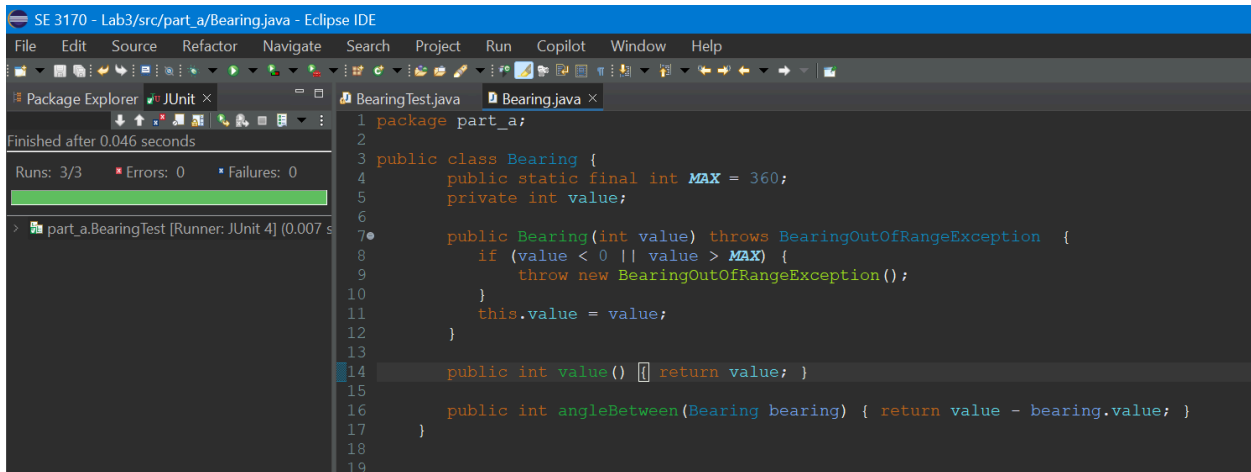


Part a:

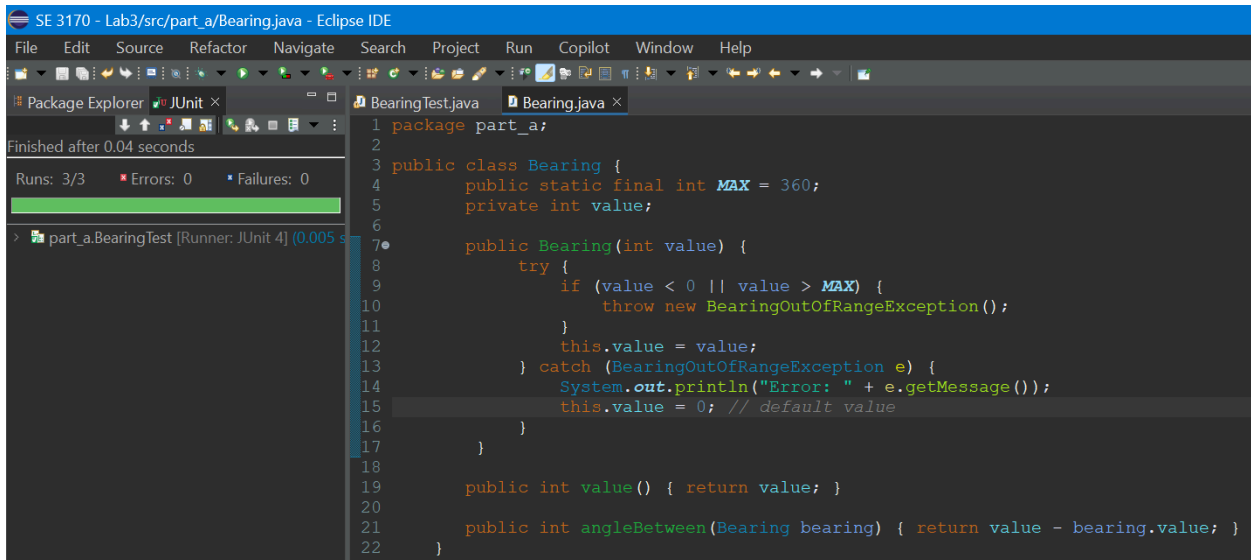
Part 1: Fixed exception



```
SE 3170 - Lab3/src/part_a/Bearing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Copilot Window Help
Package Explorer JUnit x
BearingTest.java Bearing.java x
Finished after 0.046 seconds
Runs: 3/3 Errors: 0 Failures: 0
> part_a.BearingTest [Runner: JUnit 4] (0.007 s)

1 package part_a;
2
3 public class Bearing {
4     public static final int MAX = 360;
5     private int value;
6
7     public Bearing(int value) throws BearingOutOfRangeException {
8         if (value < 0 || value > MAX) {
9             throw new BearingOutOfRangeException();
10        }
11        this.value = value;
12    }
13
14    public int value() { return value; }
15
16    public int angleBetween(Bearing bearing) { return value - bearing.value; }
17
18 }
19
```

