

## 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`.

## 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`.