While folks are joining

Get you laptops ready and login to www.crio.do

Start up your workspace.

Crio Sprint: JAVA-2

Session 9

Today’s Session Agenda

- VSCode Debugger
- QCalc Module Introduction
  - Module 6 - Java Debugging
- Introduction to XURL

Novice Debuggers

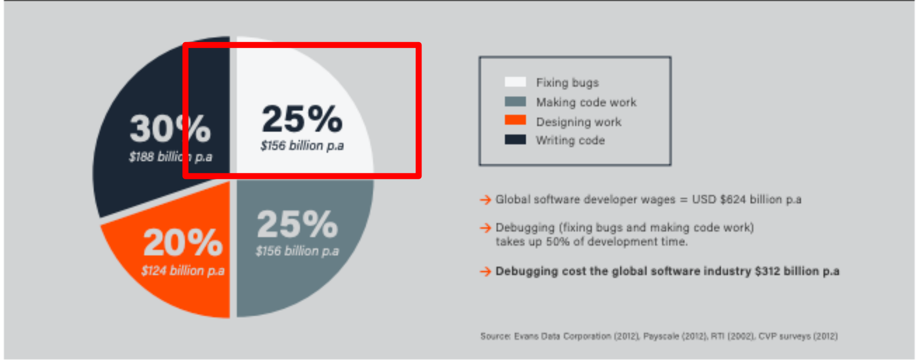
Why become good at debugging?

PROGRAMMING

How to write good software faster (we spend 90% of our time debugging)

If we spend the majority of our programming time and effort on debugging, we should focus our efforts on speeding up our debugging (rather than trying to write code faster).

Impact of debugging on time spent developing code and its cost in terms of wages per annum



→ Global software developer wages = USD $624 billion p.a

→ Debugging (fixing bugs and making code work) takes up 50% of development time.

→ **Debugging cost the global software industry $312 billion p.a**

Source: Evans Data Corporation (2012), Payscale (2012), RTI (2002), CVP surveys (2012)

The study identified that developers spend 50% of their development time fixing bugs or making code work, rather than designing or writing new code. The vast majority of debugging time is spent locating the bug – once it has been found, correcting it is normally relatively simple.

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

— **Brian Kernighan** / *Co-author of the first book on C*

50% of a Developer's Time

According to Rogue Wave Software, 50% of a developers time is spent debugging. .

Create a New Demo Java Project

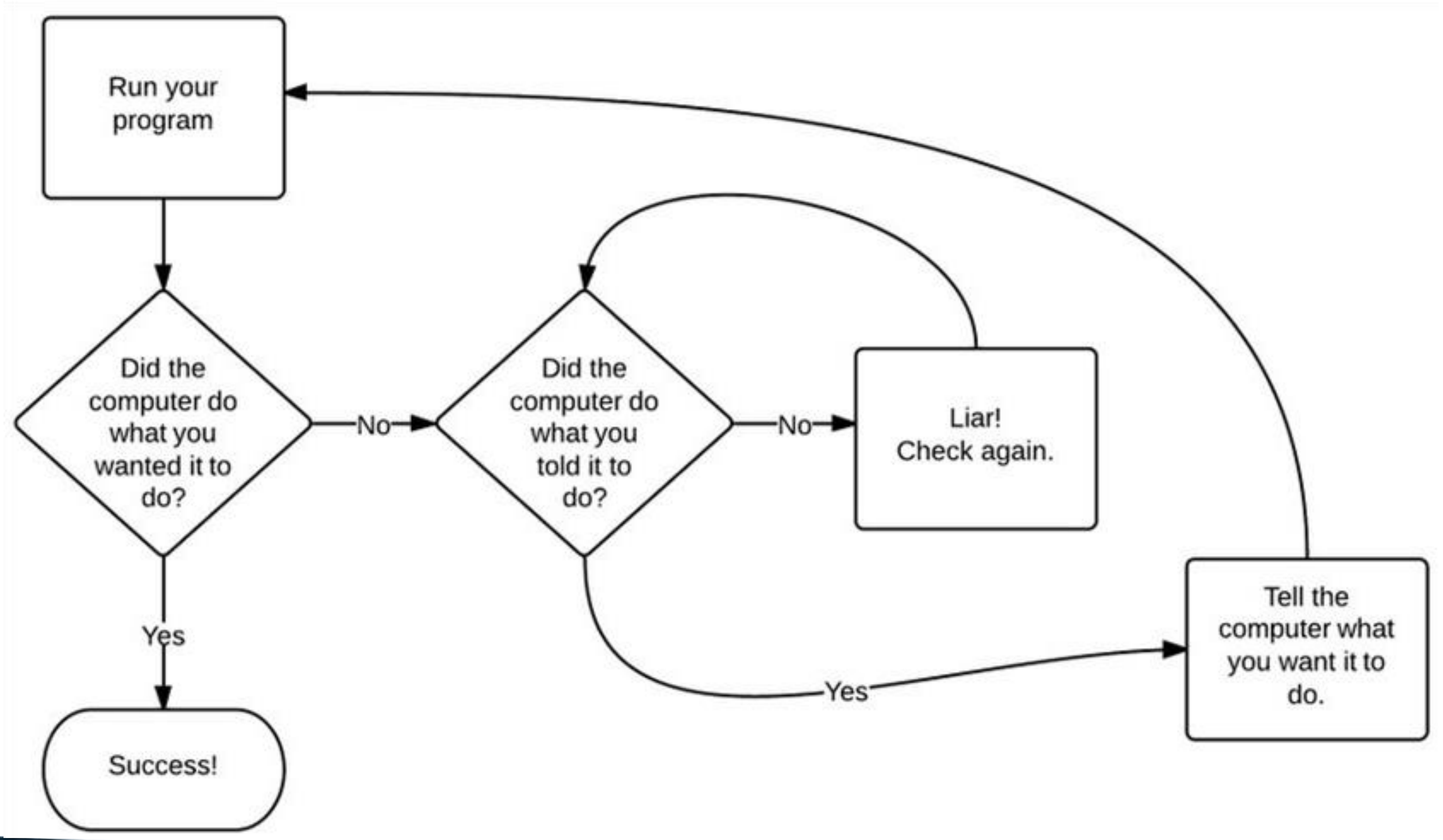
- Create a new “debuggingdemo” Project using the Spring Initializer Extension (cd ~/workspace)
- Create a new file `Main.java` under src > com > crio > debugging folder.
- Create a new file `MainTest.java` under test > com > crio > debuggingdemo folder.
- We will use these files to write code and use the debugger.