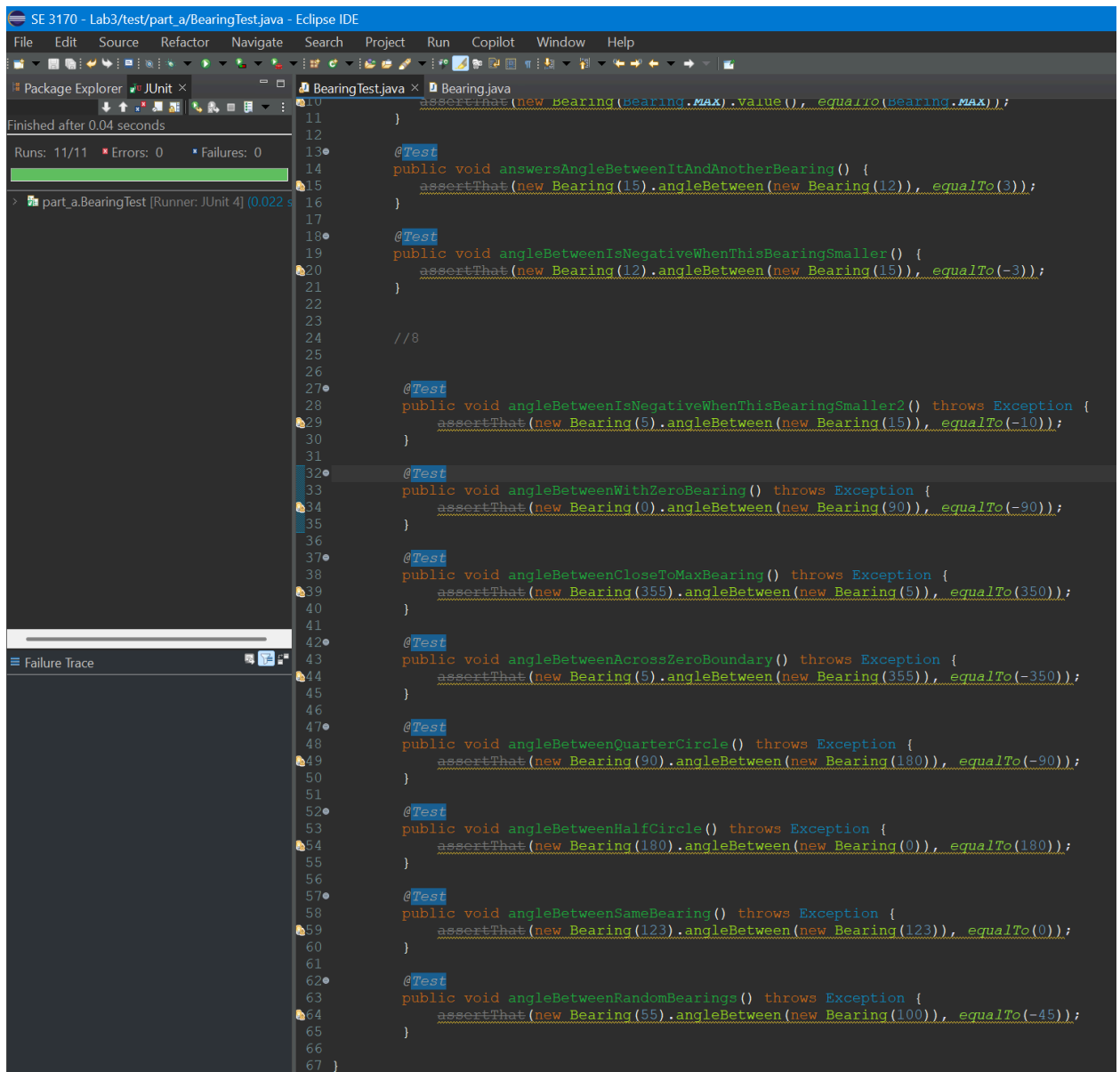
Using try catch



```
SE 3170 - Lab3/src/part_a/Bearing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Copilot Window Help
Package Explorer JUnit x
BearingTest.java Bearing.java x
Finished after 0.04 seconds
Runs: 3/3 Errors: 0 Failures: 0
> part_a.BearingTest [Runner: JUnit 4] (0.005 s)

1 package part_a;
2
3 public class Bearing {
4     public static final int MAX = 360;
5     private int value;
6
7     public Bearing(int value) {
8         try {
9             if (value < 0 || value > MAX) {
10                throw new BearingOutOfRangeException();
11            }
12            this.value = value;
13        } catch (BearingOutOfRangeException e) {
14            System.out.println("Error: " + e.getMessage());
15            this.value = 0; // default value
16        }
17    }
18
19    public int value() { return value; }
20
21    public int angleBetween(Bearing bearing) { return value - bearing.value; }
22
23 }
```

