

```
1 const { assertEquals } = require("../test-framework");
2 const airport = require("../src/airport")
3
4
5 let = expectedOutput, actualOutput, result;
6 let input = '';
7 let testName = '';
8
9 //arrange
10 testName = 'Test 1 to see if plane has left the airport list';
11 input = 'plane1';
12 expectedOutput = input;
13
14 //act
15 actualOutput = airport.takeOff(input);
16
17 //assert
18 result = assertEquals(expectedOutput, actualOutput);
19
20
21 //report
22 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 20, Col 1 Spaces: 4 UTF-8 LF JavaScript Spell

```
1 //initial Arrange
2 const { assertEquals } = require("../test-framework");
3 const airport = require("../src/airport")
4
5
6 let = expectedOutput, actualOutput, result;
7 let input = '';
8 let testName = '';
9
10 //arrange
11 testName = 'Test 1 to see if plane has been added to airport';
12 input = 'plane1';
13 expectedOutput = input;
14
15 //act
16 actualOutput = airport.landPlane.push(input);
17
18 //assert
19 result = assertEquals(expectedOutput, actualOutput);
20
21
22 //report
23 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 6, Col 44 Spaces: 4 UTF-8 LF JavaScript Spell

```
1 const { assertEquals } = require("../testing-framework");
2 const airport = require("../src/airport");
3
4
5 let = expectedOutput, actualOutput, result;
6 let input = '';
7 let testName = '';
8
9 //arrange
10 testName = 'Test 1 try to add plane when it is full';
11 input = [1];
12 expectedOutput = 'full';
13
14 //act
15 actualOutput = airport.isAirportFull(input);
16
17 //assert
18 result = assertEquals(expectedOutput, actualOutput);
19
20
21 //report
22 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
23
24
```

```
1 const { assertEquals } = require("../testing-framework");
2 const airport = require("../src/airport");
3
4
5 let = expectedOutput, actualOutput, result;
6 let input = '';
7 let testName = '';
8
9 //arrange
10 testName = 'Test 1 confirming plane is in the airport';
11 input = 'plane1';
12 expectedOutput = 'plane in Airport';
13
14 //act
15 actualOutput = airport.confirmWherePlanesAre(input);
16
17 //assert
18 result = assertEquals(expectedOutput, actualOutput);
19
20
21 //report
22 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
```

```
1 const { assertEquals } = require("../testing-framework");
2 const airport = require("../src/airport");
3
4
5 let = expectedOutput, actualOutput, result;
6 let input = '';
7 let testName = '';
8
9 //arrange
10 testName = 'Test 1 see default capacity';
11 input = 10;
12 expectedOutput = input;
13
14 //act
15 actualOutput = airport.checkingCapacity(input);
16
17 //assert
18 result = assertEquals(expectedOutput, actualOutput);
19
20
21 //report
22 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
23
24 testName = 'TEST 2 change capacity';
25 input = 10;
26 expectedOutput = input
27
28 //act
29
30 actualOutput = airport.changingCapacity(input);
31
32 //assert
33 result = assertEquals(expectedOutput, actualOutput);
34
35
36 //report
37 console.log(`${testName} : ${result ? 'PASS' : 'FAIL'}`);
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 10, Col 42 Spaces: 4 UTF-8 LF {} JavaScript ✓ Spell 🔍

```
1 class Airport {
2   // here's a starting point for you//1
3   landedPlanes = [];
4
5   landPlane = inputPlane => {
6     this.landedPlanes.push(inputPlane);
7     return this.landedPlanes;
8   };
9   //2
10  defaultCapacity = new this.defaultCapacity(10);
11  airportCapacity = defaultCapacity;
12  newCapacity = [];
13  checkingCapacity() {
14    return this.airportCapacity;
15  }
16
17  changingCapacity = inputCapacity => {
18    this.airportCapacity = this.newCapacity.push(inputCapacity);
19    return this.airportCapacity;
20  }
21
22  // 3
23  capacity = 1;
24  planeList = new planeList(1);
25
26  isAirportFull = inputPlane => {
27    if (capacity == this.planeList.length) {
28      return 'is full';
29    } else {
30      this.planeList.push(inputPlane);
31    }
32  }
33  planeList1 = ['plane1'];
34
35  //4
36  takeOff = inputPlane => {
37    this.planeList1.pop(inputPlane);
38    return this.planeList;
39  }
40
41  //5
42  airportList = ['plane1', 'plane2', 'plane3'];
43  confirmingWherePlanesAre = inputPlane => {
44    if (this.airportList == inputPlane) {
45      return 'plane in Airport';
46    }
47  }
48
49  module.exports = Airport;
```

Restart Visual Studio Code to apply the latest update.

Update Now Later Release Notes

Ln 46, Col 1 Spaces: 2 UTF-8 LF {} JavaScript 🔍 Spell 🔍

