ComS 228 Recitation

Welcome welcome!

What is recitation for?

- Explaining confusing material
- Asking questions (or use Canvas Discussions board)
- Exam review
- Handing exams back

All recitation slides will be posted on Canvas

Today: 227 Review

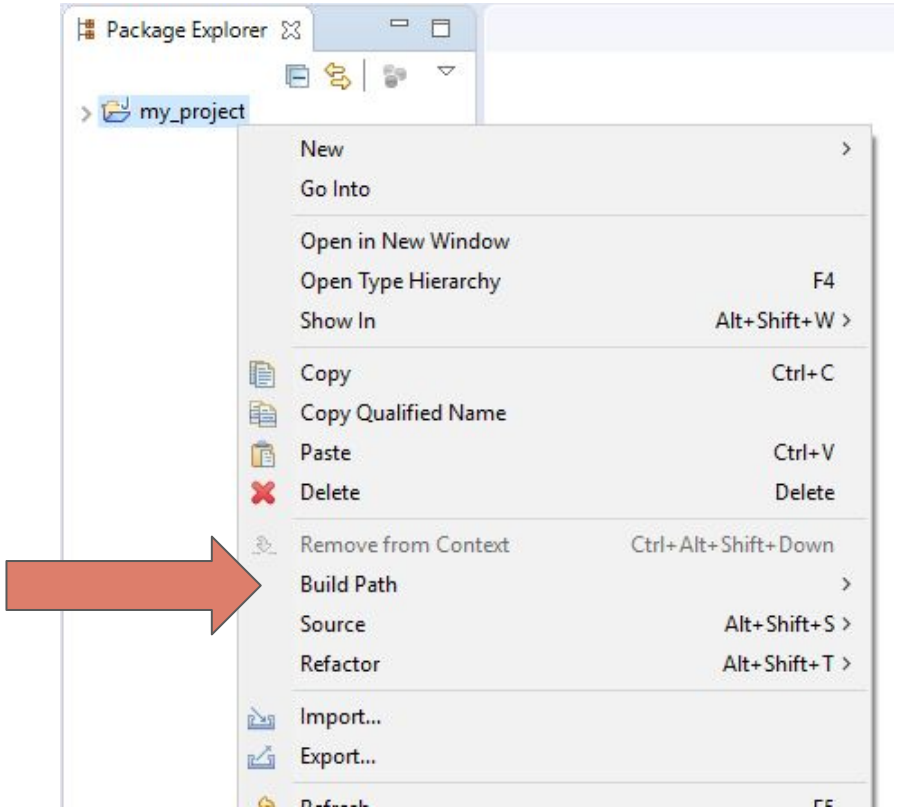
JUnit
Debugging

JUnit 5

Learn more at https://junit.org/junit5/docs/current/user-guide/

Adding JUnit 5 to your project

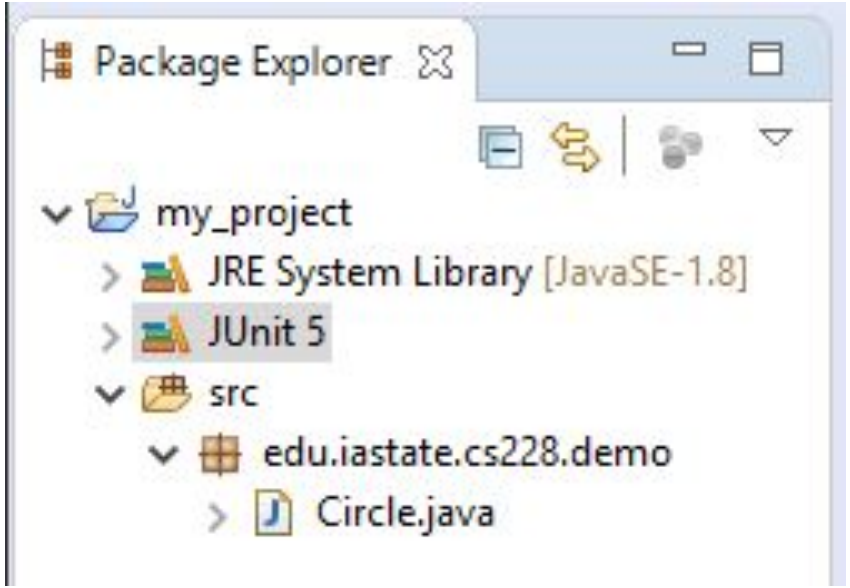
1. Right click on your project in the Package Explorer
2. Hover over “Build Path”
3. Select “Add Library”