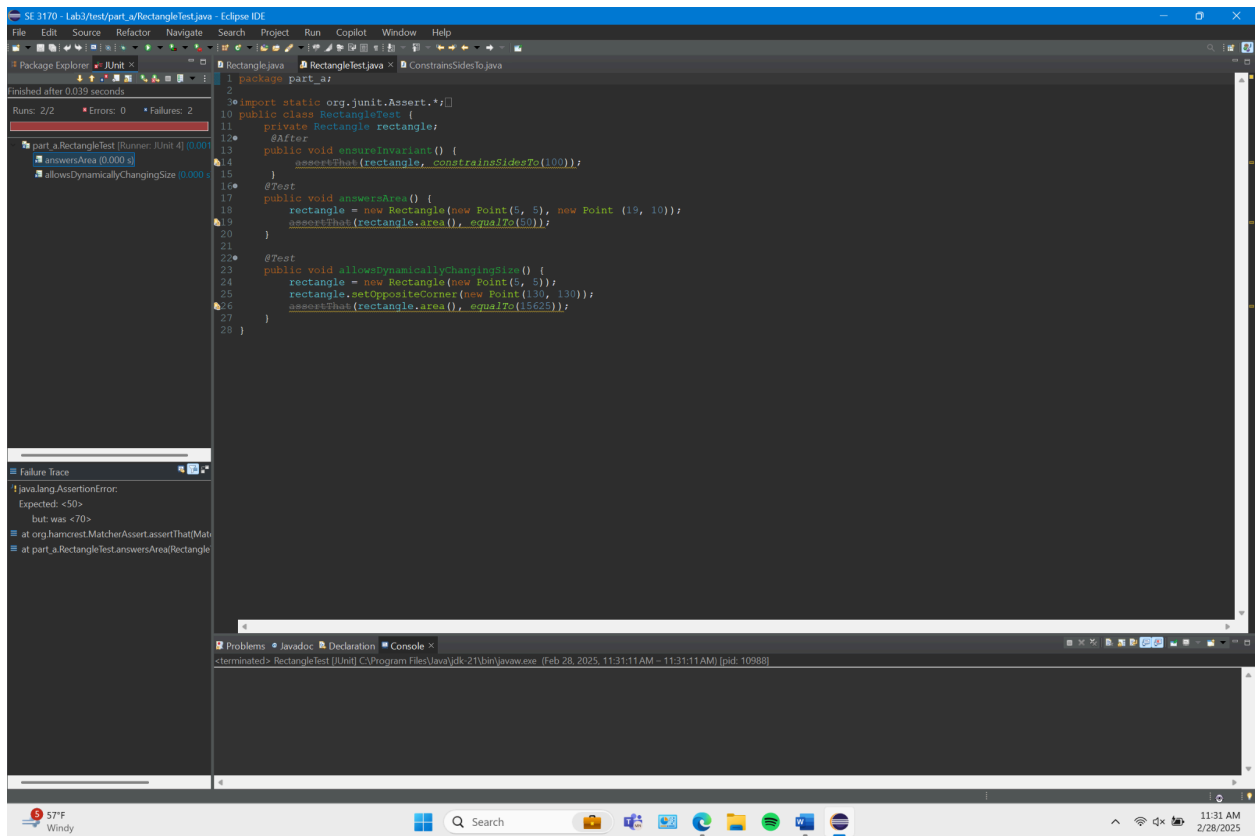
Part 2: Using try catch with 8 more test cases



