BBB Testing Documentation

[2018-11-26 Mon]

Contents

1 Iteration 1

1.1 Specify Test Cases

1.1.1 Class Ticket

TC_Ticket_1 initializes correctly

Class Ticket
Method constructor
Precondition N/A
Input { id: “T1”, seat: 1 }
Expected Output Ticket{ id: “T1”, seat: 1, boarded: false }

TC_Ticket_2 throws error for invalid id

Class Ticket
Method constructor
Precondition N/A
Input { id: “ ”, seat: 1 }
Expected Output Error(“Invalid id”)

TC_Ticket_3 throws error for invalid seat

Class Ticket
Method constructor
Precondition N/A
Input { id: “T1”, seat: -1 }
Expected Output Error(“Invalid seat”)

TC_Ticket_4 changes value correctly

Class Ticket
Method setter boarded
Precondition Ticket{ boarded: false }
Input true
Expected Output Ticket{ boarded: true }

TC_Ticket_5 creates object correctly

Class Ticket

Method toObject

Precondition Ticket{ id: “T1”, seat: 1, boarded: false }

Input N/A

Expected Output Object{id: “T1”, seat: 1, boarded: false }

TC_Ticket_6 creates ticket correctly

Class Ticket

Method fromObject

Precondition N/A

Input Object{ id: “T1”, seat: 1, boarded: false }

Expected Output Ticket{id: “T1”, seat: 1, boarded: false }

TC_Ticket_7 throws error for invalid ticket object

Class Ticket

Method fromObject

Precondition N/A

Input Object{ id_X: “T1”, seat: 1, boarded: false }

Expected Output Error(“Invalid object”)

1.1.2 Class Route

TC_Route_1 initializes correctly

Class Route

Method constructor

Precondition N/A

Input { id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10 }

Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

TC_Route_2 throws error on invalid id

Class Route

Method constructor

Precondition N/A

Input { id: " ", source: "Madrid", destination: "Toledo", capacity: 10 }

Expected Output Error("Invalid id")

TC_Route_3 throws error on invalid source

Class Route

Method constructor

Precondition N/A

Input { id: "R1", source: " ", destination: "Toledo", capacity: 10 }

Expected Output Error("Invalid source")

TC_Route_4 throws error on invalid destination

Class Route

Method constructor

Precondition N/A

Input { id: "R1", source: "Madrid", destination: null, capacity: 10 }

Expected Output Error("Invalid source")

TC_Route_5 throws error on invalid capacity

Class Route

Method constructor

Precondition N/A

Input { id: "R1", source: "Madrid", destination: "Toledo", capacity: -1 }

Expected Output Error("Invalid capacity")

TC_Route_6 returns status "travelling" on travelling

Class Route

Method getter status

Precondition Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: "2008-09-15T15:53:00", availableSeats: [0, . . . , 9]}

Input N/A

Expected Output “travelling”

Note The date set for departed is an example. For the test the current date and time will be set

TC_Route_7 returns status “empty” on empty

Class Route

Method getter status

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Input N/A

Expected Output “empty”

TC_Route_8 returns status “available” on available

Class Route

Method getter status

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}

Input N/A

Expected Output “available”

TC_Route_9 returns status “full” on full

Class Route

Method getter status

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9, . . . , T_R1_0], departed: null, availableSeats: []}

Input N/A

Expected Output “full”

TC_Route_10 successfully purchase ticket

Class Route

Method purchaseTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Input N/A

Expected Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: false } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9], departed: null, availableSeats: [0, . . . , 8]}

TC_Route_11 purchase ticket fails on no available tickets

Class Route

Method purchaseTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

Input N/A

Expected Output { success: false, reason: “No tickets available” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

TC_Route_12 successfully board ticket

Class Route

Method boardTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: false }

Input { ticketId: “T1_R1_9” }

Expected Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: true } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

TC_Route_13 board ticket fails for invalid ticketId

Class Route

Method boardTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

Input { ticketId: “T1_R1_XXX” }

Expected Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

TC_Route_14 board ticket fails for already boarded ticketId

Class Route

Method boardTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

Input { ticketId: “T1_R1_9” }

Expected Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

TC_Route_15 successfully cancel ticket

Class Route

Method cancelTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: false }

Input { ticketId: “T1_R1_9” }

Expected Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: false } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_8, ... T1_R1_0], departed: null, availableSeats: [9]}

TC_Route_16 cancel ticket fails for invalid ticketId

Class Route

Method cancelTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

Input { ticketId: “T1_R1_XXX” }

Expected Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

TC_Route_17 cancel ticket fails for already boarded ticketId

Class Route

Method cancelTicket

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

Input { ticketId: “T1_R1_9” }

Expected Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

TC_Route_18 depart successfully sets departure time

Class Route

Method depart

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Input N/A

Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 9]}

Note The date set for departed is an example. For the test the current date and time will be set

TC_Route_19 hasArrived successfully resets the Route

Class Route

Method hasArrived

Precondition Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: "2008-09-15T15:53:00", availableSeats: []}

