Wyatt Duberstein

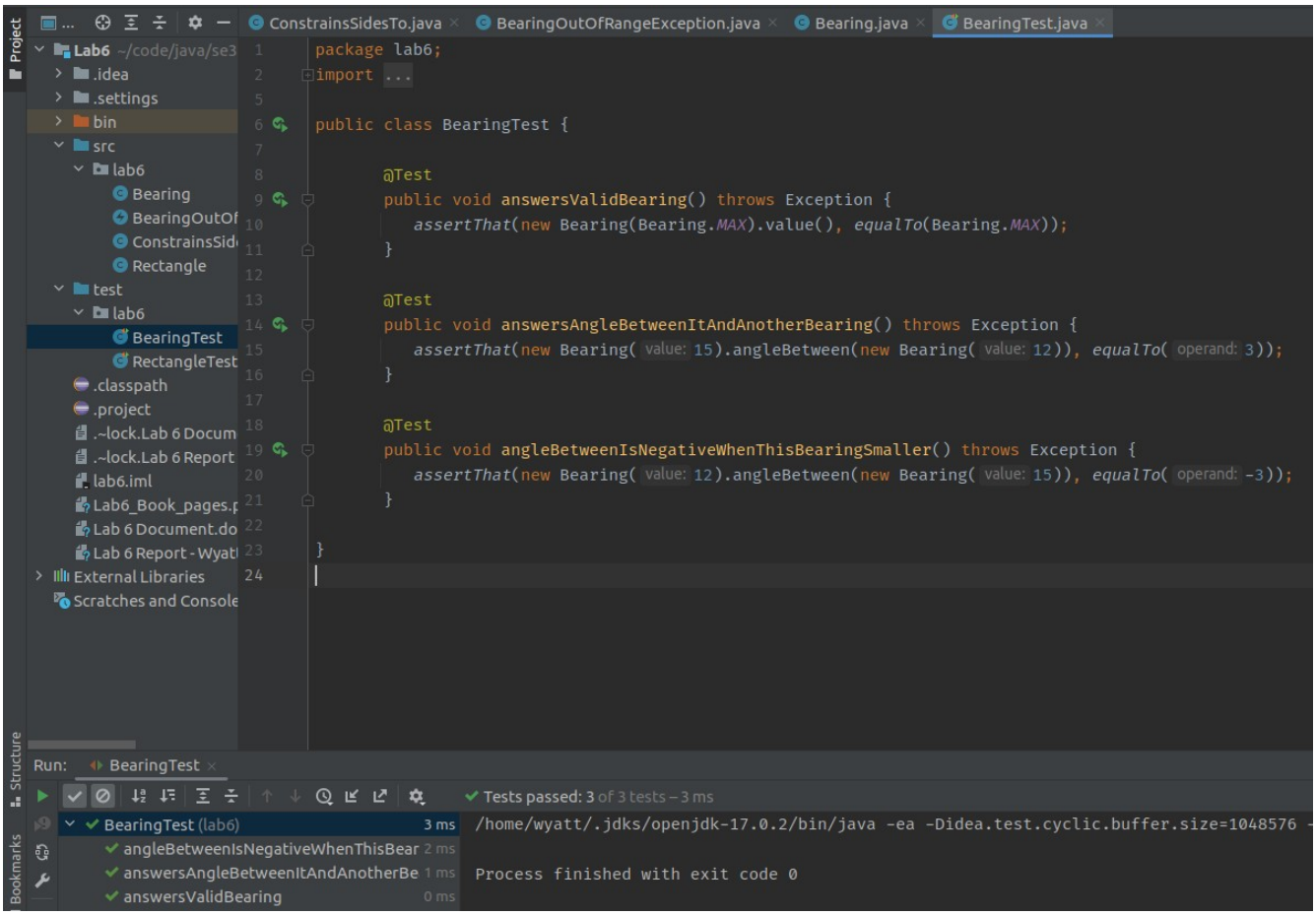
SE 317

06/25/2022

Lab 6 Report

Part 1:

1)



2)

```java
package lab6;
import ...

public class BearingTest {

        @Test
        public void answersValidBearing() throws Exception {
            try {
                assertThat(new Bearing(Bearing.MAX).value(), equalTo(Bearing.MAX));
            } catch (Exception ignored) {
            }
        }

        @Test
        public void answersAngleBetweenItAndAnotherBearing() throws Exception {
            try {
                assertThat(new Bearing( value: 15).angleBetween(new Bearing( value: 12)), equalTo( operand: 3)
            } catch (Exception ignored) {}
        }

        @Test
        public void angleBetweenIsNegativeWhenThisBearingSmaller() throws Exception {
            try {
                assertThat(new Bearing( value: 12).angleBetween(new Bearing( value: 15)), equalTo( operand: -3
            } catch (Exception ignored) {}
        }

}
```

```
Run: BearingTest
Tests passed: 3 of 3 tests – 1 ms
BearingTest (lab6)                                  1 ms
    angleBetweenIsNegativeWhenThisBear              1 ms
    answersAngleBetweenItAndAnotherBe               0 ms
    answersValidBearing                             0 ms

/home/wyatt/.jdks/openjdk-17.0.2/bin/java -ea -Didea.test.cyclic.buffer.size=1

Process finished with exit code 0
```

