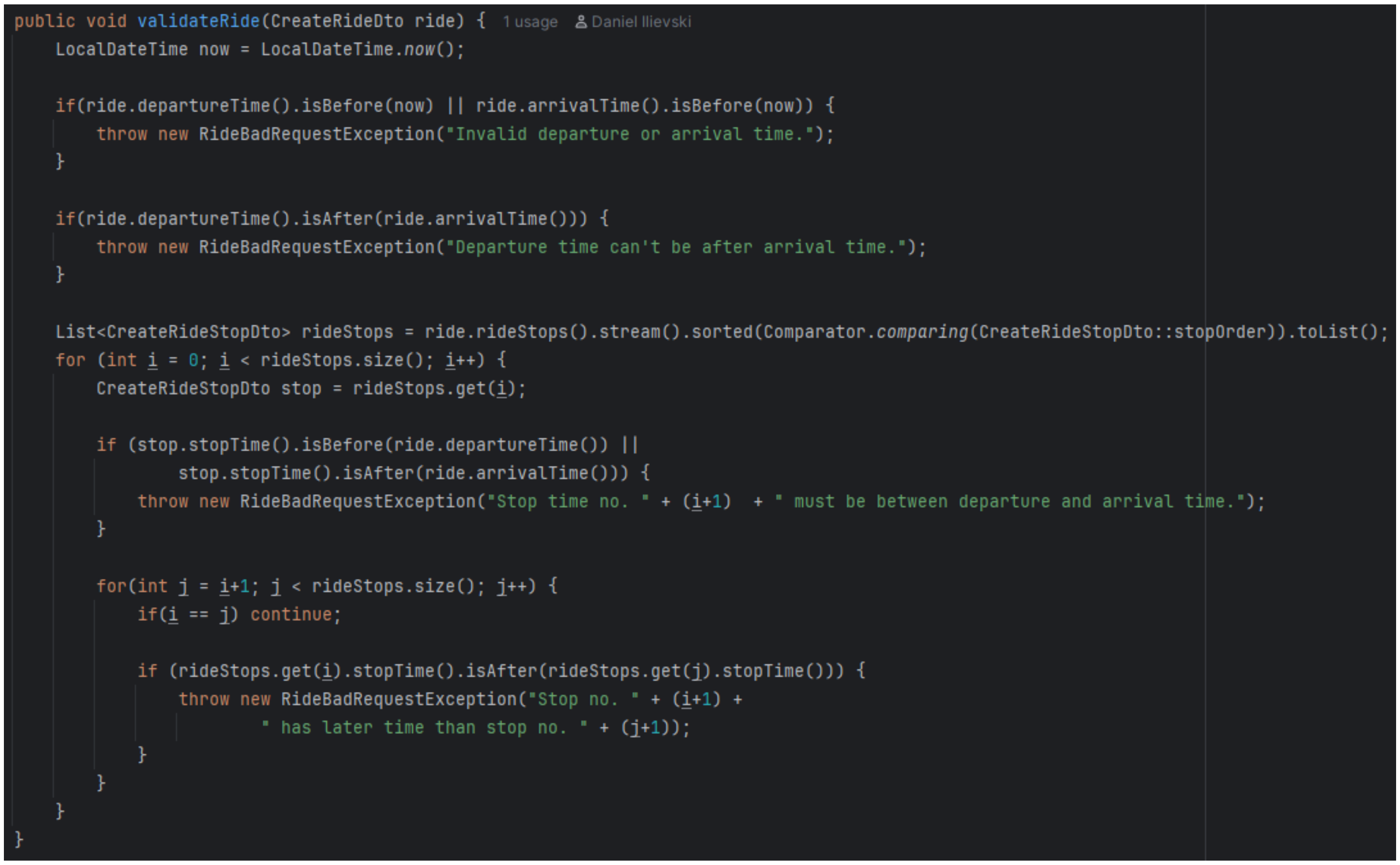
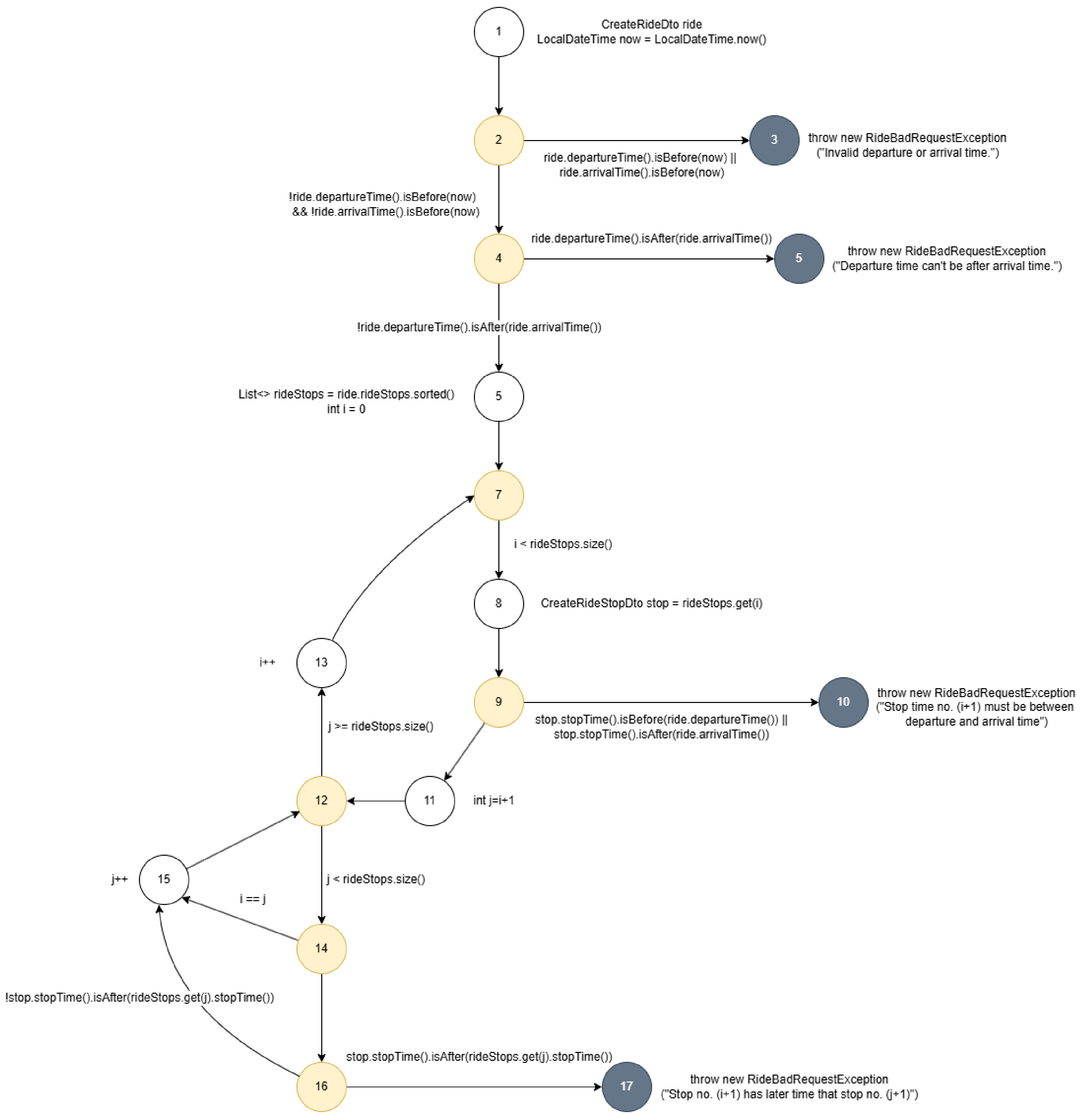
Software quality and testing