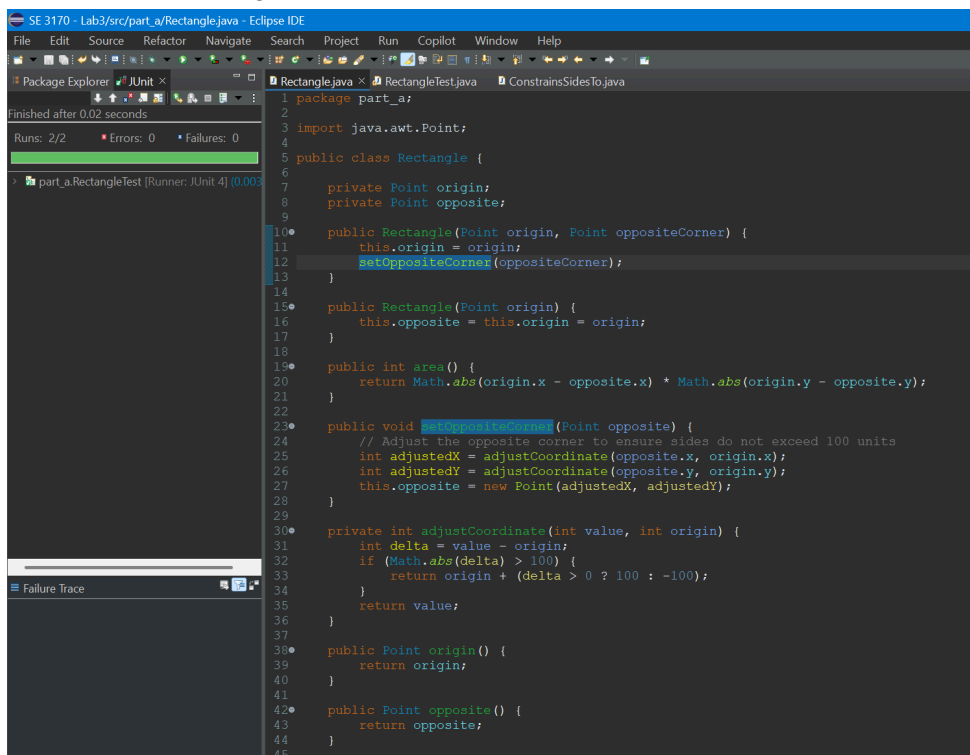
```
SE 3170 - Lab3/test/part_a/BearingTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Copilot Window Help
Package Explorer JUnit x
Finished after 0.04 seconds
Runs: 11/11 Errors: 0 Failures: 0
> part_a.BearingTest [Runner: JUnit 4] (0.022 s)

BearingTest.java x Bearing.java
10 assertEquals(new Bearing(Bearing.MAX).value(), equalTo(Bearing.MAX));
11 }
12
13 @Test
14 public void answersAngleBetweenItAndAnotherBearing() {
15     assertEquals(new Bearing(15).angleBetween(new Bearing(12)), equalTo(3));
16 }
17
18 @Test
19 public void angleBetweenIsNegativeWhenThisBearingSmaller() {
20     assertEquals(new Bearing(12).angleBetween(new Bearing(15)), equalTo(-3));
21 }
22
23 //8
24
25
26
27 @Test
28 public void angleBetweenIsNegativeWhenThisBearingSmaller2() throws Exception {
29     assertEquals(new Bearing(5).angleBetween(new Bearing(15)), equalTo(-10));
30 }
31
32 @Test
33 public void angleBetweenWithZeroBearing() throws Exception {
34     assertEquals(new Bearing(0).angleBetween(new Bearing(90)), equalTo(-90));
35 }
36
37 @Test
38 public void angleBetweenCloseToMaxBearing() throws Exception {
39     assertEquals(new Bearing(355).angleBetween(new Bearing(5)), equalTo(350));
40 }
41
42 @Test
43 public void angleBetweenAcrossZeroBoundary() throws Exception {
44     assertEquals(new Bearing(5).angleBetween(new Bearing(355)), equalTo(-350));
45 }
46
47 @Test
48 public void angleBetweenQuarterCircle() throws Exception {
49     assertEquals(new Bearing(90).angleBetween(new Bearing(180)), equalTo(-90));
50 }
51
52 @Test
53 public void angleBetweenHalfCircle() throws Exception {
54     assertEquals(new Bearing(180).angleBetween(new Bearing(0)), equalTo(180));
55 }
56
57 @Test
58 public void angleBetweenSameBearing() throws Exception {
59     assertEquals(new Bearing(123).angleBetween(new Bearing(123)), equalTo(0));
60 }
61
62 @Test
63 public void angleBetweenRandomBearings() throws Exception {
64     assertEquals(new Bearing(55).angleBetween(new Bearing(100)), equalTo(-45));
65 }
66
67 }
```

Part 3) Test failing



Fixed errors passing



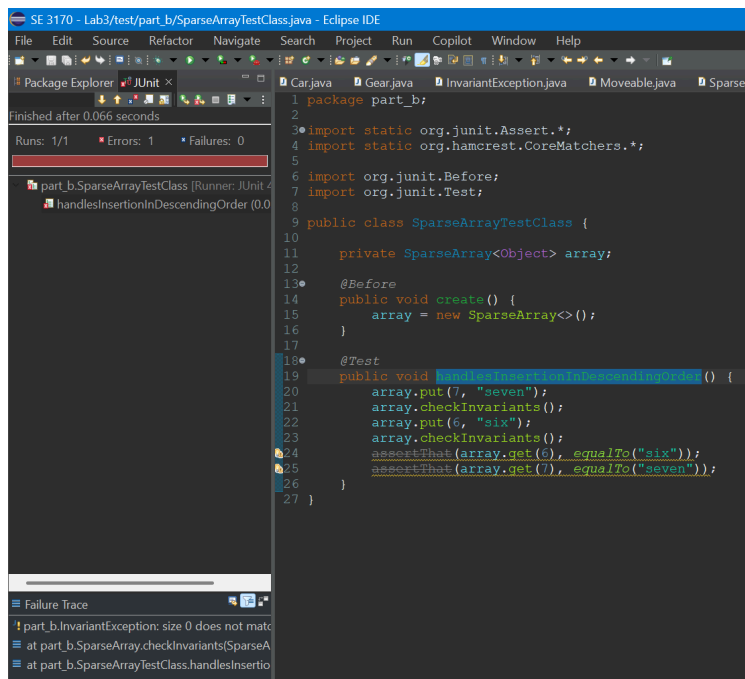
- 1) A throw exception triggers when an event disrupts it. It fixes code by enforcing constraints and preventing bugs.
- 2) A try catch block handles exceptions but trying and jumping to the catch if the try fills. Good for error handling and program crashes.
- 3) The difference is that a throw exception is used to create and signal an exception. A try catch handles the thrown exception.

