



## MOVEINSYNC – Case Study

### (Ride-Sharing Platform)

#### Introduction:

The Ride-Sharing Platform is a Java-based program developed to simulate a ride-sharing system. It provides a user-friendly interface for travelers to request rides, companions to track their rides, and an admin to manage the system. This report provides a detailed overview of the platform, including its features, functionality, development environment, and future enhancements.

#### Development Environment:

- **IDE:** IntelliJ IDEA Ultimate Edition
- **Java Development Kit (JDK):** Version 11.0.12
- **Dependencies:** No external dependencies are required as the program utilizes core Java libraries.

#### Key Features:

##### 1. User Classes:

- **User:** Represents a base class with attributes like name, phone number, and user ID.
- **Traveler:** Extends the User class and allows travelers to request rides, share ride details, and track their rides.
- **TravelerCompanion:** Extends the User class and enables companions to track rides, receive notifications, and provide feedback.

## 2. **Admin Class:**

- Manages the system and performs administrative tasks such as viewing all rides, viewing completed rides, and receiving feedback from users.

## 3. **Ride Class:**

- Encapsulates ride details such as trip ID, driver information, cab details, and ride completion status.
- Provides methods to set and check if the ride is completed.

# User Interface and Functionality:

## 1. **Traveler Interface:**

- Allows users to input their details and ride information.
- Facilitates sharing ride details with the admin and adding rides to the traveler's list.
- Enables tracking of rides by the traveler.

## 2. **TravelerCompanion Interface:**

- Similar to the traveler interface, companions can input their details and receive notifications about ride status.
- They can also provide feedback about the ride experience.

## 3. **Admin Interface:**

- Provides functionalities for managing the system, including viewing all rides, completed rides, and user feedback.
- Allows the admin to receive feedback from users and track overall user satisfaction.

## 4. **Error Handling:**

- The program includes error handling for invalid inputs, ensuring numeric values for phone numbers and cab numbers.
- It prompts users to re-enter data if invalid inputs are provided.

# Functionality Flow:

## 1. **User Registration:**

- Users input their details, including name, phone number, and user ID.
- Ride details such as ride ID, driver information, and cab number are also provided.

## 2. **Sharing Ride Details:**

- Travelers share ride details with the admin and add the ride to their list.

## 3. **Companion Tracking:**

- Companions track the ride of the traveler and receive notifications about ride status.

#### 4. **Notifications:**

- Companions receive notifications when the trip is complete and when the cab hits a geofence.

#### 5. **Feedback:**

- Companions provide feedback about the ride experience, which is stored by the admin.

#### 6. **Admin Management:**

- Admins can view all rides, completed rides, and overall feedback from users.

### **Error Handling Mechanism:**

#### 1. **Try-Catch Blocks:**

- The program utilizes try-catch blocks to catch exceptions that may occur during the parsing of input data.
- Specifically, `NumberFormatException` is caught, which occurs when a numeric string cannot be parsed into the expected data type (e.g., long or int).

#### 2. **Input Validation Loop:**

- The program incorporates a loop structure to repeatedly prompt users to enter valid numeric values until correct inputs are provided.
- This loop ensures that users are not allowed to proceed with invalid inputs, thereby maintaining the integrity of the data.

### **Detailed Explanation:**

#### 1. **User Input for Traveler and Companion Details:**

- When users input their details such as name, phone number, and user ID, the program expects numeric input for the phone number.
- If the user enters a non-numeric value, a `NumberFormatException` is thrown during the parsing process.

#### 2. **Ride Details Input:**

- Similarly, when inputting ride details such as the driver's phone number and cab number, the program expects numeric inputs.
- If the user provides non-numeric values for these fields, a `NumberFormatException` is thrown.

#### 3. **Handling Invalid Inputs:**

- Upon catching a `NumberFormatException`, the program prints an error message indicating that the input was invalid.
- Users are then prompted to re-enter the data, ensuring that only valid numeric inputs are accepted.

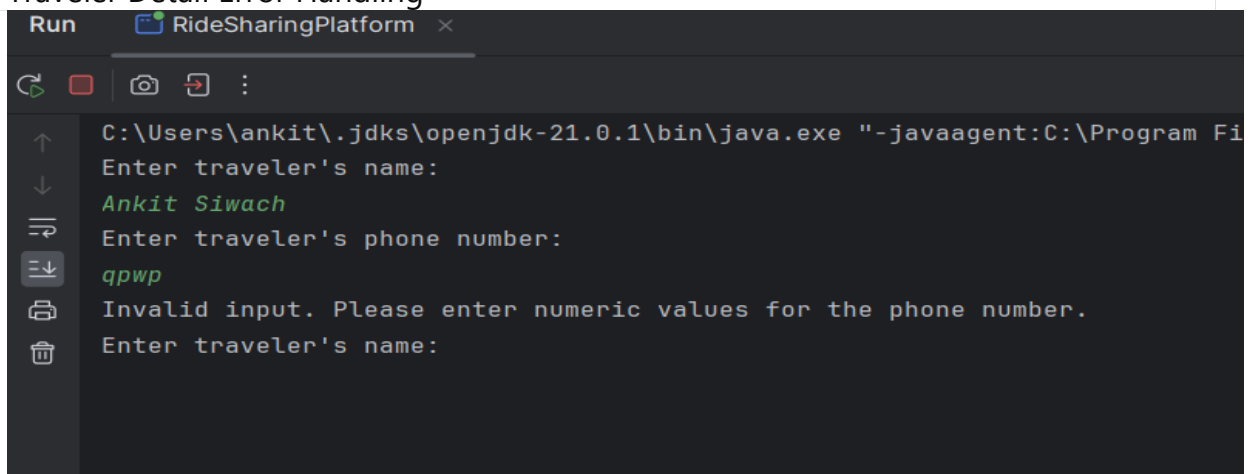
- This process continues until users provide correct inputs, at which point the loop exits and the program proceeds.