Adding JUnit 5 to your project

4. Select “JUnit”, and click Next
5. Select “JUnit 5” from the dropdown, click Finish

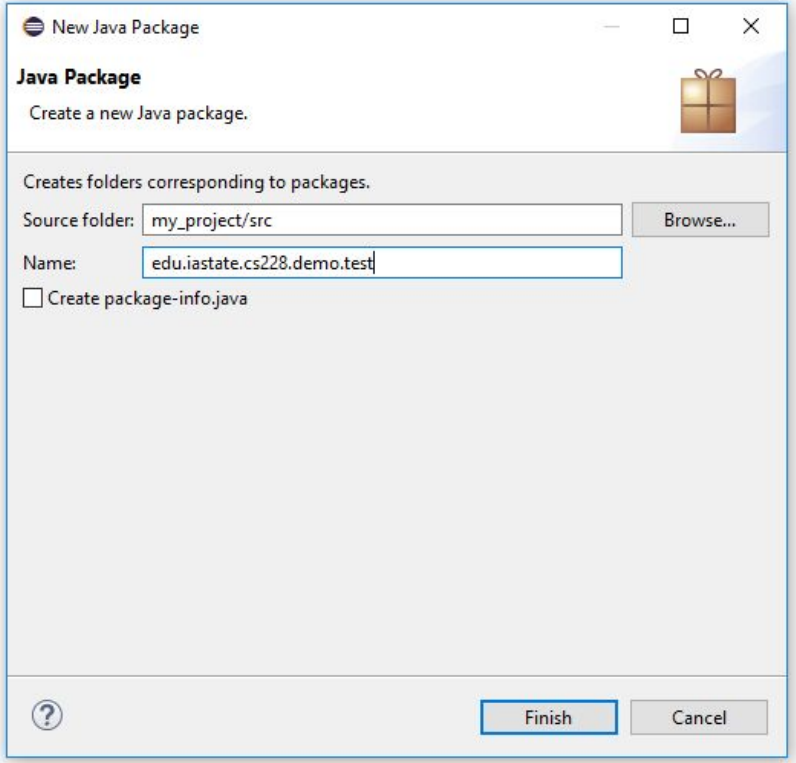
You should see the JUnit 5 library in your project



Create a test package

All your project files will be stored in “edu.iastate.cs228.proj__”

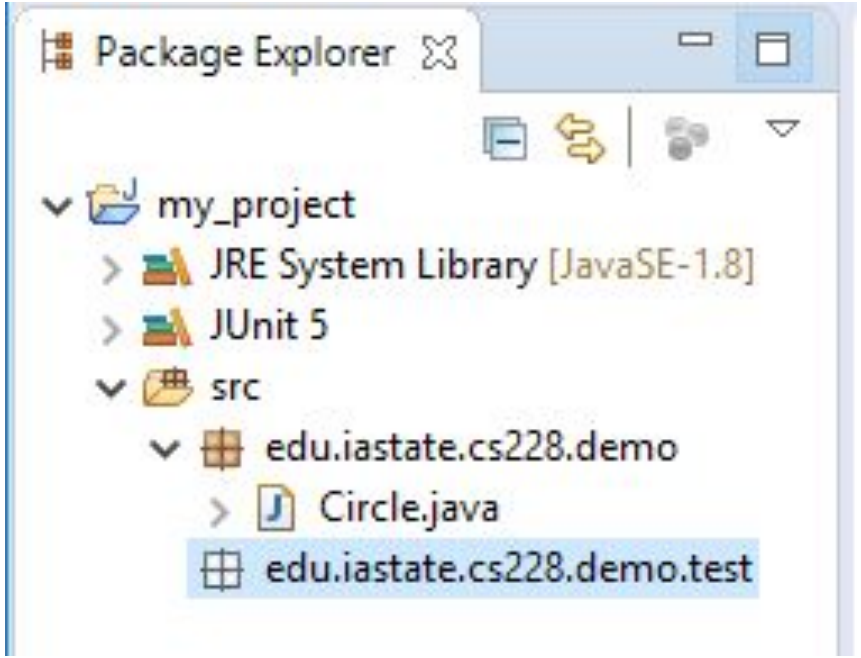
1. Right click on your “src” folder
2. Hover over “New”
3. Select “Package”