Part b)

Binary search implementation

```
int binarySearch(int n, int[] nums, int size) {
    int low = 0;
    int high = size - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (nums[mid] == n) {
            return mid;
        } else if (nums[mid] < n) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}
```

Added test



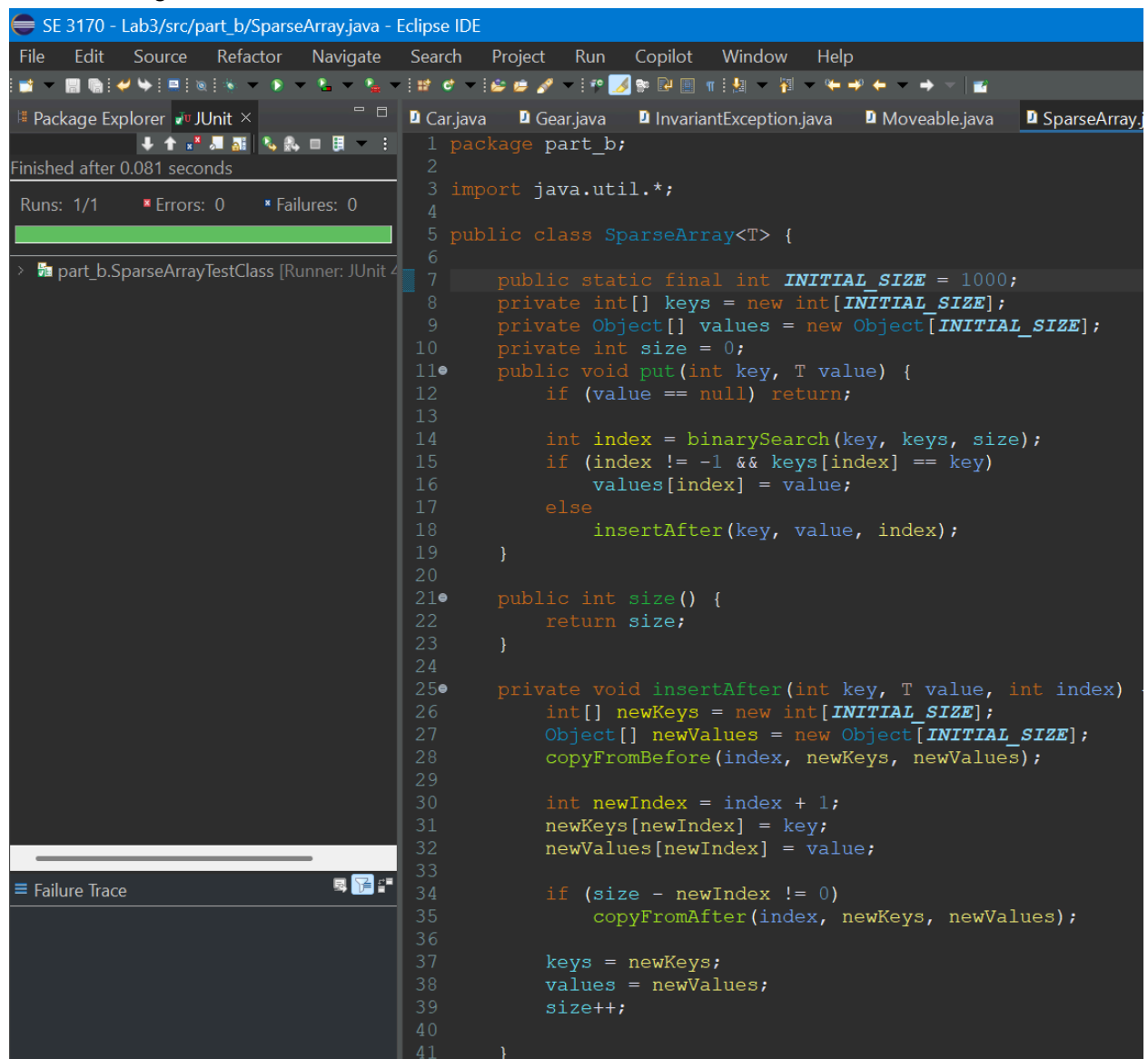
The screenshot shows the Eclipse IDE interface. The top toolbar includes icons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Copilot, Window, and Help. The Package Explorer on the left shows a project named 'part_b' with a sub-package 'test' containing 'SparseArrayTestClass.java'. The main editor displays the code for 'SparseArrayTestClass.java', which includes imports for JUnit and Hamcrest, and a test method 'handlesInsertionInDescendingOrder'. The bottom pane shows the 'Failure Trace' for a failed test run, indicating an 'InvariantException: size 0 does not match' at line 24.

```
1 package part_b;
2
3 import static org.junit.Assert.*;
4 import static org.hamcrest.CoreMatchers.*;
5
6 import org.junit.Before;
7 import org.junit.Test;
8
9 public class SparseArrayTestClass {
10
11     private SparseArray<Object> array;
12
13     @Before
14     public void create() {
15         array = new SparseArray<>();
16     }
17
18     @Test
19     public void handlesInsertionInDescendingOrder() {
20         array.put(7, "seven");
21         array.checkInvariants();
22         array.put(6, "six");
23         array.checkInvariants();
24         assertThat(array.get(6), equalTo("six"));
25         assertThat(array.get(7), equalTo("seven"));
26     }
27 }
```

Failure Trace

```
! part_b.InvariantException: size 0 does not match
at part_b.SparseArray.checkInvariants(SparseA
at part_b.SparseArrayTestClass.handlesInsertio
```