Input N/A

Expected Output true, Route{ id: "R1", source: "Toledo", destination: "Madrid", capacity: 10, tickets: [], departed: null, availableSeats: [0, ..., 9]}

Note The date set for departed is an example. For the test the current date and time will be set

TC_Route_20 hasArrived does not reset the Route if no departed yet

Class Route

Method hasArrived

Precondition Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

Input N/A

Expected Output false, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

TC_Route_21 hasArrived does not reset the Route if still travelling

Class Route

Method hasArrived

Precondition Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: "2008-09-15T15:53:00", availableSeats: []}

Input N/A

Expected Output false, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: “2008-09-15T15:53:00”, availableSeats: []}

Note The date set for departed is an example. For the test the current date and time will be set so that the 10 seconds have not passed yet

TC_Route_22 fromObject successfully creates new Route with set departure

Class Route

Method fromObject

Precondition N/A

Input { id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

Note The date set for departed is an example

TC_Route_23 fromObject successfully creates new Route without set departure and tickets

Class Route

Method fromObject

Precondition N/A

Input { id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, ..., 9]}

Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, ..., 9]}

TC_Route_24 toObject successfully creates new Object with set departure

Class Route

Method toObject

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

Input N/A

Expected Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

TC_Route_25 toObject successfully creates new Object without departure

Class Route

Method toObject

Precondition Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: null, availableSeats: [0, 1, 2]}

Input N/A

Expected Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_3], departed: null, availableSeats: [0, 1, 2]}

1.1.3 IBBBCommand based classes

TC_RegisterRouteCommand_1 returns correct id

Class RegisterRouteCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output ‘registerroute’

TC_RegisterRouteCommand_2 fails for invalid number of arguments

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input []

Expected Output BBB{ _routes: [] } Console: 'Invalid number of arguments given'

TC_RegisterRouteCommand_3 fails for invalid route

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input [“ ”, “Madrid”, “Toledo”, 10]

Expected Output BBB{ _routes: [] } Console: ‘Invalid value for route given’

TC_RegisterRouteCommand_4 fails for invalid source

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input [“R1”, null, “Toledo”, 10]

Expected Output BBB{ _routes: [] } Console: ‘Invalid value for source given’

TC_RegisterRouteCommand_5 fails for invalid destination

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input [“R1”, “Madrid”, undefined, 10]

Expected Output BBB{ _routes: [] } Console: ‘Invalid value for destination given’

TC_RegisterRouteCommand_6 fails for invalid capacity

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input [“R1”, “Madrid”, “Toledo”, “asdf”]

Expected Output BBB{ _routes: [] } Console: ‘Invalid value for capacity’

TC_RegisterRouteCommand_7 succeeds for valid input

Class RegisterRouteCommand

Method execute

Precondition BBB{ _routes: [] }

Input [“R1”, “Madrid”, “Toledo”, 10”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]} Console: “Created route R1 from Madrid to Toledo with 10 seats”

TC_DeleteRouteCommand_1 returns correct id

Class DeleteRouteCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output ‘deleteroute’

TC_DeleteRouteCommand_2 fails for invalid number of arguments

Class DeleteRouteCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input []

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]} Console: ‘Invalid number of arguments given’

TC_DeleteRouteCommand_3 fails for invalid route

Class DeleteRouteCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input [“ ”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]} Console: ‘Invalid value for route given’

TC_DeleteRouteCommand_4 fails for route with purchased tickets

Class DeleteRouteCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input [“R1”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]} Console: “Cannot delete route R1 because there are 1 tickets booked”

TC_DeleteRouteCommand_5 succeeds for valid input

Class DeleteRouteCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}

Input [“R1”]

Expected Output BBB{ _routes: [] } Console: “Successfully deleted route R1”

TC_DepartCommand_1 returns correct id

Class DepartCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output ‘depart’

TC_DepartCommand_2 fails for invalid number of arguments

Class DepartCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input []

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]} Console: ‘Invalid number of arguments given’

TC_DepartCommand_3 fails for invalid route

Class DepartCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input [“R_X”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]} Console: ‘Invalid value for route given’

TC_DepartCommand_4 succeeds for valid route

Class DepartCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Input [“R1”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 8]}]} Console: ‘R1 departed’

TC_StatusCommand_1 returns correct id

Class StatusCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output 'status'

TC_StatusCommand_2 fails for invalid number of arguments

Class StatusCommand

Method execute

Precondition BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}

Input ["A", "B"]

Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]} Console: 'Invalid number of arguments given'

TC_StatusCommand_3 does not print anything when specifying not existing route

Class StatusCommand

Method execute

Precondition BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}

Input ["R3"]

Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]} Console: 'Route R3 does not exist'

TC_StatusCommand_4 prints status of one specified route successfully

Class StatusCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}

Input [“R2”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]} Console: ‘R2: empty’

TC_StatusCommand_5 prints status without specified route successfully

Class StatusCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}

Input []

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]} Console: “R1: available R2: empty’”

TC_BuyCommand_1 returns correct id

Class BuyCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output 'buy'

TC_BuyCommand_2 fails for not existing route

Class BuyCommand

Method execute

Precondition BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}

Input ["R3"]

Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]} Console: 'Route R3 does not exist'

TC_BuyCommand_3 fails for sold out route

Class BuyCommand

Method execute

Precondition BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9, ... T_R1_0], departed: null, availableSeats: []}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}

Input ["R1"]

Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9, ... T_R1_0]], departed: null, availableSeats: []}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]} Console: 'Sorry! You were too late! Tickets are sold out!'

