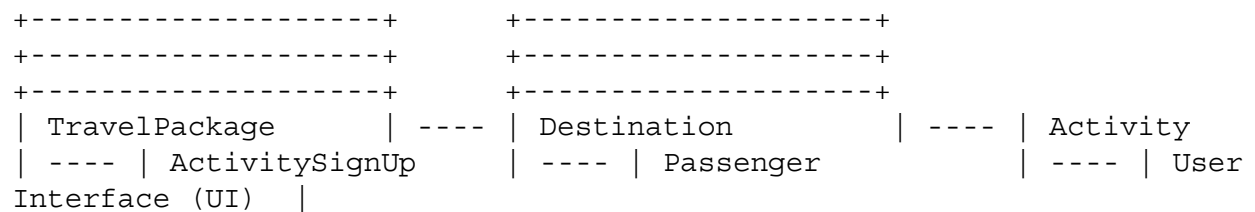


## Travel Management System Design

### High-Level Diagram:



### Low-Level Diagram (UML Class Diagram):



```

+-----+ +-----+
+-----+ +-----+
+-----+ +-----+
| - name | | - name | | - name |
| | - activityId | | - name | | -
displayItinerary |
| - capacity | ---- | - listActivities | | -
description | | - passengerId | | -
displayPassengerList |
| - itinerary | | + addActivity(act) | | - cost
| | - destinationId | | - displayDetails(p) |
| - listPassengers | | | -
capacity | | - signUp(act) | | -
displayActivities |
| - addPassenger(p) | | |
| | | - getBalance() | (if
standard/gold)
| - printItinerary() | | |
| | | - subtractBalance(amt) | (if
standard/gold)
| - printPassengerList() | | |
| | | - listActivities() |
| | |
| | |
+-----+ +-----+
+-----+ +-----+
+-----+ +-----+

```

(Relationships: One-to-Many, Many-to-Many)

### Implementation:

. Consider implementing the following:

- **TravelPackage:** Handles package details, adds/removes destinations, and manages passengers.
- **Destination:** Stores destination name, a list of available activities, and methods to add activities.
- **Activity:** Stores activity details like name, description, cost, capacity, and a reference to its destination (use composition).
- **ActivitySignUp:** Stores information about passenger signup for activities (activity, passenger, destination).
- **Passenger:** Stores passenger details like name, number, type (standard/gold/premium), balance (if applicable), and a list of activity signups.
- Ensure proper documentation for each class and method.

**Note:** This design provides a basic framework. Additional functionalities like user authentication, data persistence (database integration), and exception handling can be implemented for a complete system.

Remember, this is a complex system, and additional functionalities and considerations might be necessary depending on the specific requirements.