Part a:

Part 1: Fixed exception

SE 3170 - Lab3/src/part_a/Bearing.java - Eclipse IDE

Finished after 0.046 seconds

Runs: 3/3 Errors: 0 Failures: 0

part_a.BearingTest [Runner: JUnit 4] (0.007 s

```java
package part_a;

public class Bearing {
        public static final int MAX = 360;
        private int value;

        public Bearing(int value) throws BearingOutOfRangeException  {
            if (value < 0 || value > MAX) {
                throw new BearingOutOfRangeException();
            }
            this.value = value;
        }

        public int value() { return value; }

        public int angleBetween(Bearing bearing) { return value - bearing.value; }
    }
```

Using try catch

SE 3170 - Lab3/src/part_a/Bearing.java - Eclipse IDE

Finished after 0.04 seconds

Runs: 3/3 Errors: 0 Failures: 0

part_a.BearingTest [Runner: JUnit 4] (0.005 s

```java
package part_a;

public class Bearing {
        public static final int MAX = 360;
        private int value;

        public Bearing(int value) {
            try {
                if (value < 0 || value > MAX) {
                    throw new BearingOutOfRangeException();
                }
                this.value = value;
            } catch (BearingOutOfRangeException e) {
                System.out.println("Error: " + e.getMessage());
                this.value = 0; // default value
            }
        }

        public int value() { return value; }

        public int angleBetween(Bearing bearing) { return value - bearing.value; }
    }
```

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

Finished after 0.04 seconds

Runs: 11/11 Errors: 0 Failures: 0

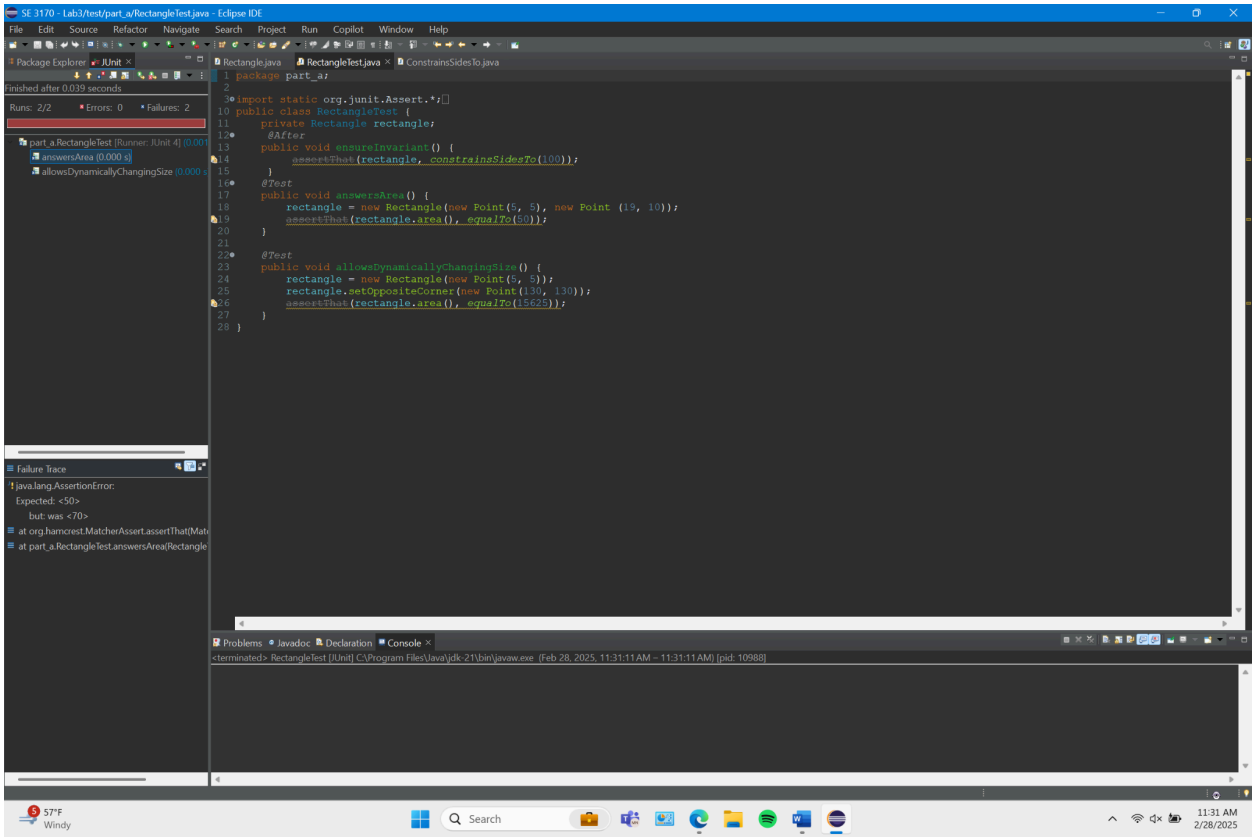
part_a.BearingTest [Runner: JUnit 4] (0.022 s