TC_BuyCommand_4 succeeds for valid route

Class BuyCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}

Input [“R1”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9, T_R1_8], departed: null, availableSeats: [0, . . . , 7]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]} Console: ‘Successfully purchased ticket T_R1_8 on route R1 from Madrid to Toledo’

TC_CheckinCommand_1 returns correct id

Class CheckinCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output ‘checkin’

TC_CheckinCommand_2 fails for invalid number of arguments

Class CheckinCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input []

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Invalid number of arguments given”

TC_CheckinCommand_3 fails for invalid value for ticket

Class CheckinCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“ “]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Invalid value for ticket given”

TC_CheckinCommand_4 fails for not existing ticket

Class CheckinCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“T_R1_X”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Ticket with id T_R1_X does not exist”

TC_CheckinCommand_5 fails already boarded ticket

Class CheckinCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true }

Input [“T_R1_9”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true } Console: “Unable to checkin ticket T_R1_9: Ticket is already boarded”

TC_CheckinCommand_6 succeeds for valid ticket

Class CheckinCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“T_R1_9”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true } Console: “Successfully checked in ticket T_R1_9 on route R1 from Madrid to Toledo and assigned seat 9”

TC_CancelCommand_1 returns correct id

Class CancelCommand

Method commandId get

Precondition N/A

Input N/A

Expected Output ‘cancel’

TC_CancelCommand_2 fails for invalid number of arguments

Class CancelCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input []

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Invalid number of arguments given”

TC_CancelCommand_3 fails for invalid value for ticket

Class CancelCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“ “]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Invalid value for ticket given”

TC_CancelCommand_4 fails for not existing ticket

Class CancelCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“T_R1_X”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false } Console: “Ticket with id T_R1_X does not exist”

TC_CancelCommand_5 fails already boarded ticket

Class CancelCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true }

Input [“T_R1_9”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true } Console: “Unable to cancel ticket T_R1_9: Ticket is already boarded”

TC_CancelCommand_6 succeeds for valid ticket

Class CancelCommand

Method execute

Precondition BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }

Input [“T_R1_9”]

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]} Console: “Cancelled ticket T_R1_9 on route R1 from Madrid to Toledo”

1.1.4 Class BBB

TC_BBB_1 successfully writes file

Class BBB

Method saveRoutes

Precondition routes: [{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, { id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]

Input N/A

Expected Output file: [{ “id”: “R1”, “source”: “Madrid”, “destination”: “Toledo”, “capacity”: 10, “tickets”: [{id: “T_R1_9”, “seat”: 9, “boarded”: false)}], “departed”: null, “availableSeats”: [0, . . . , 8]}, { “id”: “R2”, “source”: “Barcelona”, “destination”: “Valencia”, “capacity”: 10, “tickets”: [], “departed”: null, “availableSeats”: [0, . . . , 9]}]

TC_BBB_2 successfully reads file with routes

Class BBB

Method loadRoutes

Precondition routes: undefined file: [{ “id”: “R1”, “source”: “Madrid”, “destination”: “Toledo”, “capacity”: 10, “tickets”: [{id: “T_R1_9”, “seat”: 9, “boarded”: false)}], “departed”: null, “availableSeats”: [0, . . . , 8]}, { “id”: “R2”, “source”: “Barcelona”, “destination”: “Valencia”, “capacity”: 10, “tickets”: [], “departed”: null, “availableSeats”: [0, . . . , 9]}]

Input N/A

Expected Output routes: [{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, { id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]

TC_BBB_3 successfully reads without routes

Class BBB

Method loadRoutes

Precondition routes: undefined file: []

Input N/A

Expected Output routes: []

TC_BBB_4 does not read not existing file

Class BBB

Method loadRoutes

Precondition: routes undefined, filePath: “asdf”

Input N/A

Expected Output routes: []

TC_BBB_5 fails for no arguments given

Class BBB

Method parseCommand

Precondition N/A

Input: args []

Expected Output Console: “No argument was given”

TC_BBB_6 fails for not existing command

Class BBB

Method parseCommand

Precondition N/A

Input: args [“asdf”]

Expected Output Console: “Command asdf does not exist”

TC_BBB_7 succeeds for existing command

Class BBB

Method parseCommand

Precondition N/A

Input: args [“status”]

Expected Output _commands[“status”].execute was called

1.2 Run Test Cases

1.2.1 Class Ticket

- **TC_Ticket_1**

  Expected Output Ticket{ id: “T1”, seat: 1, boarded: false }

  Observed Output Ticket{ id: “T1”, seat: 1, boarded: false }

  Failure None

- **TC_Ticket_2**

  Expected Output Error(“Invalid id”)

