Operation Contracts – Iteration 1

Cities Class
Contract CO1: get_or_create
Operation: get_or_create(String city)
Cross Reference: None.
Preconditions: There is a record for a city that is being created or looked for
Postconditions: A new city object has been instantiated (instance creation)
A new city object has been associated with cities (association formed)

Trains Class
Contract CO2: get_or_create
Operation: get_or_create(String train)
Cross Reference: None.
Preconditions: There is a record for a train that is being created or looked for
Postconditions: A new train object has been instantiated (instance creation)
A new train object has been associated with trains (association formed)

RailNetwork Class
Contract CO3: addConnection
Operation: addConnection (Connection connection)
Cross Reference: None.
Preconditions: There is a record for a connection in the data file
Postconditions: A new connection object has been instantiated (instance creation)
The connection object is associated to City objects (association formed)
The connection object is associated to a Train object (associated formed)

Contract CO4: find_direct
Operation: find_direct(String departureCity, String arrivalCity, Integer weekday):
[Connection] connections
Cross Reference: Use case UC01
Preconditions:
- Client is on the starting (main) menu.
- Client has selected option 2.
- Client inputs non-empty departureCity.
- Client inputs non-empty arrivalCity that is attainable from the departure city with at least one connection.

Postconditions: None. The system isn’t modified after a connections search.

Contract CO5: sortConnections
Operation: sortConnections([Connection] connections, String priceClass): [Connection] connections
Cross Reference: UC03
Preconditions:

- Client has previously selected option 2.
- Client has already entered a departure city and an arrival city for which at least one connection exists.
- Client is on the sorting menu.

Postconditions: None. The system isn’t modified after sorting a filtered connections search.

Contract CO6: search_ connections
Operation: search_connections(String depart_city, String arrival_city, String train_type, String min_first_class_price, Integer max_first_class_price, Integer min_second_class_price, Integer max_second_class_price, Integer min_departure_time,
Time max_departure_time, Time min_arrival_time, Time max_arrival_time,
Time min_duration, Integer max_duration, Integer weekday, Integer sort_by): [Connection]
Cross Reference: Use Case UC01
Preconditions:

- Client is on the starting (main) menu.
- Client has selected option 2.
- Client inputs non-empty departure city.
- Client inputs non-empty arrival city that is attainable from the departure city with at least one connection.

Postconditions: None. The system isn’t modified after a connections search.

Contract CO7: find_indirect
Operation: find_indirect_connections(String departureCity, String arrivalCity, Integer max_stops): [Connection] connections
Cross Reference: UC01
Preconditions:

- Client is on the starting (main) menu.
- Client has selected option 2.
- Client inputs non-empty departure city.
- Client inputs non-empty arrival city that is attainable from the departure city with at least one connection.

Postconditions: None. The system isn’t modified after a connections search.