Part 2:

```java
@Test
public void partTwoTest1() throws Exception {
    assertThat(new Bearing( value: 0).angleBetween(new Bearing( value: 355)), equalTo( operand: -355));
}

@Test
public void partTwoTest2() throws Exception {
    assertThat(new Bearing( value: 355).angleBetween(new Bearing( value: 90)), equalTo( operand: 265));
}
public void partTwoTest3() throws Exception {
    assertThat(new Bearing( value: 90).angleBetween(new Bearing( value: 55)), equalTo( operand: 35));
}

@Test
public void partTwoTest4() throws Exception {
    assertThat(new Bearing( value: 55).angleBetween(new Bearing( value: 100)), equalTo( operand: -45));
}

@Test
public void partTwoTest5() throws Exception {
    assertThat(new Bearing( value: 100).angleBetween(new Bearing( value: 12)), equalTo( operand: 88));
}

@Test
public void partTwoTest6() throws Exception {
    assertThat(new Bearing( value: 12).angleBetween(new Bearing( value: 123)), equalTo( operand: -111));
}
@Test
public void partTwoTest7() throws Exception {
    assertThat(new Bearing( value: 123).angleBetween(new Bearing( value: 78)), equalTo( operand: 45));
}
@Test
public void partTwoTest8() throws Exception {
    assertThat(new Bearing( value: 78).angleBetween(new Bearing( value: 360)), equalTo( operand: -282));
}
```

BearingTest (lab6)	2 ms
angleBetweenIsNegativeWhenThisBear	2 ms
answersAngleBetweenItAndAnotherBe	0 ms
partTwoTest1	0 ms
partTwoTest2	0 ms
partTwoTest3	0 ms
partTwoTest4	0 ms
partTwoTest5	0 ms
partTwoTest6	0 ms
partTwoTest7	0 ms
partTwoTest8	0 ms
answersValidBearing	0 ms

Part 3:

1)

```
RectangleTest (lab6)                    24 ms
    answersArea                          8 ms
    allowsDynamicallyChangingSize       16 ms

/home/wyatt/.jdks/openjdk-17.0.2/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent

java.lang.AssertionError:
Expected: <50>
     but: was <70>
Expected :<50>
Actual   :<70>
<Click to see difference>


    at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20) <2 internal lines>
    at lab6.RectangleTest.answersArea(RectangleTest.java:19) <25 internal lines>


java.lang.AssertionError:
Expected: both sides must be <= 100
     but: was <Rectangle(origin java.awt.Point[x=5,y=5] opposite java.awt.Point[x=130,y=130])>
Expected :both sides must be <= 100
Actual   :<Rectangle(origin java.awt.Point[x=5,y=5] opposite java.awt.Point[x=130,y=130])>
<Click to see difference>


    at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20) <2 internal lines>
    at lab6.RectangleTest.ensureInvariant(RectangleTest.java:14) <24 internal lines>


Process finished with exit code 255
```

2)

```java
package lab6;

import ...
public class RectangleTest {
    private Rectangle rectangle;
     @After
    public void ensureInvariant() { assertThat(rectangle, constrainsSidesTo( length: 100)); }
    @Test
    public void answersArea() {
        rectangle = new Rectangle(new Point( x: 5,  y: 5), new Point ( x: 19,  y: 10));
        assertThat(rectangle.area(), equalTo( operand: 70));
    }

    @Test
    public void allowsDynamicallyChangingSize() {
        rectangle = new Rectangle(new Point( x: 5,  y: 5));
        rectangle.setOppositeCorner(new Point( x: 100,  y: 100));
        assertThat(rectangle.area(), equalTo( operand: 9025));
    }
}
```

```
✔ RectangleTest (lab6)                  2 ms
    ✔ answersArea                       2 ms
    ✔ allowsDynamicallyChangingSize     0 ms
```

3) Answer the following questions:

1. What is throw exception and how does it fix the code?

The throw keyword lets the compiler know that this specific method might throw, or is designed to throw, an exception at some point. This is so that the programmer can define a custom exception to throw as well, as you can write custom exceptions in java. It fixes the code because it tells the compiler that an exception might be thrown from this method

2. What is try-catch method and how does it fix the code?

The try/catch method is a method that “tries” a piece of code, knowing that it *could* throw an exception at some point within that code. If an exception is thrown in a specific piece of code, it does not stop the program unless the programmer sets it to do so. This creates two branches, one way if the code does not throw an exception, and the other way if it does. The programmer can choose to continue the program execution, stop the execution, or take a different path in the execution, in the event of an exception. This fixes the code in this example because it lets the compiler know that the exception is being handled and when it is thrown, the code knows where to go and what to do rather than if the try/catch block didn’t exist; the program would just end.

3. Is there any difference between throw exception and try-catch method? If yes, explain.

Yes. Throw exceptions are defined at the method level, meaning that any and all code in the method could throw the exception at any time. This is a very broad area for an exception to be thrown. With the try/catch exception, it allows the programmer to be more specific as to which lines of code might throw an exception. It also gives the programmer the ability to run other code in the event that an exception is thrown.