

# CS62 Lab4: The Debugger

Basic SWE skills

The screenshot shows an IDE window titled "fractions" with a Java file named "FractionTester.java". The code is as follows:

```
8 public class FractionTester {
10     public static void main(String[] args)
11     {
12         Fraction f = new Fraction("3/4");
13         Fraction g = new Fraction("1/5");
14         Fraction[] myFractions = new Fraction[5];
15         // Add the fractions, store the result
16         Fraction sum = f.add(g); f = Fraction.sum(f, sum);
17     }
18     myFractions[0] = f;
```

The IDE interface includes a "VARIABLES" panel on the left showing local variables: `add()` (Fraction@16), `args` (String[0]@8), `f` (Fraction@10), `g` (Fraction@11), and `myFractions` (Fraction[5]@12). A "WATCH" panel is also visible at the bottom left. The debugger toolbar at the top shows a breakpoint icon on line 16, and the code editor has a yellow highlight on that line.

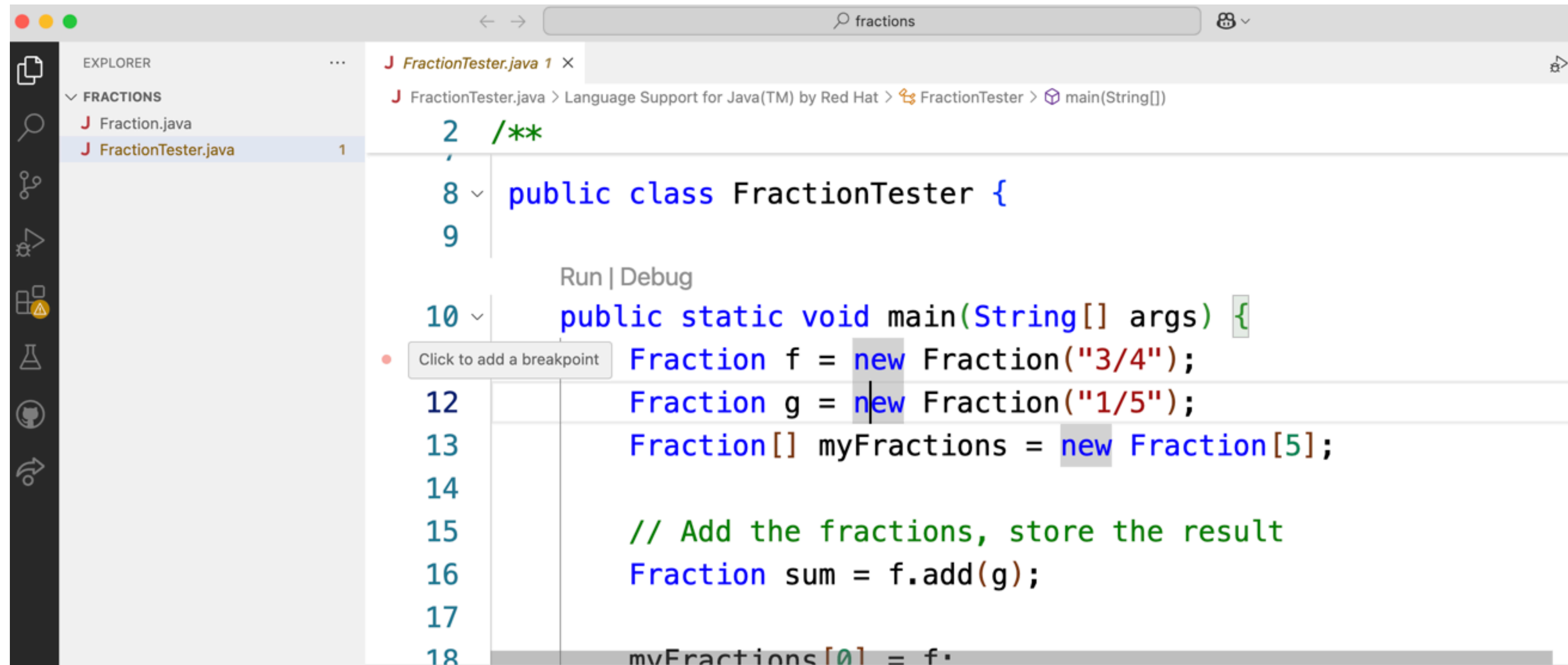
# Lab 4 agenda

- Quiz
- Lab

# Debugger philosophy

- Software is about managing a lot of *state*
- What variables are in my current scope? What are the values of my variables?  
When I run this line of code, where does my code jump? What does my function evaluate to? WHY ISN'T IT WORKING???
- Debuggers let us pause this state (through setting **breakpoints**) and show us all this information at once so we can reason about it
- Once you avoid compiler errors, a debugger will slowly and meticulously let you **step through** your code, line by line, to catch runtime errors.

