**Part 1, Step 1: Mutation analysis**

1. Original:

|  |  |
| --- | --- |
| Live Mutants # | 16 |
| Killed Mutants # | 148 |
| Total Mutants # | 164 |
| Mutant Score | 90.0% |

1. Added tests
2. testTwoSegmentsDiffMoreThan500SecondLarger(),

killed mutant is AOIU\_4.

1. testTwoSegmentsSumMoreThan500(),

killed mutant is AORB\_4

1. testTwoSegmentsDepartureMoreThan14DaysFromNowVeryLongTime(),

killed mutant is AORB\_43

1. testTwoSegmentsDepartureLessThan3DaysFromNowIsFreqFlierV1(),

killed mutants are AORB\_62, AORB\_63, AORB\_65

1. testInvalidPrice1V1(),

killed mutant is COR\_1

1. testInvalidPrice1Price2()

killed mutant is COR\_2

1. testInvalidPrice1AndDepartureTime()

killed mutant is COR\_4

1. testInvalidPrice1AndDuration()

killed mutant is COR\_6

1. Mutants that can not be killed:
2. ROR\_1.

Because original program P tests if price1 < 0 in

**if** **(**price1 **<** 0 **||** price2 **<** 0 **||** departureTime **<** System**.**currentTimeMillis**()** **||** duration **<** 0**)**

and then tests if price1 > 0 in

**if** **(**price1 **!=** 0 **&&** price2 **!=** 0**)**

Mutant program P’ tests if price1 < 0 in the same code block and then tests if price1 > 0 in

**if** **(**price1 **>** 0 **&&** price2 **!=** 0**) //this is the mutant code**

With other codes being the same, we can see that P and P’ are equivalent for all inputs, thus this mutant can not be killed.