  Observed Output Error(“Invalid id”)

  Failure None

- **TC_Ticket_3**

  Expected Output Error("Invalid seat")

  Observed Output Error("Invalid seat")

  Failure None

- **TC_Ticket_4**

  Expected Output Ticket{ boarded: true }

  Observed Output Ticket{ boarded: true }

  Failure None

- **TC_Ticket_5**

  Expected Output Object{id: "T1", seat: 1, boarded: false }

  Observed Output Object{id: "T1", seat: 1, boarded: false }

  Failure None

- **TC_Ticket_6**

  Expected Output Ticket{id: "T1", seat: 1, boarded: false }

  Observed Output Ticket{id: "T1", seat: 1, boarded: false }

  Failure None

- **TC_Ticket_7**

  Expected Output Error("Invalid object")

  Observed Output Error("Invalid object")

  Failure None

1.2.2 Class Route

- **TC_Route_1**

  Expected Output Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}

  Observed Output Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}

Failure None

- **TC_Route_2**

  Expected Output Error("Invalid id")
  Observed Output Error("Invalid id")
  Failure None

- **TC_Route_3**

  Expected Output Error("Invalid source")
  Observed Output Error("Invalid source")
  Failure None

- **TC_Route_4**

  Expected Output Error("Invalid source")
  Observed Output Error("Invalid source")
  Failure None

- **TC_Route_5**

  Expected Output Error("Invalid capacity")
  Observed Output Error("Invalid capacity")
  Failure None

- **TC_Route_6**

  Expected Output "travelling"
  Observed Output 0
  Failure Yes

- **TC_Route_7**

  Expected Output "empty"
  Observed Output 1
  Failure Yes

- **TC_Route_8**

  Expected Output "available"

Observed Output 3

Failure Yes

- **TC_Route_9**

  Expected Output "full"

  Observed Output 2

  Failure Yes

- **TC_Route_10**

  Expected Output { success: true, ticket: Ticket{ id: "T1_R1_9", seat: 9, boarded: false } }, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9], departed: null, availableSeats: [0, . . . , 8]}

  Observed Output { success: true, ticket: Ticket{ id: "T1_R1_9", seat: 9, boarded: false } }, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9], departed: null, availableSeats: [0, . . . , 8]}

  Failure None

- **TC_Route_11**

  Expected Output { success: false, reason: "No tickets available" }, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Observed Output { success: false, reason: "No tickets available" }, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Failure None

- **TC_Route_12**

  Expected Output { success: true, ticket: Ticket{ id: "T1_R1_9", seat: 9, boarded: true } }, Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

Observed Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: true } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

Failure None

- **TC_Route_13**

  Expected Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

  Observed Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}

  Failure None

- **TC_Route_14**

  Expected Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

  Observed Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, ... T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

  Failure None

- **TC_Route_15**

  Expected Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: false } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_8, ... T1_R1_0], departed: null, availableSeats: [9]}

  Observed Output { success: true, ticket: Ticket{ id: “T1_R1_9”, seat: 9, boarded: false } }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_8, ... T1_R1_0], departed: null, availableSeats: []}

Failure Yes

- **TC_Route_16**

  Expected Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Observed Output { success: false, reason: “Ticket does not exist” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Failure None

- **TC_Route_17**

  Expected Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

  Observed Output { success: false, reason: “Ticket is already boarded” }, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}, T1_R1_9{ id: “T1_R1_9”, seat: 9, boarded: true }

  Failure None

- **TC_Route_18**

  Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 9]}

  Observed Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 9]}

  Failure None

- **TC_Route_19**

Expected Output true, Route{ id: “R1”, source: “Toledo”, destination: “Madrid”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Observed Output true, Route{ id: “R1”, source: “Toledo”, destination: “Madrid”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Failure None

- **TC_Route_20**

  Expected Output false, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Observed Output false, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: null, availableSeats: []}

  Failure None

- **TC_Route_21**

  Expected Output false, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: “2008-09-15T15:53:00”, availableSeats: []}

  Observed Output false, Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_0], departed: “2008-09-15T15:53:00”, availableSeats: []}

  Failure None

- **TC_Route_22**

  Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

  Observed Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

  Failure None

- **TC_Route_23**

Expected Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Observed Output Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}

Failure None

- **TC_Route_24**

  Expected Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

  Observed Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: “2008-09-15T15:53:00”, availableSeats: [0, 1, 2]}

  Failure None

- **TC_Route_25**

  Expected Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: null, availableSeats: [0, 1, 2]}

  Observed Output Object{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T1_R1_9, . . . T1_R1_3], departed: null, availableSeats: [0, 1, 2]}

  Failure None

1.2.3 IBBBCommand based classes

- **TC_RegisterRouteCommand_1**

  Expected Output ‘registerroute’

  Observed Output ‘registerroute’

  Failure None

- **TC_RegisterRouteCommand_2**

  Expected Output BBB{ _routes: [] }
  Console: ’Invalid number of arguments given’