Activity #1 - Find the syntax error

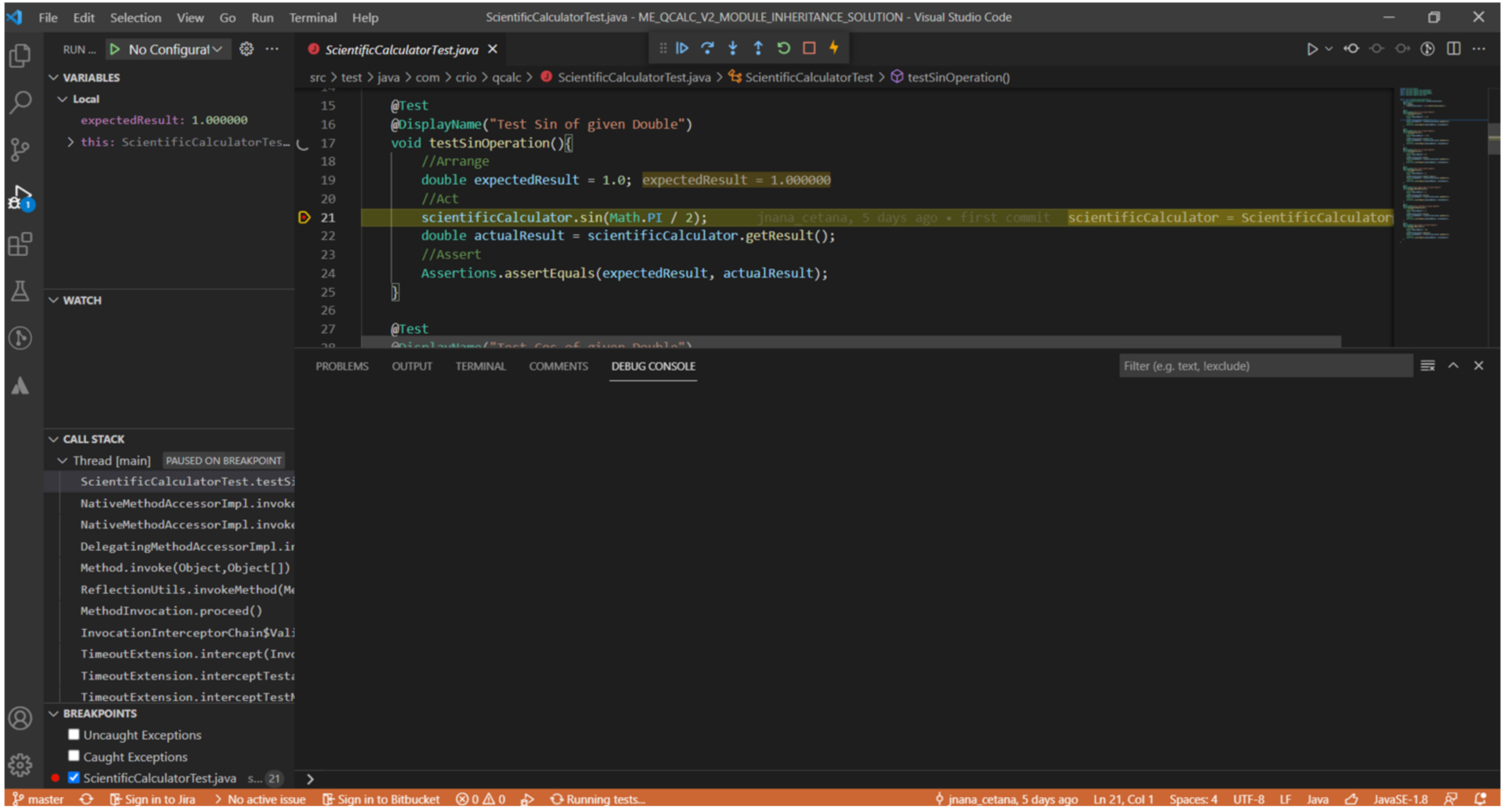
```
class Main {
    public static void main(String args[]) {

        System.out.println("Multiplication Table of 7");
        int a = 7, ans;
        int i;

        for (i = 1, i <= 10; i++) {
            ans = a * i;
            System.out.println(ans + "\n");
        }
    }
}
```