Create a test package

4. Name the sub-package: “edu.iastate.cs228.proj__.test”
5. Click Finish

You should see a new testing sub-package for your project. All of your tests will go here.



Create a JUnit 5 test

1. Right click on package
2. Select “New”, then “JUnit Test Case”
3. Enter test class name, then press “Finish”

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class CircleTest {

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

Assertions

If one fails, the test fails

assertEquals, assertNotEquals

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test
    void checkGetMethods() {

        double radius = 5;
        double x = 1, y = 3;
        Circle c = new Circle(radius, x, y);

        //Expected value first, then actual
        assertEquals(radius, c.getRadius());
        assertEquals(x, c.getX());
        assertEquals(y, c.getY());
    }
}
```

assertTrue, assertFalse

```java
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test
    void checkBoolCalculation() {

        double radius = 5;
        double x = 0, y = 0;
        Circle c = new Circle(radius, x, y);

        assertTrue(c.isPointInside(1, 1));
        assertFalse(c.isPointInside(7, 0));

        //Importance of testing: Is a point on
        //the edge of a circle inside it?
        assertTrue(c.isPointInside(5, 0));
    }
}
```

assertThrows

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test
    void checkInvalidInput() {

        //A circle can't have a negative radius
        double radius = -3;
        double x = 0, y = 0;

        assertThrows(IllegalArgumentException.class, () -> {
            Circle c = new Circle(radius, x, y);
        });
    }
}
```

assertThrows (continued…)

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test
    void checkInvalidInput2() {

        double radius = Double.POSITIVE_INFINITY;
        double x = 1, y = 1;

        Throwable exception = assertThrows(IllegalArgumentException.class, () -> {
                    Circle c = new Circle(radius, x, y);
        });

        assertEquals(exception.getMessage(), "Radius can't be infinite");
    }
}
```

assertTimeout

```java
import static java.time.Duration.ofMinutes;
import static java.time.Duration.ofMillis;
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test
    void checkRuntimes() {

        Circle c = new Circle(10, 0, 0);

        //Passes if slow calculations take less than 10 minutes
        assertTimeout(ofMinutes(10), () -> {
            c.slowCalculation();
        });

        //Passes if fast calculations take less than 100 milliseconds
        assertTimeout(ofMillis(100), () -> {
            c.fastCalculation();
        });
    }
}
```

Annotations

Asserting more control over your tests

BeforeAll, AfterAll

```java
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeAll;

import edu.iastate.cs228.demo.Circle;

class CircleTest {
    private Circle c;

    @BeforeAll //Executed before all testing starts
    void setup() {
        c = new Circle(10, 0, 0);
    }

    @AfterAll //Executed after all testing stops
    void tearDown() {
        c = null;
    }
}
```

BeforeEach, AfterEach

```java
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeEach;

import edu.iastate.cs228.demo.Circle;

class CircleTest {
    private Circle c;

    @BeforeEach //Executed before each test
    void setupEach() {
        c = new Circle(10, 0, 0);
    }

    @AfterEach //Executed after each test
    void tearDownEach() {
        c = null;
    }
}
```

JUnit 4 Timeout Alternative

```
import org.junit.Test;

import edu.iastate.cs228.demo.Circle;

class CircleTest {

    @Test(timeout = 100)
    void testSlowCalculation() {

        Circle c = new Circle(10, 0, 0);
        c.slowCalculation();
    }
}
```

Final Notes

- Each test method should test one “thing”
- Write tests for all your projects!!!

Eclipse Debugging

a.k.a, when println fails