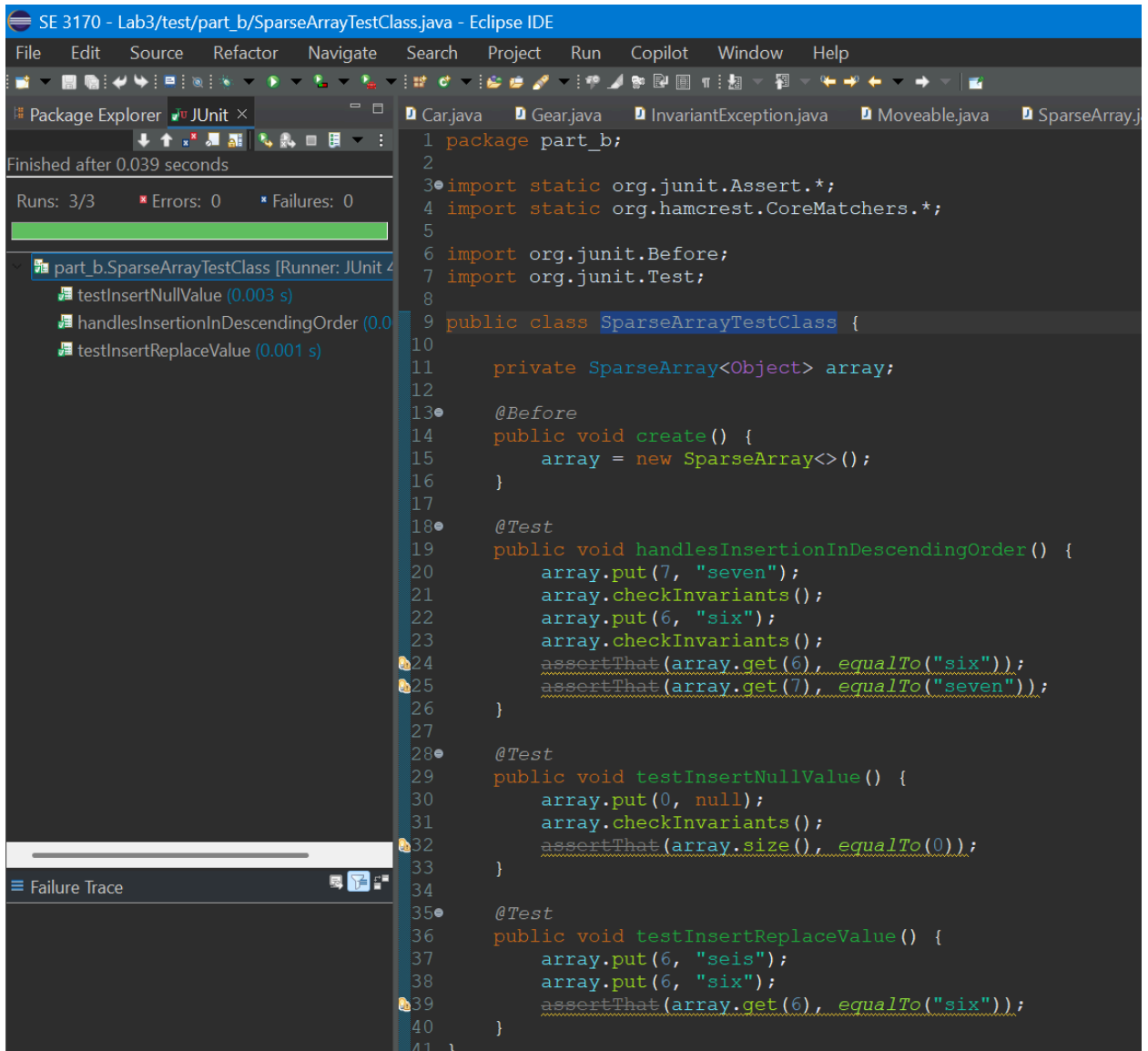
Fixed missing size increment



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Copilot, Window, and Help. The Package Explorer on the left shows a project named 'SE 3170 - Lab3/src/part_b' containing a package 'part_b' with a class 'SparseArrayTest' and a runner 'JUnit4'. The main editor displays the 'SparseArray.java' file. The code defines a generic class 'SparseArray<T>' with a fixed initial size of 1000. It includes methods for 'put', 'size', and 'insertAfter'. The 'put' method uses a binary search to find the insertion point. The 'insertAfter' method creates new arrays to shift elements and insert the new value, ensuring the 'size' is incremented correctly. The bottom panel shows a 'Failure Trace' which is currently empty.

```
1 package part_b;
2
3 import java.util.*;
4
5 public class SparseArray<T> {
6
7     public static final int INITIAL_SIZE = 1000;
8     private int[] keys = new int[INITIAL_SIZE];
9     private Object[] values = new Object[INITIAL_SIZE];
10    private int size = 0;
11    public void put(int key, T value) {
12        if (value == null) return;
13
14        int index = binarySearch(key, keys, size);
15        if (index != -1 && keys[index] == key)
16            values[index] = value;
17        else
18            insertAfter(key, value, index);
19    }
20
21    public int size() {
22        return size;
23    }
24
25    private void insertAfter(int key, T value, int index) {
26        int[] newKeys = new int[INITIAL_SIZE];
27        Object[] newValues = new Object[INITIAL_SIZE];
28        copyFromBefore(index, newKeys, newValues);
29
30        int newIndex = index + 1;
31        newKeys[newIndex] = key;
32        newValues[newIndex] = value;
33
34        if (size - newIndex != 0)
35            copyFromAfter(index, newKeys, newValues);
36
37        keys = newKeys;
38        values = newValues;
39        size++;
40    }
41 }
```