Failure Trace

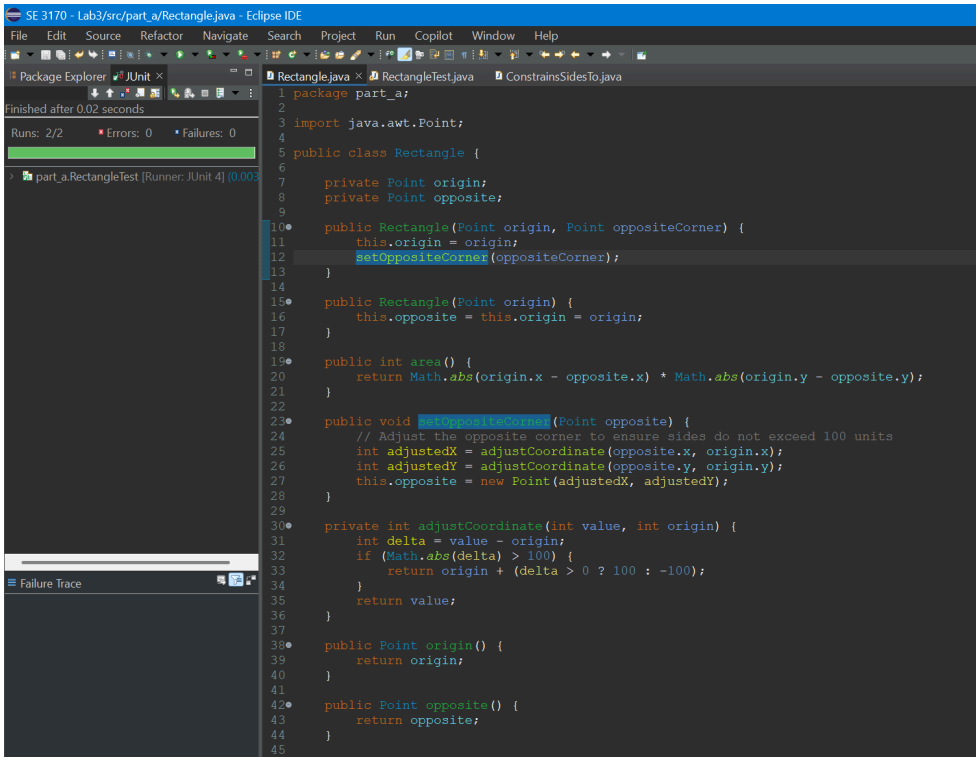
BearingTest.java | Bearing.java

```java
        assertThat(new Bearing(Bearing.MAX).value(), equalTo(Bearing.MAX));
    }

    @Test
    public void answersAngleBetweenItAndAnotherBearing() {
        assertThat(new Bearing(15).angleBetween(new Bearing(12)), equalTo(3));
    }

    @Test
    public void angleBetweenIsNegativeWhenThisBearingSmaller() {
        assertThat(new Bearing(12).angleBetween(new Bearing(15)), equalTo(-3));
    }


    //8


     @Test
     public void angleBetweenIsNegativeWhenThisBearingSmaller2() throws Exception {
         assertThat(new Bearing(5).angleBetween(new Bearing(15)), equalTo(-10));
     }

     @Test
     public void angleBetweenWithZeroBearing() throws Exception {
         assertThat(new Bearing(0).angleBetween(new Bearing(90)), equalTo(-90));
     }

     @Test
     public void angleBetweenCloseToMaxBearing() throws Exception {
         assertThat(new Bearing(355).angleBetween(new Bearing(5)), equalTo(350));
     }

     @Test
     public void angleBetweenAcrossZeroBoundary() throws Exception {
         assertThat(new Bearing(5).angleBetween(new Bearing(355)), equalTo(-350));
     }

     @Test
     public void angleBetweenQuarterCircle() throws Exception {
         assertThat(new Bearing(90).angleBetween(new Bearing(180)), equalTo(-90));
     }

     @Test
     public void angleBetweenHalfCircle() throws Exception {
         assertThat(new Bearing(180).angleBetween(new Bearing(0)), equalTo(180));
     }

     @Test
     public void angleBetweenSameBearing() throws Exception {
         assertThat(new Bearing(123).angleBetween(new Bearing(123)), equalTo(0));
     }

     @Test
     public void angleBetweenRandomBearings() throws Exception {
         assertThat(new Bearing(55).angleBetween(new Bearing(100)), equalTo(-45));
     }

}
```