Observed Output BBB{ _routes: [] }
Console: 'Invalid number of arguments given'

Failure None

- **TC_RegisterRouteCommand_3**

  Input [" ", "Madrid", "Toledo", 10]

  Expected Output BBB{ _routes: [] }
  Console: 'Invalid value for route given'

  Observed Output BBB{ _routes: [] }
  Console: 'Invalid value for route given'

  Failure None

- **TC_RegisterRouteCommand_4**

  Expected Output Console: 'Invalid value for source given'

  Observed Output TypeError('Cannot read property 'trim' of null')

  Failure Yes

- **TC_RegisterRouteCommand_5**

  Expected Output Console: 'Invalid value for destination given'

  Observed Output TypeError('Cannot read property 'trim' of undefined')

  Failure Yes

- **TC_RegisterRouteCommand_6**

  Expected Output Console: 'Invalid value for capacity'

  Observed Output RangeError(Invalid array length)

  Failure Yes

- **TC_RegisterRouteCommand_7**

  Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}
  Console: "Created route R1 from Madrid to Toledo with 10 seats"

Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: “Created route R1 from Madrid to Toledo with 10 seats”

Failure None

- **TC_DeleteRouteCommand_1**

  Expected Output ‘deleteroute’

  Observed Output ‘deleteroute’

  Failure None

- **TC_DeleteRouteCommand_2**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid number of arguments given’

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid number of arguments given’

  Failure None

- **TC_DeleteRouteCommand_3**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid value for route given’

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid value for route given’

  Failure None

- **TC_DeleteRouteCommand_4**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}

Console: “Cannot delete route R1 because there are 1 tickets booked”

Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
Console: “Cannot delete route R1 because there are 1 tickets booked”

Failure None

- **TC_DeleteRouteCommand_5**

  Expected Output BBB{ _routes: [] }
  Console: “Successfully deleted route R1”

  Observed Output BBB{ _routes: [] }
  Console: “Successfully deleted route R1”

  Failure None

- **TC_DepartCommand_1**

  Expected Output ‘depart’

  Observed Output ‘depart’

  Failure None

- **TC_DepartCommand_2**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid number of arguments given’

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}
  Console: ‘Invalid number of arguments given’

  Failure None

- **TC_DepartCommand_3**

  Expected Output Console: ‘Invalid value for route given’

  Observed Output Console: ‘Route R_X does not exist’

  Failure Yes

- **TC_DepartCommand_4**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 8]}]}
  Console: ‘R1 departed’

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: “2008-09-15T15:53:00”, availableSeats: [0, . . . , 8]}]}
  Console: ‘R1 departed’

  Failure None

- **TC_StatusCommand_1**

  Expected Output ‘status’

  Observed Output ‘status’

  Failure None

- **TC_StatusCommand_2**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
  Console: ‘Invalid number of arguments given’

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
  Console: ‘Invalid number of arguments given’

  Failure None

- **TC_StatusCommand_3**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
  Console: ‘Route R3 does not exist’

Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: ‘Route R3 does not exist’

Failure None

- **TC_StatusCommand_4**

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: ‘R2: empty’

Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: ‘R2: empty’

Failure None

- **TC_StatusCommand_5**

Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: “R1: available R2: empty’”

Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}, Route{ id: “R2”, source: “Barcelona”, destination: “Valencia”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
Console: “R1: available R2: empty’”

Failure None

- **TC_BuyCommand_1**

  Expected Output 'buy'

  Observed Output 'buy'

  Failure None

- **TC_BuyCommand_2**

  Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}
  Console: 'Route R3 does not exist'

  Observed Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}
  Console: 'Route R3 does not exist'

  Failure None

- **TC_BuyCommand_3**

  Expected Output Console: 'Sorry! You were too late! Tickets are sold out!'

  Observed Output TypeError(Cannot read property 'id' of undefined)

  Failure Yes

- **TC_BuyCommand_4**

  Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9, T_R1_8], departed: null, availableSeats: [0, ... , 7]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}
  Console: 'Successfully purchased ticket T_R1_8 on route R1 from Madrid to Toledo'

  Observed Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9, T_R1_8],

departed: null, availableSeats: [0, ... , 7]}, Route{ id: "R2", source: "Barcelona", destination: "Valencia", capacity: 10, tickets: [], departed: null, availableSeats: [0, ... , 9]}]}
Console: 'Successfully purchased ticket T_R1_8 on route R1 from Madrid to Toledo'

Failure None

- **TC_CheckinCommand_1**

  Expected Output 'checkin'

  Observed Output 'checkin'

  Failure None

- **TC_CheckinCommand_2**

  Expected Output Console: "Invalid number of arguments given"

  Observed Output Console: "Invalid number of arguments given" "Ticket with id null does not exist"

  Failure Yes

- **TC_CheckinCommand_3**

  Expected Output Console: "Invalid value for ticket given"

  Observed Output Console: "Invalid value for ticket given" "Ticket with id null does not exist"

