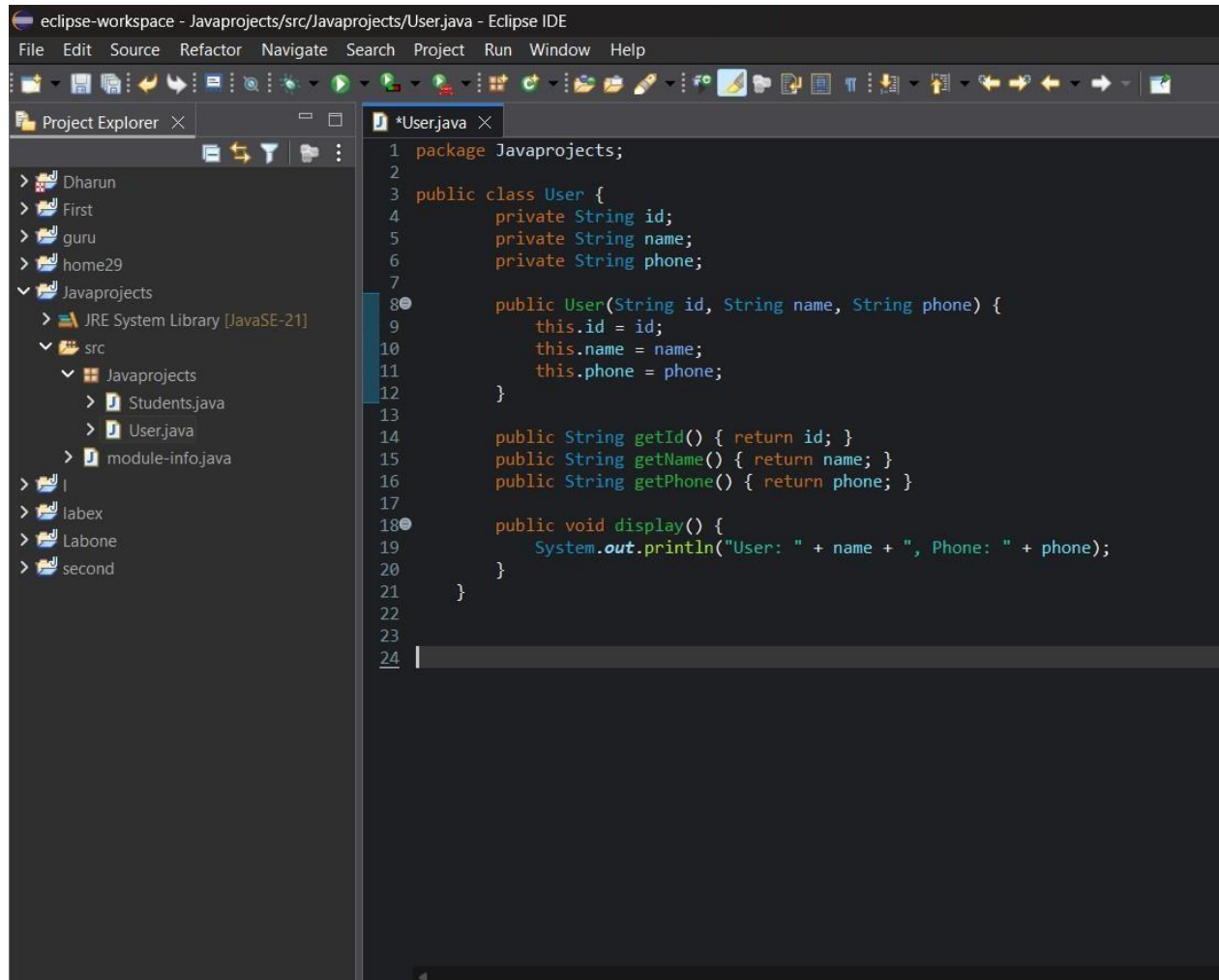
Course In-charge

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:

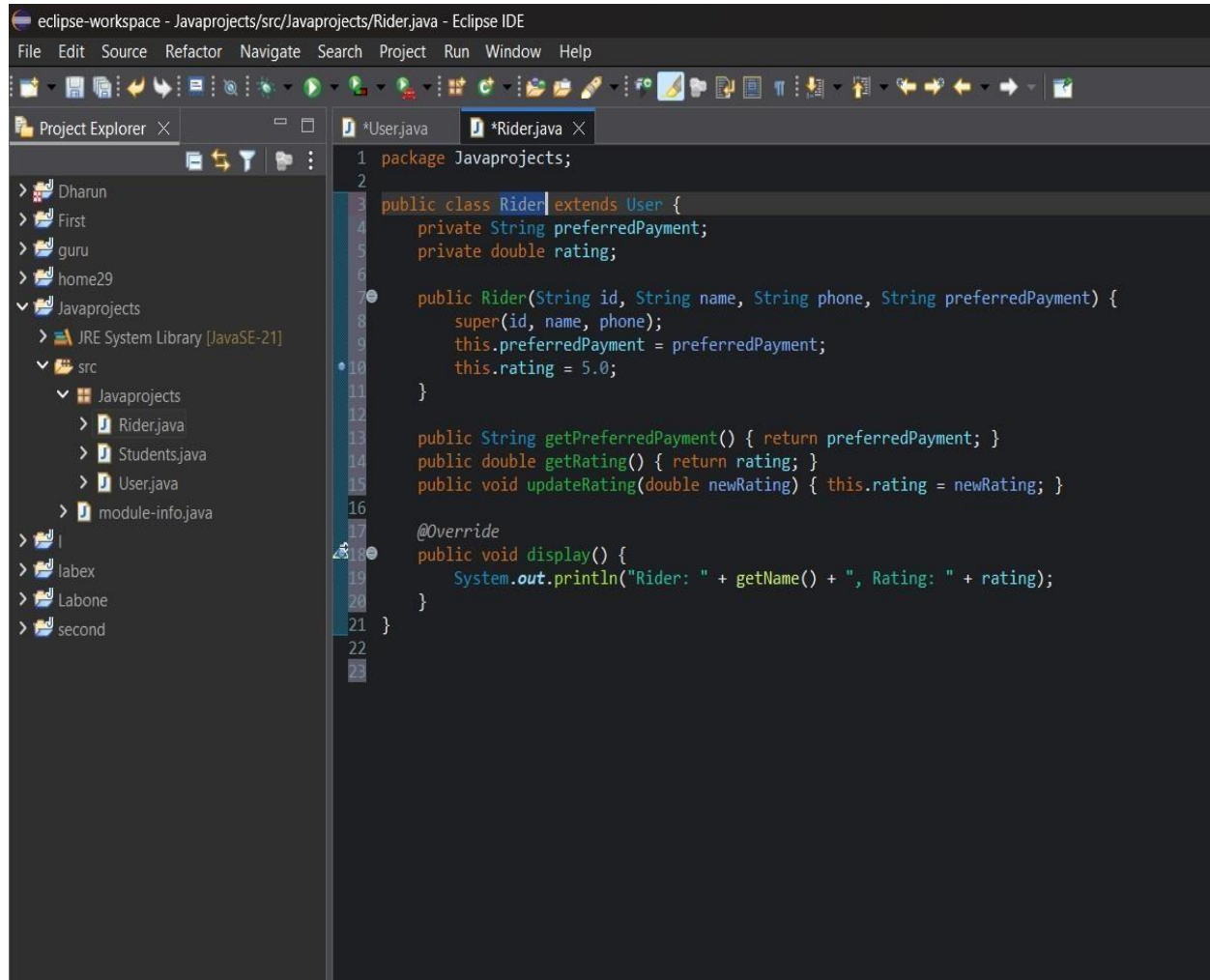
1. Create at least 4 classes:
 - o User – base with id, name, phone.
 - o Rider – extends User, preferredPayment, rating.
 - o Driver – extends User, vehicleNo, vehicleType, availability, rating.
 - o RideService – request matching, fare calculation, trip history.