Compile this code

Debugging FlowChart


Run your program
Did the computer do what you wanted it to do?
No
Did the computer do what you told it to do?
No
Liar! Check again.
Yes
Success!
Yes
Tell the computer what you want it to do.

VSCode Debugger Interface


File Edit Selection View Go Run Terminal Help
ScientificCalculatorTest.java - ME_QCALC_V2_MODULE_INHERITANCE_SOLUTION - Visual Studio Code
RUN ... No Configurat
ScientificCalculatorTest.java
src > test > java > com > crio > qcalc > ScientificCalculatorTest.java > ScientificCalculatorTest > testSinOperation()
PROBLEMS OUTPUT TERMINAL COMMENTS DEBUG CONSOLE
Filter (e.g. text, !exclude)
master Sign in to Jira No active issue Sign in to Bitbucket 0 0 Running tests... jnana_cetana, 5 days ago Ln 21, Col 1 Spaces: 4 UTF-8 LF Java JavaSE-1.8

VARIABLES

Local

```
expectedResult: 1.000000
> this: ScientificCalculatorTes…
```

```java
    @Test
    @DisplayName("Test Sin of given Double")
    void testSinOperation(){
        //Arrange
        double expectedResult = 1.0;  expectedResult = 1.000000
        //Act
        scientificCalculator.sin(Math.PI / 2);      jnana_cetana, 5 days ago • first commit   scientificCalculator = ScientificCalculator
        double actualResult = scientificCalculator.getResult();
        //Assert
        Assertions.assertEquals(expectedResult, actualResult);
    }

    @Test
    @DisplayName("Test Cos of given Double")
```

WATCH

CALL STACK

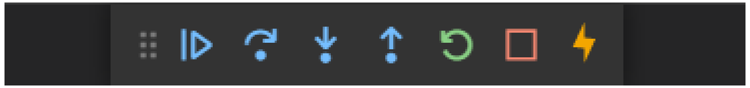
Thread [main] PAUSED ON BREAKPOINT

```
ScientificCalculatorTest.testSi
NativeMethodAccessorImpl.invoke
NativeMethodAccessorImpl.invoke
DelegatingMethodAccessorImpl.ir
Method.invoke(Object,Object[])
ReflectionUtils.invokeMethod(Me
MethodInvocation.proceed()
InvocationInterceptorChain$Vali
TimeoutExtension.intercept(Invc
TimeoutExtension.interceptTesta
TimeoutExtension.interceptTestM
```

BREAKPOINTS

- [ ] Uncaught Exceptions
- [ ] Caught Exceptions
- [x] ScientificCalculatorTest.java s... 21

Debugging Toolbar



Continue (F5) - This command resumes the program and continues to run it normally, without pausing unless it hits another breakpoint.

Step Over (F11) - The Step Over command takes a single step. It executes the currently highlighted line, and then pauses again.

Step Into (F10) - Step Into goes *into* that function and pauses on the first line inside.

Step Out (Shift + F11) - Step Out command executes all the code in the current function, and then pauses at the next statement (if there is one).