Wayzi - Graph Coverage



```java
public void validateRide(CreateRideDto ride) {  1 usage  Daniel Ilievski
    LocalDateTime now = LocalDateTime.now();

    if(ride.departureTime().isBefore(now) || ride.arrivalTime().isBefore(now)) {
        throw new RideBadRequestException("Invalid departure or arrival time.");
    }

    if(ride.departureTime().isAfter(ride.arrivalTime())) {
        throw new RideBadRequestException("Departure time can't be after arrival time.");
    }

    List<CreateRideStopDto> rideStops = ride.rideStops().stream().sorted(Comparator.comparing(CreateRideStopDto::stopOrder)).toList();
    for (int i = 0; i < rideStops.size(); i++) {
        CreateRideStopDto stop = rideStops.get(i);

        if (stop.stopTime().isBefore(ride.departureTime()) ||
                stop.stopTime().isAfter(ride.arrivalTime())) {
            throw new RideBadRequestException("Stop time no. " + (i+1)  + " must be between departure and arrival time.");
        }

        for(int j = i+1; j < rideStops.size(); j++) {
            if(i == j) continue;

            if (rideStops.get(i).stopTime().isAfter(rideStops.get(j).stopTime())) {
                throw new RideBadRequestException("Stop no. " + (i+1) +
                        " has later time than stop no. " + (j+1));
            }
        }
    }
}
```