Part 3) Test failing



Fixed errors passing



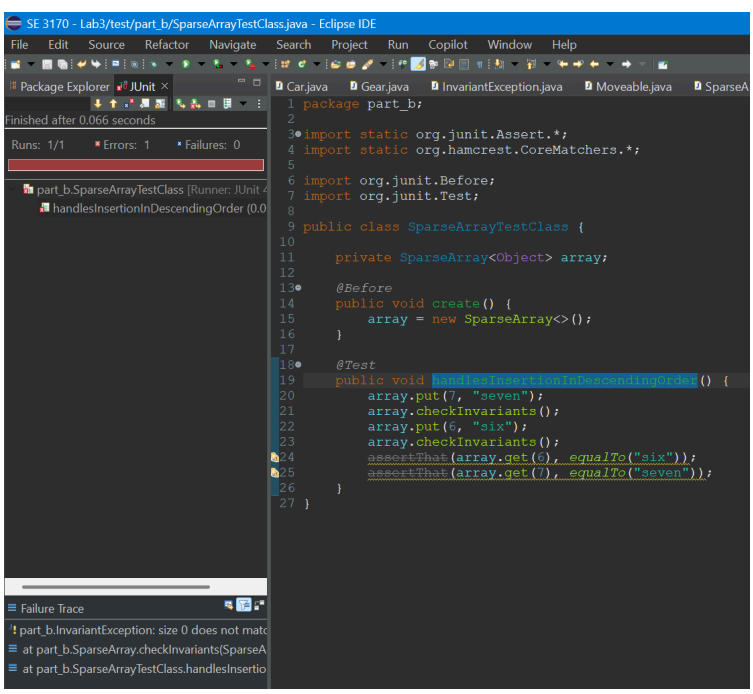
1) A throw exception triggers when an event disrupts it. It fixes code by enforcing constraints and preventing bugs.
2) A try catch block handles exceptions but trying and jumping to the catch if the try fills. Good for error handling and program crashes.
3) The difference is that a throw exception is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



Fixed missing size increment

SE 3170 - Lab3/src/part_b/SparseArray.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Copilot Window Help

Package Explorer | JUnit

Finished after 0.081 seconds

Runs: 1/1 Errors: 0 Failures: 0

part_b.SparseArrayTestClass [Runner: JUnit 4

Failure Trace

Car.java | Gear.java | InvariantException.java | Moveable.java | SparseArray.j

```java
package part_b;

import java.util.*;

public class SparseArray<T> {

    public static final int INITIAL_SIZE = 1000;
    private int[] keys = new int[INITIAL_SIZE];
    private Object[] values = new Object[INITIAL_SIZE];
    private int size = 0;
    public void put(int key, T value) {
        if (value == null) return;

        int index = binarySearch(key, keys, size);
        if (index != -1 && keys[index] == key)
            values[index] = value;
        else
            insertAfter(key, value, index);
    }

    public int size() {
        return size;
    }

    private void insertAfter(int key, T value, int index)
        int[] newKeys = new int[INITIAL_SIZE];
        Object[] newValues = new Object[INITIAL_SIZE];
        copyFromBefore(index, newKeys, newValues);

        int newIndex = index + 1;
        newKeys[newIndex] = key;
        newValues[newIndex] = value;

        if (size - newIndex != 0)
            copyFromAfter(index, newKeys, newValues);

        keys = newKeys;
        values = newValues;
        size++;

    }
```

2 more test cases running

SE 3170 - Lab3/test/part_b/SparseArrayTestClass.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Copilot Window Help