  Failure Yes

- **TC_CheckinCommand_4**

  Expected Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}]}, Ticket{ id: "T_R1_9", seat: 9, boarded: false }
  Console: "Ticket with id T_R1_X does not exist"

  Observed Output BBB{ _routes: [Route{ id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, ... , 8]}]}, Ticket{ id: "T_R1_9", seat: 9, boarded: false }
  Console: "Ticket with id T_R1_X does not exist"

  Failure None

- **TC_CheckinCommand_5**

  Expected Output Console: “Unable to checkin ticket T_R1_9: Ticket is already boarded”

  Observed Output TypeError(Cannot read property ‘seat’ of undefined)

  Failure Yes

- **TC_CheckinCommand_6**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true }
  Console: “Successfully checked in ticket T_R1_9 on route R1 from Madrid to Toledo and assigned seat 9”

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: true }
  Console: “Successfully checked in ticket T_R1_9 on route R1 from Madrid to Toledo and assigned seat 9”

  Failure None

- **TC_CancelCommand_1**

  Expected Output ‘cancel’

  Observed Output ‘cancel’

  Failure None

- **TC_CancelCommand_2**

  Expected Output Console: “Invalid number of arguments given”

  Observed Output Console: “Invalid number of arguments given”
  Console: “Ticket with id null does not exist”

  Failure Yes

- **TC_CancelCommand_3**

  Expected Output Console: “Invalid value for ticket given”

Observed Output Console: “Invalid value for ticket given”
Console: “Ticket with id null does not exist”

Failure Yes

- **TC_CancelCommand_4**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }
  Console: “Ticket with id T_R1_X does not exist”

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [T_R1_9], departed: null, availableSeats: [0, . . . , 8]}]}, Ticket{ id: “T_R1_9”, seat: 9, boarded: false }
  Console: “Ticket with id T_R1_X does not exist”

  Failure None

- **TC_CancelCommand_5**

  Expected Output Console: “Unable to cancel ticket T_R1_9: Ticket is already boarded”

  Observed Output Console: “Unable to cancel ticket T_R1_9: Ticket is already boarded”
  Console: “Cancelled ticket T_R1_9 on route R1 from Madrid to Toledo”

  Failure Yes

- **TC_CancelCommand_6**

  Expected Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
  Console: “Cancelled ticket T_R1_9 on route R1 from Madrid to Toledo”

  Observed Output BBB{ _routes: [Route{ id: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, . . . , 9]}]}
  Console: “Cancelled ticket T_R1_9 on route R1 from Madrid to Toledo”

  Failure None

1.2.4 Class BBB

1.3 Check Coverage

```
-----------------|----------|----------|----------|----------|-------------------
File             |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s
-----------------|----------|----------|----------|----------|-------------------
All files        |    36.06 |    39.22 |    35.37 |    36.24 |
 BBB.js          |        0 |        0 |        0 |        0 | ... 58,61,66,68,69
 IBBBCommand.js  |        0 |        0 |        0 |        0 | ... 78,280,285,287
 Route.js        |    98.18 |    95.92 |      100 |    98.11 |         191,200
 RouteStatus.js  |      100 |      100 |      100 |      100 |
 Ticket.js       |      100 |      100 |      100 |      100 |
-----------------|----------|----------|----------|----------|-------------------
Test Suites: 1 failed, 1 passed, 2 total
Tests:       5 failed, 27 passed, 32 total
Snapshots:   0 total
Time:        3.741s, estimated 4s
Ran all test suites.
```

1.3.1 Identify Missing Tests

```
Route.fromObject = function (object) {
    if (!object.hasOwnProperty('id') ||
        !object.hasOwnProperty('source') ||
        !object.hasOwnProperty('destination') ||
        !object.hasOwnProperty('capacity') ||
        !object.hasOwnProperty('departed') ||
        !object.hasOwnProperty('availableSeats') ||
        !object.hasOwnProperty('tickets')) {
        throw new Error('Invalid object');
    }
    var route = new Route(object.id, object.source, object.destination, object.capacity
    if (object.departed === null) {
        route._departed = null;
    }
    else {
        route._departed = moment(object.departed);
        if (!route._departed.isValid()) {
            throw new Error('Invalid departed time');
        }
    }
    route._availableSeats = object.availableSeats;
    for (var i in object.tickets) {
        var ticket = Ticket_1.Ticket.fromObject(object.tickets[i]);
        route._tickets.push(ticket);
    }
    return route;
};
return Route;
```

TC_Route_26 fromObject fails on invalid object

Class Route

Method fromObject

Precondition N/A

Input { id_X: “R1”, source: “Madrid”, destination: “Toledo”, capacity: 10, tickets: [], departed: null, availableSeats: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]}

Expected Output Error('Invalid object')

Note The date set for departed is an example

TC_Route_27 fromObject fails on invalid departure time

Class Route

Method fromObject

Precondition N/A

Input { id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: "4711", availableSeats: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]}

Expected Output Error('Invalid departed time')

Detected new failure in TC_Route_27 Pasted Graphic 6.tiff ¬

TC_Route_27: fromObject fails on invalid departure time Input: { id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: "4711", availableSeats: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]} Expected Output: Error('Invalid departed time') Observed Output: { id: "R1", source: "Madrid", destination: "Toledo", capacity: 10, tickets: [], departed: "4711-01-01T00:00:00.000Z", availableSeats: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]}

Pasted Graphic 8.tiff ¬

Fault: moment is not parsed enforcing ISO_8601 date format

Fix: parse enforcing ISO_8601 date format Pasted Graphic 7.tiff ¬

Passing all tests and 100% coverage Pasted Graphic 9.tiff ¬

1.4 Trace failures to faults

1.4.1 TC_Route_6, TC_Route_7, TC_Route_8, TC_Route_9

Failure Output is a number instead of a readable message.