2. Each class must include:
 - o ≥ 4 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:



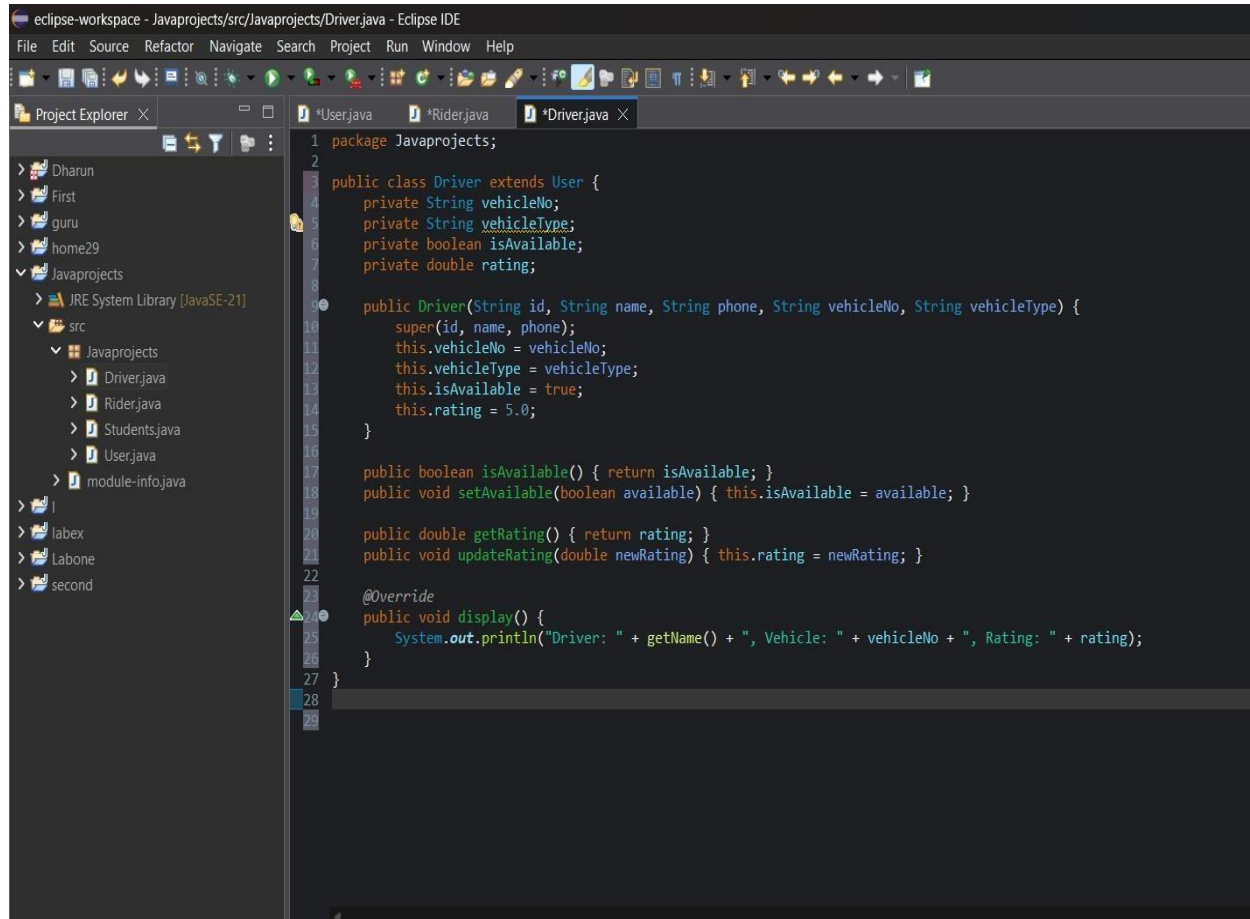
The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - Javaprojects/src/Javaprojects/User.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations, editing, and running. The Project Explorer on the left shows a project named "Javaprojects" with a source folder "src" containing "Students.java", "User.java", and "module-info.java". The main editor displays the code for "User.java".

```
1 package Javaprojects;
2
3 public class User {
4     private String id;
5     private String name;
6     private String phone;
7
8     public User(String id, String name, String phone) {
9         this.id = id;
10        this.name = name;
11        this.phone = phone;
12    }
13
14    public String getId() { return id; }
15    public String getName() { return name; }
16    public String getPhone() { return phone; }
17
18    public void display() {
19        System.out.println("User: " + name + ", Phone: " + phone);
20    }
21 }
22
23
24
```



The screenshot shows the Eclipse IDE interface. The title bar reads "eclipse-workspace - Javaprojects/src/Javaprojects/Rider.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development. The Project Explorer on the left shows a project named "Javaprojects" with a "src" folder containing "Rider.java", "Students.java", "User.java", and "module-info.java". The main editor displays the code for "Rider.java".