```
EXPLORER  ...  README.md U  test-framework.js U  airport.js M  landedPlane.spec.js 1, U  airportCapacity.js U  defaultAirportCapacity.spec.js U  ...
AIRPORT-CHALLENGE  src > airport.js > ...
  src  ...
    airport.js  M
    airportCapacity... U
    confirmationO... U
    planesInAirAn... U
    testingAirport... U
    test  ...
      eslintrc.js  U
      gitignore  U
      airportsFull... U
      defaultAirport... U
      landedPlane... 1, U
      package-lock... U
      package.json  U
      planesInAirAn... U
      README.md  U
      specRunner.js  U
      test-framework... U

//Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Test 2 confirms it can be modified
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Prevent planes landing when full
const airport = new Airport(10);
const capacity = 30;
let inputPlanes = [];
//if(this.airport.length < capacity){
//this.airport.push(inputPlanes);
//}else return 'Capacity met'
//()
//Test 1 tries to add plane to airport when capacity is not full
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Test 2 tries to add plane when capacity is full
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
```

```
EXPLORER  ...  README.md U  test-framework.js U  airport.js M  landedPlane.spec.js 1, U  airportCapacity.js U  defaultAirportCapacity.spec.js U  ...
AIRPORT-CHALLENGE  src > airport.js > ...
  src  ...
    airport.js  M
    airportCapacity... U
    confirmationO... U
    planesInAirAn... U
    testingAirport... U
    test  ...
      eslintrc.js  U
      gitignore  U
      airportsFull... U
      defaultAirport... U
      landedPlane... 1, U
      package-lock... U
      package.json  U
      planesInAirAn... U
      README.md  U
      specRunner.js  U
      test-framework... U

//confirmationsPlanes(takeoff)
//--set length and try to add item into array
//planesInAir
//--remove item and check it is not there (boolean)
//pop method
const airport = [];
const inputPlane = {};
confirmingTakeoff = inputPlane => {
  //when (this.airport.pop(inputPlane);
  //return this.airport;
}
//Test 1 confirms plane is in airport
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Test 2 removes plane from airport
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Test 3 confirms plane is not in airport
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
// checking planes in airport
// const airport = ['plane1'];
```

```
EXPLORER  ...  README.md U  test-framework.js U  airport.js M  landedPlane.spec.js 1, U  airportCapacity.js U  defaultAirportCapacity.spec.js U  ...
AIRPORT-CHALLENGE  src > airport.js > ...
  src  ...
    airport.js  M
    airportCapacity... U
    confirmationO... U
    planesInAirAn... U
    testingAirport... U
    test  ...
      eslintrc.js  U
      gitignore  U
      airportsFull... U
      defaultAirport... U
      landedPlane... 1, U
      package-lock... U
      package.json  U
      README.md  U
      specRunner.js  U
      test-framework... U

//Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
//Test 3 confirms plane is not in airport
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
// checking planes in airport
const airport = ['plane1'];
const inputPlane = 'plane1';
const
confirmingPlane = airport.filter(airport) => {
  //for (this.airport.push(inputPlane) === inputPlane);
}
//Test 1 checks if plane is in airport
//--Arrange
//--Act
//--Assert result = assertEquals(expectedOutput, actualOutput);
//--report console.log(`${testName}: ${result ? 'PASS' : 'FAIL'}`);
```