2 more test cases running

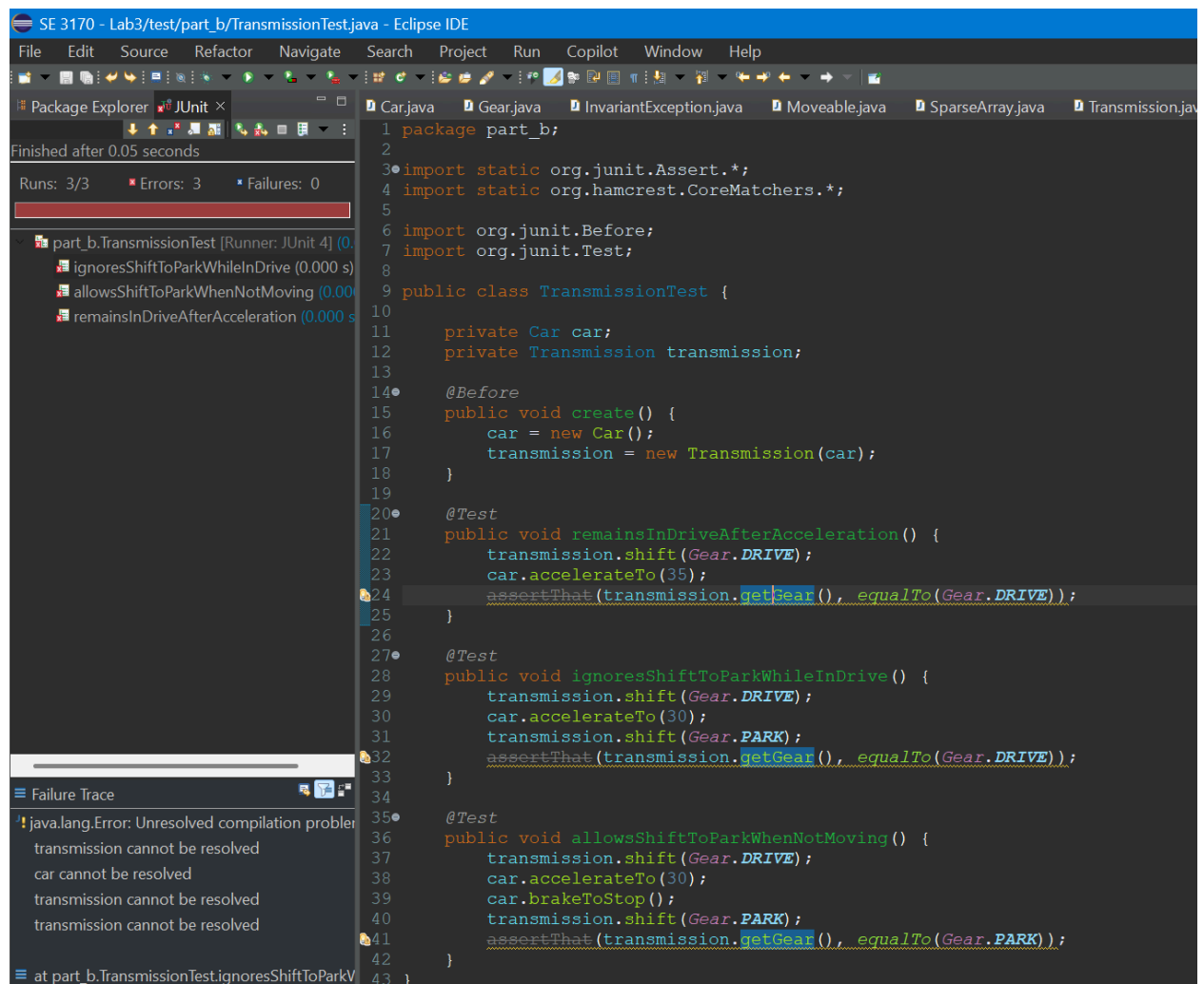


The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Copilot, Window, and Help. The Package Explorer on the left shows the project structure with 'part_b.SparseArrayTestClass' selected. The Run and Debug console shows the test results: 'Finished after 0.039 seconds', 'Runs: 3/3', 'Errors: 0', and 'Failures: 0'. The main editor displays the Java code for 'SparseArrayTestClass'.

```
SE 3170 - Lab3/test/part_b/SparseArrayTestClass.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Copilot Window Help
Package Explorer JUnit x
Finished after 0.039 seconds
Runs: 3/3 Errors: 0 Failures: 0
part_b.SparseArrayTestClass [Runner: JUnit 4]
  testInsertNullValue (0.003 s)
  handlesInsertionInDescendingOrder (0.001 s)
  testInsertReplaceValue (0.001 s)

1 package part_b;
2
3 import static org.junit.Assert.*;
4 import static org.hamcrest.CoreMatchers.*;
5
6 import org.junit.Before;
7 import org.junit.Test;
8
9 public class SparseArrayTestClass {
10     private SparseArray<Object> array;
11
12     @Before
13     public void create() {
14         array = new SparseArray<>();
15     }
16
17     @Test
18     public void handlesInsertionInDescendingOrder() {
19         array.put(7, "seven");
20         array.checkInvariants();
21         array.put(6, "six");
22         array.checkInvariants();
23         assertThat(array.get(6), equalTo("six"));
24         assertThat(array.get(7), equalTo("seven"));
25     }
26
27     @Test
28     public void testInsertNullValue() {
29         array.put(0, null);
30         array.checkInvariants();
31         assertThat(array.size(), equalTo(0));
32     }
33
34     @Test
35     public void testInsertReplaceValue() {
36         array.put(6, "seis");
37         array.put(6, "six");
38         assertThat(array.get(6), equalTo("six"));
39     }
40 }
41
```