Package Explorer | JUnit

Finished after 0.039 seconds

Runs: 3/3 Errors: 0 Failures: 0

part_b.SparseArrayTestClass [Runner: JUnit 4

testInsertNullValue (0.003 s)

handlesInsertionInDescendingOrder (0.0

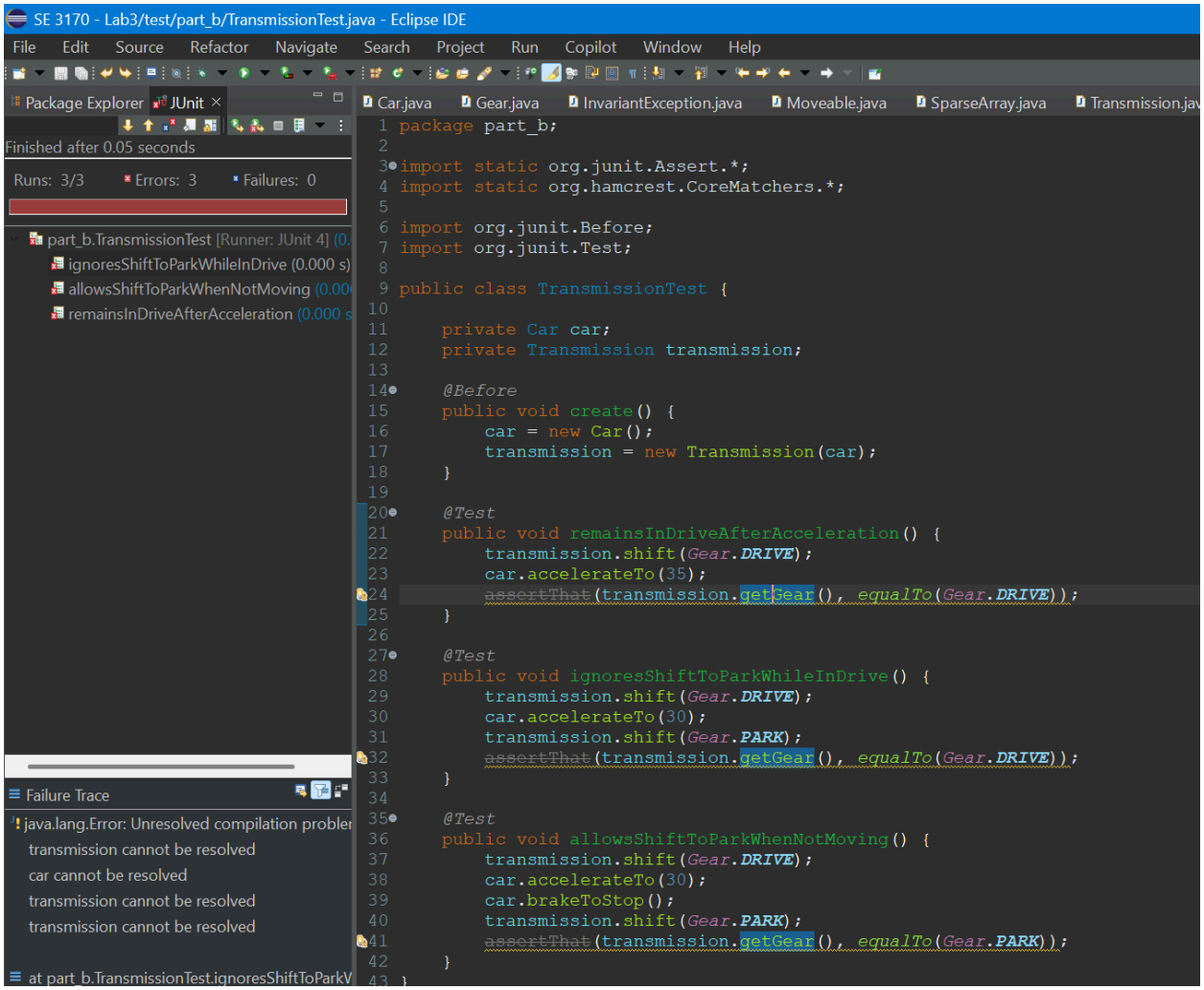
testInsertReplaceValue (0.001 s)