Control Flow Graph:


1
CreateRideDto ride
LocalDateTime now = LocalDateTime.now()
2
ride.departureTime().isBefore(now) \|\| ride.arrivalTime().isBefore(now)
3
throw new RideBadRequestException ("Invalid departure or arrival time.")
!ride.departureTime().isBefore(now) && !ride.arrivalTime().isBefore(now)
4
ride.departureTime().isAfter(ride.arrivalTime())
5
throw new RideBadRequestException ("Departure time can't be after arrival time.")
!ride.departureTime().isAfter(ride.arrivalTime())
5
List<> rideStops = ride.rideStops.sorted()
int i = 0
7
i < rideStops.size()
8
CreateRideStopDto stop = rideStops.get(i)
i++
13
9
stop.stopTime().isBefore(ride.departureTime()) \|\| stop.stopTime().isAfter(ride.arrivalTime())
10
throw new RideBadRequestException ("Stop time no. (i+1) must be between departure and arrival time")
j >= rideStops.size()
12
11
int j=i+1
j++
15
i == j
j < rideStops.size()
14
!stop.stopTime().isAfter(rideStops.get(j).stopTime())
16
stop.stopTime().isAfter(rideStops.get(j).stopTime())
17
throw new RideBadRequestException ("Stop no. (i+1) has later time that stop no. (j+1)")

Prime Path Coverage

All prime paths:

1. [1,2,3]
2. [1,2,4,5]
3. [1,2,4,6,18]
4. [1,2,4,6,7,8,9,10]
5. [1,2,4,6,7,8,9,11,12,13]
6. [1,2,4,6,7,8,9,11,12,14,15]
7. [1,2,4,6,7,8,9,11,12,14,16,17]
8. [1,2,4,6,7,8,9,11,12,14,16,15]

9. [7,8,9,11,12,13,7],

10. [8,9,11,12,13,7,8]

11. [9,11,12,13,7,8,9]

12. [11,12,13,7,8,9,11]

12/2. [11,12,13,7,8,9,10],

13. [12,14,15,12]
14. [12,14,16,15,12]
15. [12,13,7,8,9,11,12]

16. [13,7,8,9,11,12,13],

17/1. [13,7,8,9,11,12,14,16,17]

17/2. [13,7,8,9,11,12,14,16,15]

18. [14,15,12,13,7,8,9,11]

19/1. [14,15,12,14],

19/2. [14,16,15,12,14]

20. [14,16,15,12,13,7,8,9,10]

20. [14,16,15,12,13,7,8,9,10]

21. [14,16,15,12,13,7,8,9,11]

22. [15,12,14,16,15],

23. [15,12,14,16,17]

24. [15,12,14,15]

25. [16,15,12,14,16]