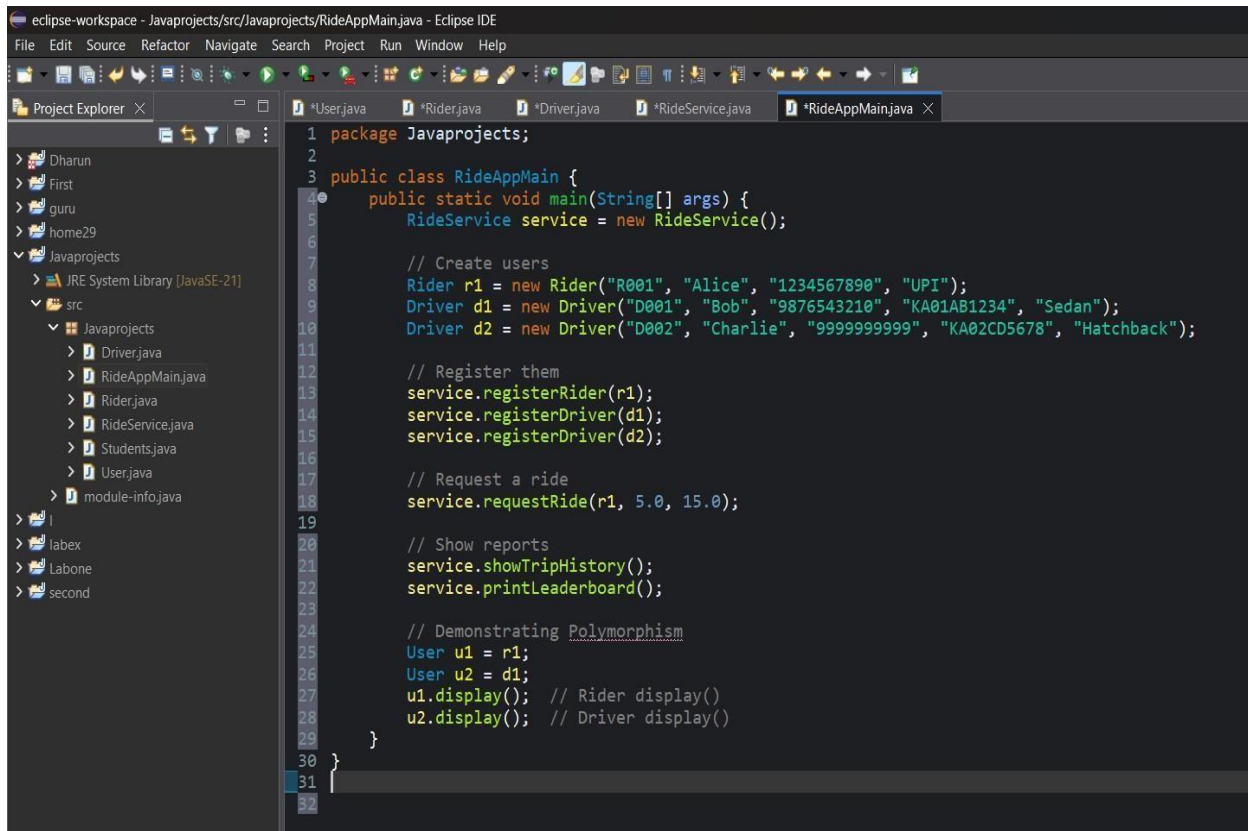
```
1 package Javaprojects;
2
3 public class Rider extends User {
4     private String preferredPayment;
5     private double rating;
6
7     public Rider(String id, String name, String phone, String preferredPayment) {
8         super(id, name, phone);
9         this.preferredPayment = preferredPayment;
10        this.rating = 5.0;
11    }
12
13    public String getPreferredPayment() { return preferredPayment; }
14    public double getRating() { return rating; }
15    public void updateRating(double newRating) { this.rating = newRating; }
16
17    @Override
18    public void display() {
19        System.out.println("Rider: " + getName() + ", Rating: " + rating);
20    }
21 }
22
23
```



```
1 package Javaprojects;
2
3 public class Driver extends User {
4     private String vehicleNo;
5     private String vehicleType;
6     private boolean isAvailable;
7     private double rating;
8
9     public Driver(String id, String name, String phone, String vehicleNo, String vehicleType) {
10         super(id, name, phone);
11         this.vehicleNo = vehicleNo;
12         this.vehicleType = vehicleType;
13         this.isAvailable = true;
14         this.rating = 5.0;
15     }
16
17     public boolean isAvailable() { return isAvailable; }
18     public void setAvailable(boolean available) { this.isAvailable = available; }
19
20     public double getRating() { return rating; }
21     public void updateRating(double newRating) { this.rating = newRating; }
22
23     @Override
24     public void display() {
25         System.out.println("Driver: " + getName() + ", Vehicle: " + vehicleNo + ", Rating: " + rating);
26     }
27 }
28
29
```

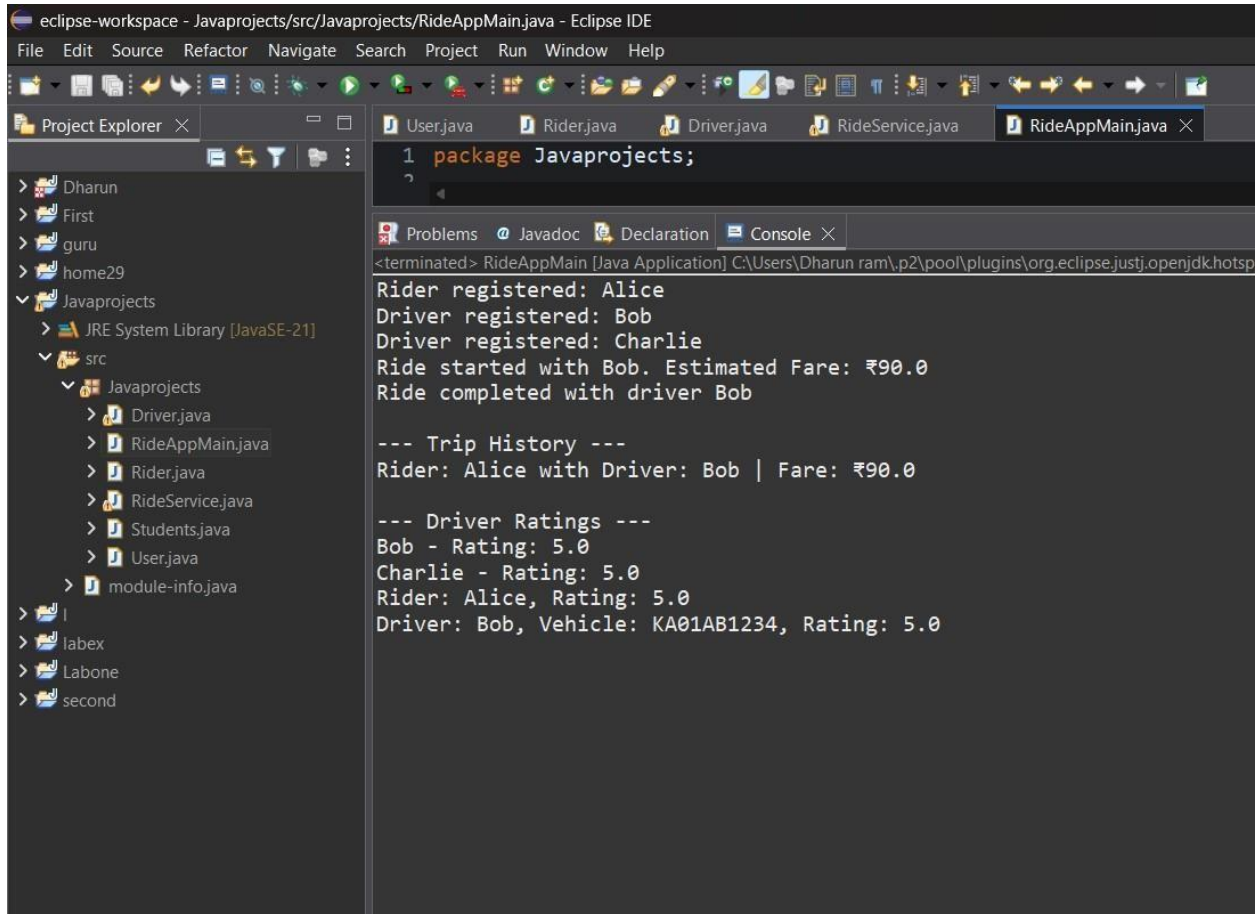
```
eclipse-workspace - Javaprojects/src/Javaprojects/RideService.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer Explorer
Dharun
JRE System Library [JavaSE-21]
src
Welcome
module-info.java
First
JRE System Library [JavaSE-21]
src
guru
JRE System Library [JavaSE-21]
src
home29
JRE System Library [JavaSE-21]
src
Javaprojects
JRE System Library [JavaSE-21]
src
Javaprojects
Content.java
Driver.java
Movie.java
Plan.java
RideAppMain.java
Rider.java
RideService.java
Series.java
StreamingAppMain.java
StreamingService.java