Failure Trace

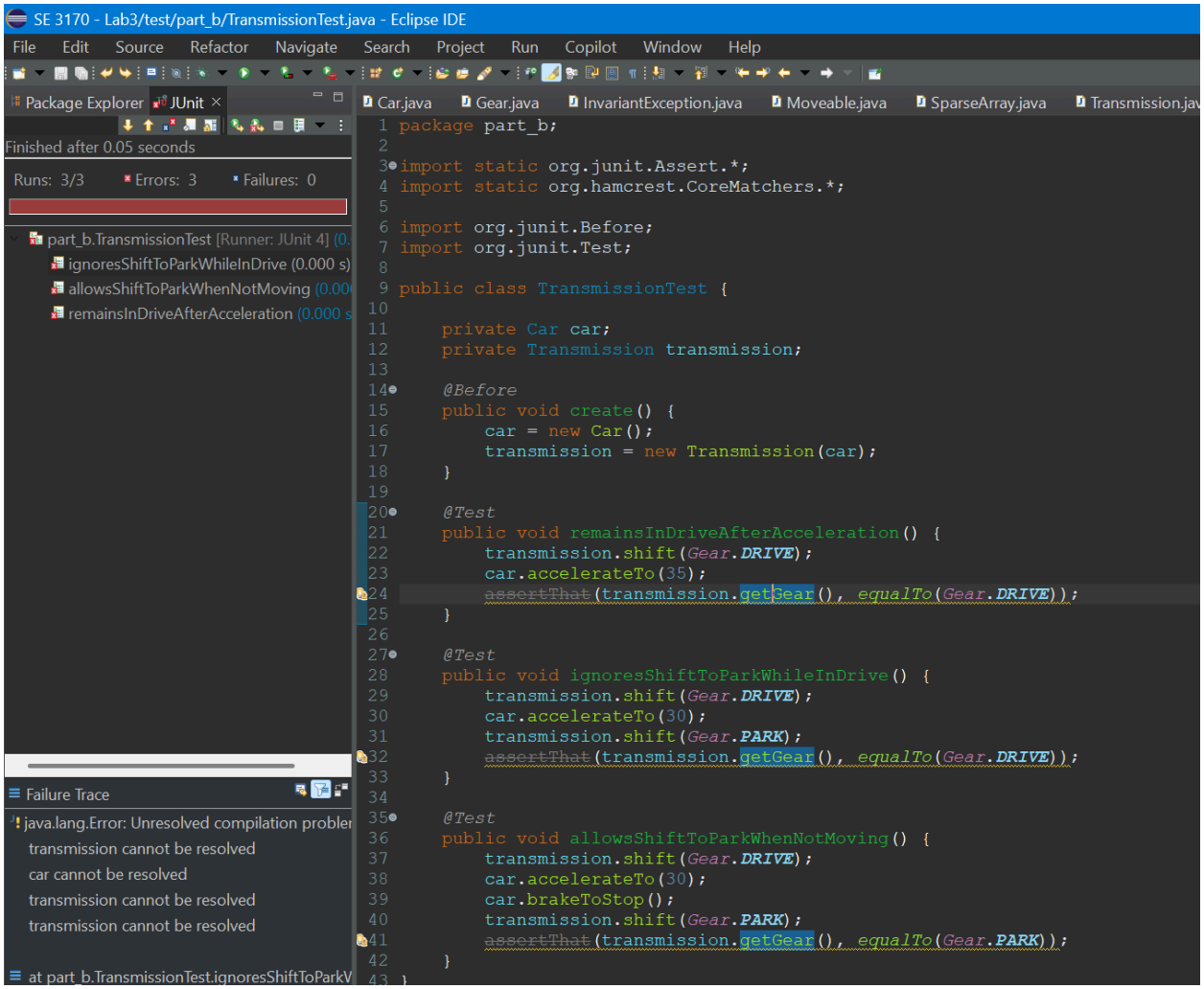
Car.java | Gear.java | InvariantException.java | Moveable.java | SparseArray.j

```java
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
28     @Test
29     public void testInsertNullValue() {
30         array.put(0, null);
31         array.checkInvariants();
32         assertThat(array.size(), equalTo(0));
33     }
34
35     @Test
36     public void testInsertReplaceValue() {
37         array.put(6, "seis");
38         array.put(6, "six");
39         assertThat(array.get(6), equalTo("six"));
40     }
41 }
```

Transmission tests



Transsion missions with before instructions passing



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.