CONTRACT CO1: loadTrain()

- **Operation:**
  - **loadTrainConnectionsFromCSV(CSVPath: String).**
- **Cross reference:**
  - **Use case: Load Train Network.**
- **Pre-conditions:**
  - **CSV path is readable.**
- **Post-conditions:**
  - **Instance Train connection is created for each line.**
  - **Each connection is contained in TrainConnectionCatalog.**
  - **Instance of TrainGraph is created from TrainConnectionCatalog .**

CONTRACT CO2: planTrip()

- **Operation:**
  - **planTrip(departureCity: String, arrivalCity: String, trainTypes: Set<TrainType>, departureDays: Set<DayOfWeek>).**
- **Cross reference:**
  - **Use case: *Plan Trip.***
- **Pre-conditions:**
  - **Traingraph object exists and isn't empty.**
- **Post-conditions:**
  - **a** Predicate **p was created from trainTypes and departureDays.**
  - **a list pathResults is created by querying for routes with up to two intermediate paths.**
  - **for each possible route, a** PathResult **pr was created.**
  - **the operation returns pathResults.**

CONTRACT CO3: sortByPrice()

- **Operation:**
  - **sortByPrice(choiceClass) .**
- **Cross reference:**
  - **Use Case: sortByPrice.**
- **Pre-conditions:**
  - **A path result exists.**
  - **Choice of class (user input).**
- **Post-conditions:**
  - **If the option of price is chosen** then **pathResults was reordered in ascending order buy price .**
  - **Instance of comparator cmp was created.**
  - **Sort by second class tickets by default.**
  - **Displays reordered pathResults/ pathresultcatalog.**

x

CONTRACT CO4: sortByDuration()

- **Operation:**
  - **sortByDuration() .**
- **Cross reference:**
  - **Use Case: sortByDuration.**
- **Pre-conditions:**
  - **A path result exists.**
- **Post-conditions:**
  - **Instance of comparator cmp was created.**
  - **Display second class tickets by default.**
  - **PathResults is reordered in ascending totalDuration.**
  - **Displays reordered pathResults/ pathresultcatalog.**

CONTRACT CO5: viewTrips()

- **Operation:**
  - **viewTrips(name: String, id: String)**

- **Cross reference:**
  - **Use Case: View Trips.**
- **Pre-conditions:**
  - **CustomerCatalog instance exists and contains at least one Customer.**
  - **The input name and id are provided by the User.**
- **Post-conditions:**
  - The system searches the CustomerCatalog for a Customer whose name and id match the input.
  - If a matching Customer exists, viewTrip(Customer) is called.
  - or that Customer, the operation iterates through each Trip they own.
  - For each Trip, toString() is invoked to retrieve trip details.
  - Within each Trip, toString() is called on each Reservation to gather reservation details.
  - The complete formatted list of trips and reservations is displayed.
  - If no matching Customer exists, the system displays “Customer not found.”

CONTRACT CO6: bookTrip()

- **Operation:**
  - **bookTrip(customerCatalog: CustomerCatalog, chosenPath:**
- **Cross reference:**
  - **Use Case: Book Trip.**
- **Pre-conditions:**
  - **The system has successfully executed Use Case *Search for Connections* and produced a valid chosenPath.**
  - **The CustomerCatalog object exists and may contain previously registered customers.**
  - **The user is prompted to provide the number of travellers, followed by each traveller's name, ID, and age**
- **Post-conditions:**
  - **For each traveller:**
  - **The system checks if a customer with the provided name and ID already exists in the CustomerCatalog using find(id, name).**
  - **If no existing match is found, a new Customer object is created and added via add(name, id, age)**
  - **A new Trip object is created using the list of all travellers and the selected chosenPath.**
  - **The Trip constructor automatically creates a Reservation for each Customer associated with the trip**
  - **The newly created Trip is added to each Customer's list of trips (addTrip(trip)).**
  - **The system confirms successful booking by displaying all trip and reservation details for each participating customer.**
  - **The system returns to the main menu after confirmation.**