Fault Method `status()` returns `enum` value instead of `string`:

```
export enum RouteStatus {
    travelling,
    empty,
    full,
    available
}
```

Fix Assign values to `enum`:

```
export enum RouteStatus {
    travelling = 'travelling',
    empty = 'empty',
    full = 'full',
    available = 'available'
}
```

1.4.2 TC_Route_15

Failure Number of available seats is not increased on a cancellation.

Fault `cancelTicket` does remove the Ticket from the list of Tickets but does not add the seat of the ticket back to the list of available seats:

```
cancelTicket = (ticketId: string) => {
    const ticketIndex = this._tickets.map((t) => t.id).indexOf(ticketId)

    if (ticketIndex === -1) {
        return {
            success: false,
            reason: 'Ticket does not exist'
        }
    }

    const ticket = this._tickets[ticketIndex]

    if (ticket.boarded === true) {
        return {
            success: false,
            reason: 'Ticket is already boarded'
        }
    }

    this._tickets = this._tickets.filter((t) => t.id !== ticketId)

    return {
        success: true,
        ticket: ticket
    }
}
```

Fix Added the seat of the ticket to the list of available seats:

```
cancelTicket = (ticketId: string) => {
    const ticketIndex = this._tickets.map((t) => t.id).indexOf(ticketId)

    if (ticketIndex === -1) {
        return {
            success: false,
            reason: 'Ticket does not exist'
        }
    }

    const ticket = this._tickets[ticketIndex]

    if (ticket.boarded === true) {
        return {
            success: false,
            reason: 'Ticket is already boarded'
        }
    }

    this._tickets = this._tickets.filter((t) => t.id !== ticketId)

    const seat = ticket.seat
    this._availableSeats.push(seat)

    return {
        success: true,
        ticket: ticket
    }
}
```

1.4.3 TC_RegisterRouteCommand_4

Failure `TypeError` instead of error message

Fault `args[1]` is `null` and `trim()` cannot be called on `null`

```
execute = (args: Array<any>) => {
    if (args.length !== 4) {
        console.log('Invalid number of arguments given')
        return
    }

    const routeId = args[0].trim()
    if (!routeId || routeId.length === 0) {
        console.log('Invalid value for route given')
        return
    }

    const source = args[1].trim()
    if (!source || source.length === 0) {
        console.log('Invalid value for source given')
        return
    }

    const destination = args[2].trim()
    if (!destination || destination.length === 0) {
        console.log('Invalid value for destination given')
        return
    }

    let capacity = Number(args[3])
    if (capacity === NaN || capacity < 1) {
        console.log('Invalid value for capacity given')
        return
    }

    const route = new Route(routeId, source, destination, capacity)
    this._bbb.routes.push(route)

    console.log(`Created route ${routeId} from ${source} to ${destination} with ${capacity} seats`)
}
```

Fix Check that `args[1]` is not `null`

```
execute = (args: Array<any>) => {
    if (args.length !== 4) {
        console.log('Invalid number of arguments given')
        return
    }

    if (!args[0] || args[0].trim().length === 0) {
        console.log('Invalid value for route given')
        return
    }
    const routeId = args[0].trim()

    if (!args[1] || args[1].trim().length === 0) {
        console.log('Invalid value for source given')
        return
    }
    const source = args[1].trim()

    if (!args[2] || args[2].trim().length === 0) {
        console.log('Invalid value for destination given')
        return
    }
    const destination = args[2].trim()

    let capacity = Number(args[3])
    if (isNaN(capacity) || capacity < 1) {
        console.log('Invalid value for capacity given')
        return
    }

    const route = new Route(routeId, source, destination, capacity)
    this._bbb.routes.push(route)

    console.log(`Created route ${routeId} from ${source} to ${destination} with ${capacity} seats`)
}
```

1.4.4 TC_RegisterRouteCommand_5

Same as with TC_RegisterRouteCommand_4 but with `args[2]`

1.4.5 TC_RegisterRouteCommand_6

Failure `RangeError` instead of Error message

Fault Using `=== NaN` always yields false

Fix Use of `isNaN`

1.4.6 TC_DepartCommand_3

Test case was poorly specified (the observed output is correct and the expected one is not):

- Title: fails for not existing Route
- Expected Output: Console("Route R_X does not exist")

