Problem Definition:

Design a Ride sharing application where drivers can offer rides (origin, destination, no of seats), any rider can request rides (origin, destination, no of seats).
There is an algo to choose to calculate Ride amount charged for a given ride based on distance and no of seats

1. When the ride closes, show the amount charged to the rider.
2. Ride amount if No of seats >= 2 : No of kilometers * No of seats * 0.75 * Amount Charged per KM
3. Ride amount if No of seats = 1 : No of kilometers * Amount Charged per KM

The program should take as input two or more drivers and a set of riders requesting rides. Multiple rides can happen simultaneously.

Assumptions :

1. Assume Amount charged per KM = 20
2. No of Kilometers = destination - origin
3. All values are Integer

Test Case:

A requesting for ride R1
INPUT: A requests 50, 60, 1,
OUTPUT: Ride Amount: 10 * 20 (Amount/Km) = 200

A requesting for ride R1
INPUT: A requests 50, 60, 2,
OUTPUT: Ride Amount: 10 * 2 * 0.75 * 20 (Amount/Km) = 300

Bonus:

- Upgrade the rider to a preferred rider if he has done in more than 10 rides.
- Preferred Rider amount if No of seats >= 2 : No of kilometers * No of seats * 0.5 * Amount Charged per KM
- Preferred Ride amount if No of seats = 1 : No of kilometers * Amount Charged per KM * 0.75

Functionalities expected:

- Add Driver(name)
- Add Rider(name)
- Create Ride (id, origin, destination, no of seats)
- Create/Update Ride (id, origin, destination, no of seats)

- Withdraw Ride (id)
- Close ride and return ride amount charged

Expectations:

1. Create the sample data yourself. You can put it into a file, test case or main driver program itself.
2. The code should be demo-able. Either by using the main driver program or test cases.
3. The code should be modular. The code should have the basic OO design. Please do not jam in the responsibilities of one class into another.
4. The code should be extensible. Wherever applicable, use interfaces and contracts between different methods. It should be easy to add/remove functionality without rewriting the entire codebase.
5. The code should handle edge cases properly and fail gracefully.
6. The code should be legible, readable and DRY.
7. Database integration is not required.

Guidelines:

1. Please do not access the internet for anything EXCEPT syntax.
2. You are free to use the language and IDE of your choice.
3. The entire code should be your own.