System Operations

- **addDeparture(city)**
- **addArrival(city)**
- **updateOptionalInputs(name, value)**
- **search()**
- **toggle(sort, routes)**
- **bookTrip(userTripOption, trainConnections))**
- **viewTrips(lastName, id)**

Operation Contracts

addDeparture(city)

Goal: Captures and ensures user enters a valid departure city.

Operation: addDeparture(city: String)

Preconditions:
- CSV data loaded into memory
- Set of valid cities is available in memory

Postconditions:
- If the city is valid: the system records departureCity = city in the current search context and informs the user that departure is accepted.
- If the city is invalid: the system does not change the current search context and informs the user to provide a valid departure city.

addArrival(city):

Goal: Captures and ensures a valid arrival city.

Operation: addArrival(city: String)

Preconditions:
- CSV data loaded into memory
- Set of valid cities is available in memory

Postconditions:
- If the city is valid: the system records arrivalCity = city in the current search context and informs the user that arrival is accepted.

- If the city is invalid: the system does not change the current search context and informs the user to provide a valid arrival city.

updateOptionalInputs(name, value)

Goal: Allows the user to add optional values inputs (Departure/Arrival Time, Train Type, Days of Operation, First Class Ticket Rate, Second Class ticket rate).

Operation: updateOptionalInputs(name: String, value: String)

Preconditions:
- A valid departure city has already been added
- A valid arrival city has already been added
- CSV data is loaded in memory

Postconditions:
- If the input is valid (correct format, within allowed values): the system records the (name, value) pair in the current search context and confirms acceptance to the user.
- If the input is invalid: the system does not update the current search context and prompts the user to correct the input.

search()

Goal: Allows the user to search for matching connections based on the input provided.

Operation: search()

Preconditions:
- A valid departure city has already been added
- A valid arrival city has already been added
- CSV data is loaded in memory

Postconditions:
- If direct connections exist: system returns a list of direct connections showing all public parameters (excluding Route ID) and computed tripDuration
- If there is no direct connection between two cities: indirect connections with 1-stop and 2-stops only are returned, including trip duration and transfer time for each change.

toggle(sort, routes)

Goal: Allow the user to reorder the current list of connections by a chosen parameter (e.g., trip duration). The first click on a column header sorts the list in ascending order; a second click on the same column toggles the order to descending.

Operation: toggle(sort: String, routes: TrainConnection[1…*])

Preconditions:

- A list of routes exist and is displayed to the user

Postconditions:

- First click on parameter: list is ordered in ascending order.
- Second click on the same parameter: list is reordered in descending order.
- Subsequent clicks: order continues to toggle between ascending and descending.
- The list content itself is unchanged, only the presentation order is updated; no items are added or removed.

bookTrip(userTripOption, trainConnections)

Goal: Allow a user to book a trip for one or more travellers on a selected route connection.

Operation: bookTrip(userTripOption: int, trainConnections: TrainConnection[1…*])

Preconditions: The system has already loaded all connection data (routes) in memory. The user has already performed a successful search() and selected valid routes (userTripOption).

Postconditions:

- If a valid route was selected:
    - A Trip object is created and added to Trips
    - A unique numerical ID is assigned to the created Trip object.
    - Zero or more Client objects are created and added to Clients
    - For each created Client, a Reservation is created, added to the Trip and associated with the Client
    - Each Reservation receives a Ticket and that ticket is associated with the Reservation.
    - The trip summary is printed to the console.
- If invalid option or no routes: no Trip/reservations/tickets are created; method prints an informative message and returns.

viewTrips(lastName, id)

Goal: Allow a user to view all trips (upcoming and past) associated with their identity.

Operation: viewTrips(lastName: String, id: String)

Preconditions: The system contains existing trip and reservation records. The user provides a valid lastName and id that match an existing traveller in the system.

Postconditions:

- If the traveller exists in the system: The system retrieves and displays all current/future trips (connections yet to depart). The system also retrieves all past trips (connections that have already occurred) and presents them in a separate history collection.