Transmission tests



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Copilot, Window, and Help. The Package Explorer on the left shows the project structure with 'part_b.TransmissionTest' selected. The main editor displays the source code for 'TransmissionTest.java'. The code includes imports for JUnit and Hamcrest, and defines a 'TransmissionTest' class with three test methods: 'remainsInDriveAfterAcceleration', 'ignoresShiftToParkWhileInDrive', and 'allowsShiftToParkWhenNotMoving'. The 'ignoresShiftToParkWhileInDrive' method has a compilation error: 'java.lang.Error: Unresolved compilation problem: transmission cannot be resolved, car cannot be resolved, transmission cannot be resolved, transmission cannot be resolved'. The test runner output on the left shows that the tests passed successfully, with a total time of 0.05 seconds.

```
1 package part_b;
2
3 import static org.junit.Assert.*;
4 import static org.hamcrest.CoreMatchers.*;
5
6 import org.junit.Before;
7 import org.junit.Test;
8
9 public class TransmissionTest {
10
11     private Car car;
12     private Transmission transmission;
13
14     @Before
15     public void create() {
16         car = new Car();
17         transmission = new Transmission(car);
18     }
19
20     @Test
21     public void remainsInDriveAfterAcceleration() {
22         transmission.shift(Gear.DRIVE);
23         car.accelerateTo(35);
24         assertThat(transmission.getGear(), equalTo(Gear.DRIVE));
25     }
26
27     @Test
28     public void ignoresShiftToParkWhileInDrive() {
29         transmission.shift(Gear.DRIVE);
30         car.accelerateTo(30);
31         transmission.shift(Gear.PARK);
32         assertThat(transmission.getGear(), equalTo(Gear.DRIVE));
33     }
34
35     @Test
36     public void allowsShiftToParkWhenNotMoving() {
37         transmission.shift(Gear.DRIVE);
38         car.accelerateTo(30);
39         car.brakeToStop();
40         transmission.shift(Gear.PARK);
41         assertThat(transmission.getGear(), equalTo(Gear.PARK));
42     }
43 }
```

Failure Trace

- java.lang.Error: Unresolved compilation problem
- transmission cannot be resolved
- car cannot be resolved
- transmission cannot be resolved
- transmission cannot be resolved

at part_b.TransmissionTest.ignoresShiftToParkWhileInDrive(TransmissionTest.java:32)

Transmission tests with before instructions passing

The screenshot shows the Eclipse IDE interface. The top toolbar includes menus like File, Edit, Source, Refactor, Navigate, Search, Project, Run, Copilot, Window, and Help. The Package Explorer on the left shows a project structure with a package named 'part_b'. The main editor displays the file 'TransmissionTest.java' with the following code:

```
1 package part_b;
2
3 import static org.junit.Assert.*;
4 import static org.hamcrest.CoreMatchers.*;
5
6 import org.junit.Before;
7 import org.junit.Test;
8
9 public class TransmissionTest {
10
11     private Car car;
12     private Transmission transmission;
13
14     @Before
15     public void create() {
16         car = new Car();
17         transmission = new Transmission(car);
18     }
19
20     @Test
21     public void remainsInDriveAfterAcceleration() {
22         transmission.shift(Gear.DRIVE);
23         car.accelerateTo(35);
24         assertThat(transmission.getGear(), equalTo(Gear.DRIVE));
25     }
26
27     @Test
28     public void ignoresShiftToParkWhileInDrive() {
29         transmission.shift(Gear.DRIVE);
30         car.accelerateTo(30);
31         transmission.shift(Gear.PARK);
32         assertThat(transmission.getGear(), equalTo(Gear.DRIVE));
33     }
34
35     @Test
36     public void allowsShiftToParkWhenNotMoving() {
37         transmission.shift(Gear.DRIVE);
38         car.accelerateTo(30);
39         car.brakeToStop();
40         transmission.shift(Gear.PARK);
41         assertThat(transmission.getGear(), equalTo(Gear.PARK));
42     }
43 }
```

Below the code editor, the 'Failure Trace' panel shows the following error messages:

- java.lang.Error: Unresolved compilation problem:
- transmission cannot be resolved
- car cannot be resolved
- transmission cannot be resolved
- transmission cannot be resolved

The error messages are repeated for each of the four test methods.

What does the checkInvariants() method do?

It ensures the internal state of the SparseArray is consistent by verifying that the number of non-null values matches the size variable.

What does the Transmission.java class do?

Simulates the behavior of a car's transmission, ensuring proper gear shifts based on the car's state.