Fix Test case adapted

1.4.7 TC_BuyCommand_3

Failure `TypeError` instead of just error message

Fault Missing `return` statement after error message

```
execute = (args: Array<any>) => {

    const route = this.getRouteFromArgs(args)
    if (route === null) {
        return
    }

    const result = route.purchaseTicket()

    if (!result.success) {
        console.log('Sorry! You were too late! Tickets are sold out!')
    }

    const ticket = result.ticket

    console.log(`Successfully purchased ticket ${ticket.id} on route ${route.id} from ${route.source} to ${route.destination}`)
    return
}
```

Fix `return` statement added

```
execute = (args: Array<any>) => {

    const route = this.getRouteFromArgs(args)
    if (route === null) {
        return
    }

    const result = route.purchaseTicket()

    if (!result.success) {
        console.log('Sorry! You were too late! Tickets are sold out!')
        return
    }

    const ticket = result.ticket

    console.log(`Successfully purchased ticket ${ticket.id} on route ${route.id} from ${route.source} to ${route.destination}`)
    return
}
```

1.4.8 TC_CheckinCommand_2

Failure Additional wrong error message printed

Fault Missing `null` check for `ticketId`

```
execute = (args: Array<any>) => {

    const ticketId = this.getTicketIdFromArgs(args)
    const route = this.getRouteFromTicketId(ticketId)
    if (route == null) {
        return
    }

    const result = route.boardTicket(ticketId)

    if (!result.success) {
        console.log(`Unable to checkin ticket ${ticketId}: ${result.reason}`)
    }

    const ticket = result.ticket

    console.log(`Successfully checked in ticket ${ticketId} on route ${route.id} from ${route.source} to ${route.destination} and
    assigned seat ${ticket.seat}`)
    return
}
```

Fix Added `null` check for `ticketId`

1.4.9 TC_CheckinCommand_3

Same as TC_CheckinCommand_2

1.4.10 TC_CheckinCommand_5

Failure `TypeError` instead of just error message

Fault Missing return after error message

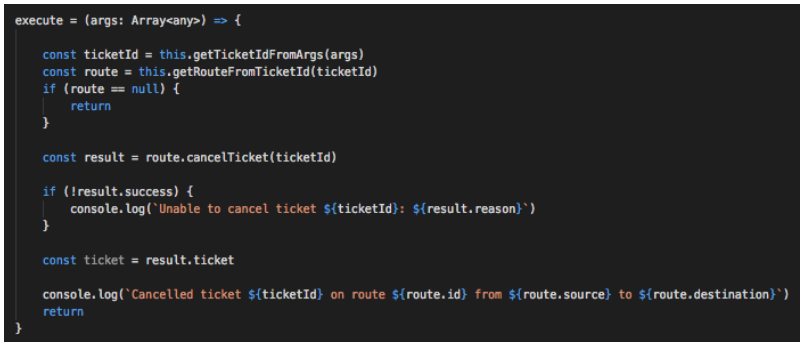
```
execute = (args: Array<any>) => {

    const ticketId = this.getTicketIdFromArgs(args)
    const route = this.getRouteFromTicketId(ticketId)
    if (route == null) {
        return
    }

    const result = route.boardTicket(ticketId)

    if (!result.success) {
        console.log(`Unable to checkin ticket ${ticketId}: ${result.reason}`)
    }

    const ticket = result.ticket

    console.log(`Successfully checked in ticket ${ticketId} on route ${route.id} from ${route.source} to ${route.destination} and
    assigned seat ${ticket.seat}`)
    return
}
```

Fix Added `return` statement

1.4.11 TC_CancelCommand_2

Failure Additional wrong error message printed

Fault Missing `null` check for `ticketId`



```
execute = (args: Array<any>) => {

    const ticketId = this.getTicketIdFromArgs(args)
    const route = this.getRouteFromTicketId(ticketId)
    if (route == null) {
        return
    }

    const result = route.cancelTicket(ticketId)

    if (!result.success) {
        console.log(`Unable to cancel ticket ${ticketId}: ${result.reason}`)
    }

    const ticket = result.ticket

    console.log(`Cancelled ticket ${ticketId} on route ${route.id} from ${route.source} to ${route.destination}`)
    return
}
```

Fix Added `null` check for `ticketId`

1.4.12 TC_CancelCommand_3

Same as TC_CancelCommand_2

1.4.13 TC_CancelCommand_5

Failure Wrong message printed, ticket data falsely manipulated

Fault Missing return after error message

```
execute = (args: Array<any>) => {

    const ticketId = this.getTicketIdFromArgs(args)
    const route = this.getRouteFromTicketId(ticketId)
    if (route == null) {
        return
    }

    const result = route.cancelTicket(ticketId)

    if (!result.success) {
        console.log(`Unable to cancel ticket ${ticketId}: ${result.reason}`)
    }

    const ticket = result.ticket

    console.log(`Cancelled ticket ${ticketId} on route ${route.id} from ${route.source} to ${route.destination}`)
    return
}
```

Fix Added `return` statement