Test Path ID	Test Paths	Test Requirements
1	[1,2,4,6,7,8,9,11,12,13,7,8,9, 11,12,13,7,8,9,10]	5, 9, 10, 12/2, 11, 12, 16, 15
2	[1,2,4,6,7,8,9,11,12,14,16,15,12,14,16, 15,12,14,16,15,12,14,16,17]	8, 23, 19/2, 25, 14
3	[1,2,3]	1
4	[1,2,4,5]	2
5	[1,2,4,6,7,8,9,11,12,14,15,12,14,15, 12,14,16,17]	6, 23, 24, 13
6	[1,2,4,6,7,8,9,11,12,14,15,12,14,16,17]	6, 23, 19/1, 13
7	[1,2,4,6,18]	3
8	[1,2,4,6,7,8,9,11,12,14,15,12,13,7 ,8,9,11,12,14,15,12,13,7,8,9,10]	6, 18, 17/2, 20, 15, 13
9	[1,2,4,6,7,8,9,11,12,13,7,8,9,10]	5, 9, 10, 12/2, 11
10	[1,2,4,6,7,8,9,11,12,13,7,8,9,11, 12,14,16,15,12,13,7,8,9,11,12,14,16,17]	5, 21, 17/1, 17/2, 9, 10, 11, 12, 15, 14
11	[1,2,4,6,7,8,9,10]	4
12	[1,2,4,6,7,8,9,11,12,14,16,15, 12,13,7,8,9,10]	7, 20, 14
13	[1,2,4,6,7,8,9,11,12,13,7,8,9, 11,12,14,16,17]	5, 17/1, 9, 10, 11, 12, 15
14	[1,2,4,6,7,8,9,11,12,14,16,17]	7

Test paths:

ID: 1 - [1,2,4,6,7,8,9,11,12,13,7,8,9, 11,12,13,7,8,9,10] - Infeasible

ID: 2 - [1,2,4,6,7,8,9,11,12,14,16,15,12,14,16,15,12,14,16,15,12,14,16,17]

Input: ride = {...; departureTime=now(); arrivalTime= now().plusHours(5); rideStops=[{...; stopOrder=1; stopTime=now().plusHours(2}, {...; stopOrder=2; stopTime=now.plusHours(4), {...; stopOrder=3; stopTime=now.plusHours(3}}] }

Output: RideBadRequestException("Stop no. 2 has later time than stop no. 3.")

ID: 3 - [1,2,3]

Input: ride = {...; departureTime=now().minusHours(2); arrivalTime= now().plusHours(5); rideStops=[] }

Output: RideBadRequestException("Invalid departure or arrival time.")

ID: 4 - [1,2,4,5]

Input: ride = {...; departureTime=now().plusHours(5); arrivalTime= now().plusHours(2); rideStops=[] }

Output: RideBadRequestException(“Departure time can’t be after arrival time.”)

ID: 5 - [1,2,4,6,7,8,9,11,12,14,15,12,14,15,12,14,16,17] - Infeasible

ID: 6 - [1,2,4,6,7,8,9,11,12,14,15,12,14,16,17] – Infeasible

ID: 7 - [1,2,4,6,18]

Input: ride = {...; departureTime=now().plusHours(2); arrivalTime= now().plusHours(5); rideStops=[] }

ID: 8 - [1,2,4,6,7,8,9,11,12,14,15,12,13,7, 8,9,11,12,14,15,12,13,7,8,9,10] – Infeasible

ID: 9 - [1,2,4,6,7,8,9,11,12,13,7,8,9,10] - Infeasible

ID: 10 - [1,2,4,6,7,8,9,11,12,13,7,8,9,11, 12,14,16,15,12,13,7,8,9,11,12,14,16,17] – Infeasible

ID: 11 – [1,2,4,6,7,8,9,10]

Input: ride = {...; departureTime=now().plusHours(3); arrivalTime= now().plusHours(5); rideStops=[{...; stopOrder=1; stopTime=now().plusHours(2}] }

Output: RideBadRequestException(“Stop time no. 1 must be between departure and arrival time.”)

ID: 12 – [1,2,4,6,7,8,9,11,12,14,16,15,12,13,7,8,9,10]

Input: ride = {...; departureTime=now().plusHours(3); arrivalTime= now().plusHours(5); rideStops=[{...; stopOrder=1; stopTime=now().plusHours(4}, {...; stopOrder=2; stopTime=now().plusHours(7}] }

Output: RideBadRequestException(“Stop time no. 2 must be between departure and arrival time.”)

ID: 13 – [1,2,4,6,7,8,9,11,12,13, 7, 8, 9,11,12,14,16,17] – Infeasible

ID: 14 - [1,2,4,6,7,8,9,11,12,14,16,17]

Input: ride = {...; departureTime=now(); arrivalTime= now().plusHours(5); rideStops=[{...; stopOrder=1; stopTime=now().plusHours(3}, {...; stopOrder=2; stopTime=now.plusHours(2)] }

Output: RideBadRequestException("Stop no. 1 has later time than stop no. 2.")