Students.java
User.java
Users.java
RideService.java
1 package Javaprojects;
2
3 import java.util.*;
4
5 public class RideService {
6     private List<Rider> riders = new ArrayList<>();
7     private List<Driver> drivers = new ArrayList<>();
8     private List<String> tripHistory = new ArrayList<>();
9
10    public void registerRider(Rider rider) {
11        riders.add(rider);
12        System.out.println("Rider registered: " + rider.getName());
13    }
14
15    public void registerDriver(Driver driver) {
16        drivers.add(driver);
17        System.out.println("Driver registered: " + driver.getName());
18    }
19
20    public Driver findAvailableDriver() {
21        for (Driver driver : drivers) {
22            if (driver.isAvailable()) return driver;
23        }
24        return null;
25    }
26
27    public void requestRide(Rider rider, double distance, double time) {
28        Driver driver = findAvailableDriver();
29        if (driver == null) {
30            System.out.println("No available drivers right now.");
31            return;
32        }
33
34        driver.setAvailable(false); // assign driver
35        double fare = fare(distance); // Method Overloading
36        double fullFare = fare(distance, time, 10); // Overloaded version
37        System.out.println("Ride started with " + driver.getName() + ". Estimated Fare: ₹" + fullFare);
38
39        tripHistory.add("Rider: " + rider.getName() + " with Driver: " + driver.getName() + " | Fare: ₹" + fullFare);
40
41        completeRide(driver);
42    }
43
44    public void completeRide(Driver driver) {
45        driver.setAvailable(true);
46        System.out.println("Ride completed with driver " + driver.getName());
47    }
48
49    // Method Overloading
50    public double fare(double distance) {
```

```
49    // Method Overloading
50    public double fare(double distance) {
51        return distance * 10; // base fare
52    }
53
54    public double fare(double distance, double time, double surcharge) {
55        return (distance * 10) + (time * 2) + surcharge;
56    }
57
58    public void showTripHistory() {
59        System.out.println("\n--- Trip History ---");
60        for (String trip : tripHistory) {
61            System.out.println(trip);
62        }
63    }
64