## Example Scenario:

Suppose a user enters "abc" as the phone number when prompted for traveler details. Here's how error handling works:

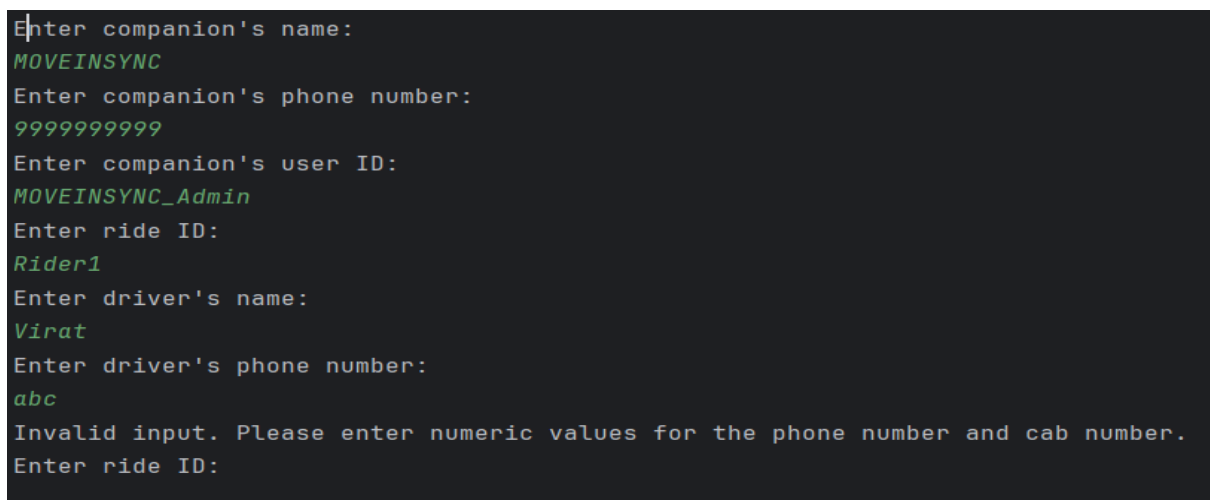
1. The program attempts to parse "abc" into a long (phone number data type).
2. Since "abc" cannot be converted into a long, a `NumberFormatException` is thrown.
3. The catch block intercepts this exception, and an error message is displayed, prompting the user to enter a numeric value for the phone number.
4. The loop continues until the user provides a valid numeric input.
5. Once a valid input is received, the program proceeds with the next steps.

### Traveler Detail Error Handling



```
Run RideSharingPlatform x
C:\Users\ankit\.jdk\openjdk-21.0.1\bin\java.exe "-javaagent:C:\Program Fi
Enter traveler's name:
Ankit Siwach
Enter traveler's phone number:
qpwp
Invalid input. Please enter numeric values for the phone number.
Enter traveler's name:
```

### Driver Detail Error Handling



```
Enter companion's name:
MOVEINSYNC
Enter companion's phone number:
9999999999
Enter companion's user ID:
MOVEINSYNC_Admin
Enter ride ID:
Rider1
Enter driver's name:
Virat
Enter driver's phone number:
abc
Invalid input. Please enter numeric values for the phone number and cab number.
Enter ride ID:
```

## Final Output :

```
Enter traveler's name:
Ankit Siwach
Enter traveler's phone number:
72000000000
Enter traveler's user ID:
Ankit1
Enter companion's name:
MOVEINSYNC
Enter companion's phone number:
9999999999
Enter companion's user ID:
MOVEINSYNC_Admin
Enter ride ID:
Rider1
Enter driver's name:
Virat
Enter driver's phone number:
5556667777
Enter cab number:
1234
Ride details shared with WhatsApp/SMS: Ride{tripId='Rider1', driverName='Virat', driverPhoneNumber=5556667777, cabNumber=1234, completed=false}
Tracking ride of traveler: Ankit1
Notification: Trip is complete.
Completed Ride: Ride{tripId='Rider1', driverName='Virat', driverPhoneNumber=5556667777, cabNumber=1234, completed=true}
Notification: Cab has hit geofence.
Enter feedback:
Very Good
Feedback received from user: MOVEINSYNC_Admin, Feedback: Very Good
Ride{tripId='Rider1', driverName='Virat', driverPhoneNumber=5556667777, cabNumber=1234, completed=true}
Completed Ride: Ride{tripId='Rider1', driverName='Virat', driverPhoneNumber=5556667777, cabNumber=1234, completed=true}
```