# Using the Debugger

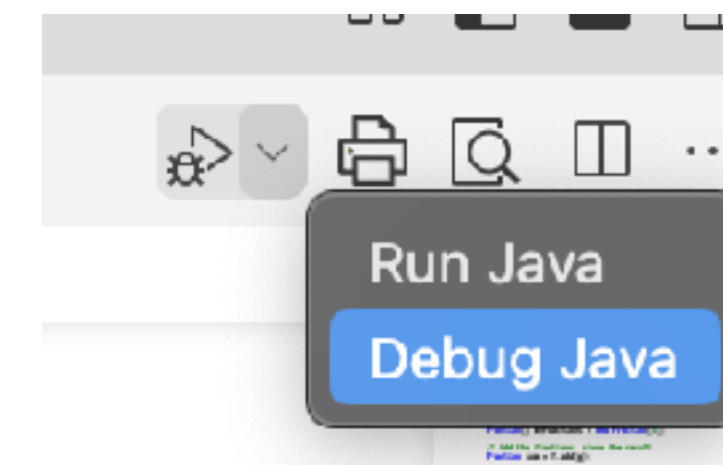


The screenshot shows an IDE window with the following code in FractionTester.java:

```
2  /**
8  public class FractionTester {
9
10 public static void main(String[] args) {
11     Fraction f = new Fraction("3/4");
12     Fraction g = new Fraction("1/5");
13     Fraction[] myFractions = new Fraction[5];
14
15     // Add the fractions, store the result
16     Fraction sum = f.add(g);
17
18     myFractions[0] = f;
```

A red circle on the left margin of line 10 indicates a breakpoint. A tooltip above it says "Click to add a breakpoint". The IDE interface includes an Explorer on the left showing the project structure and a breadcrumb trail at the top: "FractionTester.java > Language Support for Java(TM) by Red Hat > FractionTester > main(String[])".

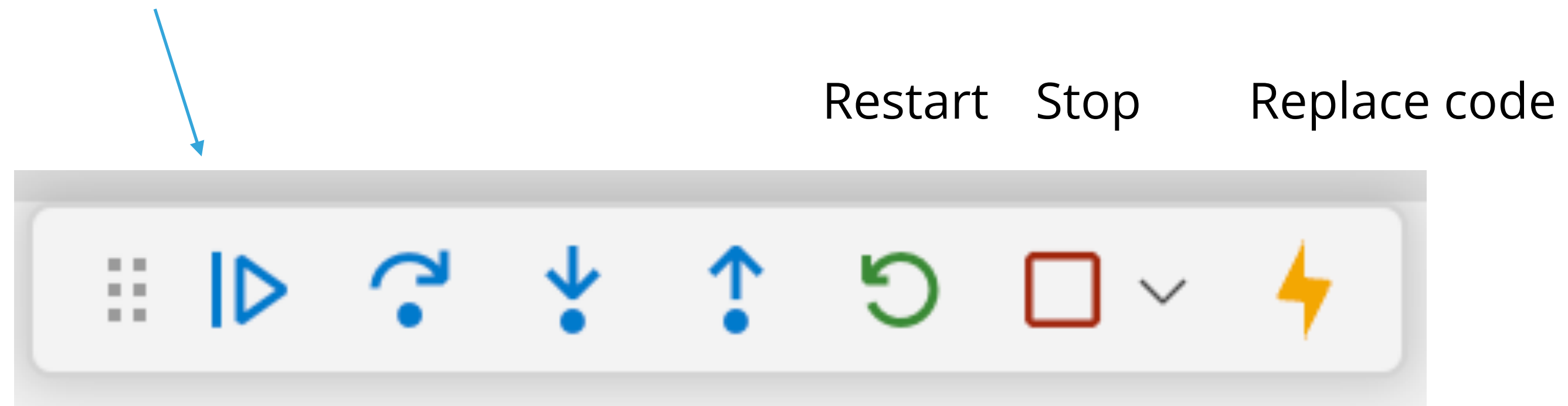
1. Click on the red circle on the side to add/remove a breakpoint



2. "Debug Java" to start debugging

# Using the Debugger

Continue until next breakpoint



Step over (if there is a function call, don't go into the function)

Step into (if there is a function call, do go into the function)

Step out of (if you are in a function, exit the call stack)

## ✓ VARIABLES

### ✓ Local

```
numerator = 19  
denominator = 20  
this = ↻ Fraction@16
```

Sidebar lists all variables in the current scope

# Lab structure

- Use the VSCode debugger to debug 3 matrix classes with different errors
- Checkoff for lab today (no style points, etc) - 3 points for finding the errors, pushing to Github, and submitting to Gradescope