65    public void printLeaderboard() {
66        System.out.println("\n--- Driver Ratings ---");
67        drivers.sort((d1, d2) -> Double.compare(d2.getRating(), d1.getRating()));
68        for (Driver d : drivers) {
69            System.out.println(d.getName() + " - Rating: " + d.getRating());
70        }
71    }
72 }
73
74
```

```
1 package Javaprojects;
2
3 public class RideAppMain {
4     public static void main(String[] args) {
5         RideService service = new RideService();
6
7         // Create users
8         Rider r1 = new Rider("R001", "Alice", "1234567890", "UPI");
9         Driver d1 = new Driver("D001", "Bob", "9876543210", "KA01AB1234", "Sedan");
10        Driver d2 = new Driver("D002", "Charlie", "9999999999", "KA02CD5678", "Hatchback");
11
12        // Register them
13        service.registerRider(r1);
14        service.registerDriver(d1);
15        service.registerDriver(d2);
16
17        // Request a ride
18        service.requestRide(r1, 5.0, 15.0);
19
20        // Show reports
21        service.showTripHistory();
22        service.printLeaderboard();
23
24        // Demonstrating Polymorphism
25        User u1 = r1;
26        User u2 = d1;
27        u1.display(); // Rider display()
28        u2.display(); // Driver display()
29    }
30 }
31
32
```

OUTPUT:



```
eclipse-workspace - Javaprojects/src/Javaprojects/RideAppMain.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer
Dharun
First
guru
home29
Javaprojects
  JRE System Library [JavaSE-21]
  src
    Javaprojects
      Driver.java
      RideAppMain.java
      Rider.java
      RideService.java
      Students.java
      User.java
    module-info.java
labex
Labone
second

1 package Javaprojects;

<terminated> RideAppMain [Java Application] C:\Users\Dharun ram\p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
Rider registered: Alice
Driver registered: Bob
Driver registered: Charlie
Ride started with Bob. Estimated Fare: ₹90.0
Ride completed with driver Bob

--- Trip History ---
Rider: Alice with Driver: Bob | Fare: ₹90.0

--- Driver Ratings ---
Bob - Rating: 5.0
Charlie - Rating: 5.0
Rider: Alice, Rating: 5.0
Driver: Bob, Vehicle: KA01AB1234, Rating: 5.0
```

GITHUB REPOSITORY LINK :

<https://github.com/dharunram533/Java-mini-project/tree/main>