Inhaltsverzeichnis

[Screenshot 1](#_Toc67234707)

[Testing of doCarTurn & calculatePath 2](#_Toc67234708)

[Testing of getWinner 3](#_Toc67234709)

[Testing of getWinner 4](#_Toc67234710)

Screenshot



## Testing of doCarTurn & calculatePath

**Equivalence Partitioning**

|  |  |
| --- | --- |
| c1 | Regular Bresenham's line algorithm |
| c2 | X-Axis inverted Bresenham's line algorithm |
| c3 | Y-Axis inverted Bresenham's line algorithm |
| c4 | X- and Y-Axis inverted Bresenham's line algorith |
| c5 | Horizontal line Bresenham's line algorithm |  |  |  |
| c6 | Vertical line Bresenham's line algorithm |  |  |  |
| d1 | Valid Movement on track |  |  |  |
| d2 | Crash into another player |  |  |  |
| d3 | Crash into wall |  |  |  |
| d4 | Valid Movement over finishline |  |  |  |

**Test log**

|  |  |  |  |
| --- | --- | --- | --- |
| **testcases** | **Expected** | **Eqivalence Partititoning** | **Result** |
| calculatePath\_BottomLeftToTopRightPath() | ExpectedList of PositionVectors | c2 | Passed |
| calculatePath\_VerticalLineUpPath() | ExpectedList of PositionVectors | c2, c6 | Passed |
| calculatePath\_VerticalLineDownPath() | ExpectedList of PositionVectors | c1, c6 | Passed |
| calculatePath\_BottomRightToTopLeftPath() | ExpectedList of PositionVectors | c3 | Passed |
| calculatePath\_HorizontalLineLeftToRightPath() | ExpectedList of PositionVectors | c1, c5 | Passed |
| calculatePath\_HorizontalLineRightToLeftPath() | ExpectedList of PositionVectors | c3, c5 | Passed |
| calculatePath\_TopRightToBottomLeftPath() | ExpectedList of PositionVectors | c4 | Passed |
| calculatePath\_TopLeftToBottomRightPath() | ExpectedList of PositionVectors | c1 | Passed |
| doCarTurn\_ValidPathOneStep() | ExpectedEndPosition, carCrashed=false | d1 | Passed |
| doCarTurn\_ValidPathMultipleStep() | ExpectedEndPosition, carCrashed=false | d1 | Passed |
| doCarTurn\_CrashWithOtherPlayer() | ExpectedEndPosition, carCrashed=true | d1, d2 | Passed |
| doCarTurn\_CrashWithWall() | ExpectedEndPosition, carCrashed=true | d3 | Passed |
| doCarTurn\_WinnerFoundReturnMethod() | ExpectedEndPosition, winner=player | d4 | Passed |

## Testing of getWinner

**Equivalence Partitioning**

|  |  |
| --- | --- |
| W1 | game is in progress (two or more cars are alive, no winner yet) |
| W2 | cross finish line correctly |
| W3 | cross finish line incorrectly |
| W4 | cross finish line and crashes |
| W5 | crosses finish line correctly, but has one lap to go |
| W6 | all car crashes except one |

**Test log**

|  |  |  |  |
| --- | --- | --- | --- |
| **testcases** | **Expected** | **Eqivalence Partititoning** | **Result** |
| getWinner\_GameIsInProgressFinishLineRight\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_GameIsInProgressFinishLineUp\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_GameIsInProgressFinishLineLeft\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_AllCarsAreAliveFinishLineRight\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_AllCarsAreAliveFinishLineUp\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_AllCarsAreAliveFinishLinLeft\_ReturnsNoWinner | NoWinner | W1 | Passed |
| getWinner\_CarCrossesFinishLineCorrectlyFinishLineRight\_ReturnsWinner | Winner | W2 | Passed |
| getWinner\_CarCrossesFinishLineCorrectlyFinishLineLeft\_ReturnsWinner | Winner | W2 | Passed |
| getWinner\_CarCrossesFinishLineBackwardsFinishLineRight\_ReturnsNoWinner | NoWinner | W3 | Passed |
| getWinner\_CarCrossesFinishLineBackwardsFinishLineUp\_ReturnsNoWinner | NoWinner | W3 | Passed |
| getWinner\_CarCrossesFinishLineCorrectlyAndCrashes\_ReturnsWinner | Winner | W4 | Passed |
| getWinner\_CarCrossesFinishLineBackwardsAndForwardAfterwardsFinishLineRight\_ReturnsNoWinner | NoWinner | W5 | Passed |
| GetWinner\_CarCrossesFinishLineBackwardsAndForwardAfterwardsFinishLineUp\_ReturnsNoWinner | NoWinner | W5 | Passed |
| getWinner\_AllCarsCrashesExceptOne\_ReturnsWinner | Winner | W6 | Passed |

## Testing of getWinner

**Equivalence Partitioning**

|  |  |
| --- | --- |
| **I1** | get existing Car Id |
| **I2** | get non-existing Car Id |
|  |  |
| **P1** | existing CarIndex |
| **P2** | non-existing CarIndex |
|  |  |
| **A1** | switch to next active player |
| **A2** | switch to next player only one player is active |

**Test log**

|  |  |  |  |
| --- | --- | --- | --- |
| **testcases** | **Expected** | **Eqivalence Partititoning** | **Result** |
| getCarId\_AskForExistingCarId\_ReturnsCarId | CarId | I1 | Passed |
| getCarId\_AskForNonExistingCarIndex\_ReturnsCarId | CarId | I1 | Passed |
|  |  |  |  |
| getCarPosition\_AskForNonExistingCarIndex\_ReturnsMinValueFromCharacter | Character.MinValue | P1 | Passed |
| getCarId\_AskForExistingCarIndex\_ReturnsNull | Null | P2 | Passed |
|  |  |  |  |
| switchToNextActiveCar\_AskForExistingCarIndex\_SwitchesCorrectly | Correct Switch | A1 | Passed |
| switchToNextActiveCar\_AskForExistingCarIndex\_StaysTheSameIndex | No Switch | A2 | Passed |