Stop (Shift + F5) - Stop Debugging

Restart (Ctrl + Shift + F5) - Restart Debugging

Activity #2 - Reverse a Number

```
public class FindTheError {
    public static void main(String[] args)
    {
        int num = 789;
        int reversednum = 0;
        int remainder;
        while (num != 0) {
            remainder = num / 10;
            reversednum = reversednum * 10 + remainder;
            num /= 10;
        }
        System.out.println("Reversed number is "+ reversednum);
    }
}
```

Execute and Debug this code

Debugging Code

- cd ~/workspace
- git clone https://gitlab.crio.do/root/shapeofshapes.git
- Run the ShapesApplication.java present in com -> crio -> shapes directory.
  - Fix any errors so that it runs successfully
- Run all the tests from ShapesApplicationTest.java

Activity #3 - Debug this test case

```
@Test
@DisplayName("Test Area of a Triangle")
void testPerimeterTriangle(){
   double expectedResult = 32.0;
   Triangle t = new Triangle(5,7);
   double actualResult = t.calculateArea();
   Assertions.assertEquals(expectedResult, actualResult);
}
```

Activity #4 - Debug Circumference of a Circle

```
@Test
@DisplayName("Test Circumference of a Circle")
void testCircumferenceCircle(){
   double expectedResult = 31.41592653589793;
   Circle c = new Circle();
   double actualResult =c.calculateCircumference();
   Assertions.assertEquals(expectedResult, actualResult);
}
```

Activity #5 - Debug for Exception

```
@Test
void testRectangleException() {
    Assertions.assertThrows(ArithmeticException.class,new Executable(){
        @Override
        public void execute() throws Throwable{
            new Rectangle(0,-3);
        }
    });
}
```

Activity #6

Filtering even numbers (Bug Fixes)

Activity #7

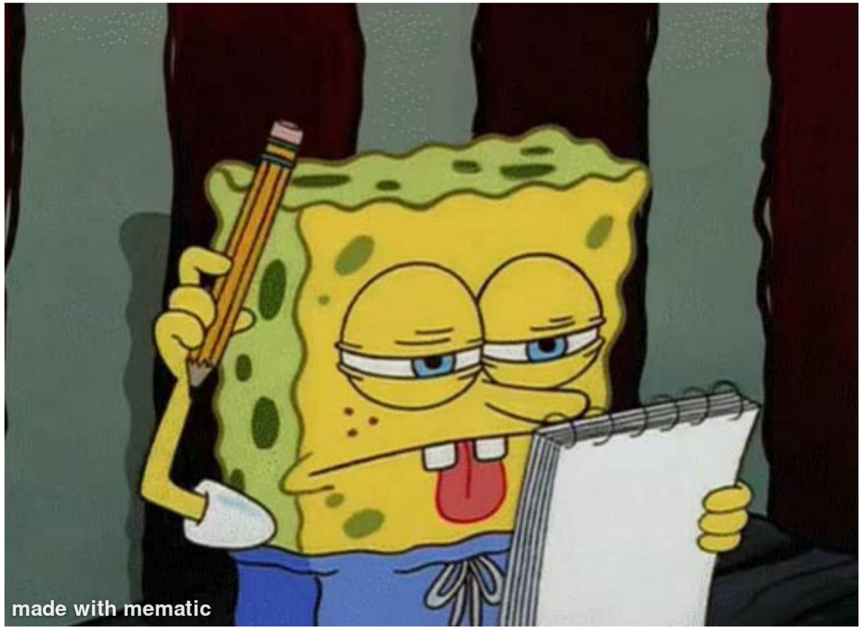
Training on Squash the bugs (Bug Fixes)

Key takeaways

- A **debugger** is a software that allows a developer to **step through the execution of code**
    - Code gets executed line by line and a debugger lets you stop and monitor the state of the program during execution. We can trace the program to debug where things are going wrong, when there are failures.
- **Breakpoints -** Lines where the JVM has to suspend/pause execution.
    - We can have any number of breakpoints across files.
- Use **“Debug Test”** or **“Debug”** to run the program in the debugging mode.
- **Variables Window -** where we can monitor all the variables values in the current method/class that we're in.
- **Watch Window -** useful to evaluate an expression, especially when you have function calls that might go multiple levels. Avoid stepping through each level of the function call being made.
- **Call Stack Window -** shows the current function and the overall function call stack which led to this function.

Random Memes

debugging your own code be like:
debug.log("yes this is a debug");



Programmers everyday..

QCalc - Module 6: Java Debugging

- In this module:
  - Fix compilation errors caused probably due to syntax / import issues.
  - Correct logical issues in the code.
  - Fix the behaviour of a method when data is invalid.
  - Write Unit Tests for edge cases initially not thought of.

Introduction to Buildout: XURL

In this Buildout:

- Build a simple application to generate short URLs for longer URLs and retrieve them upon request.
- Test it.
- Interface definition is given, you have to implement it.

Take home exercises for the session

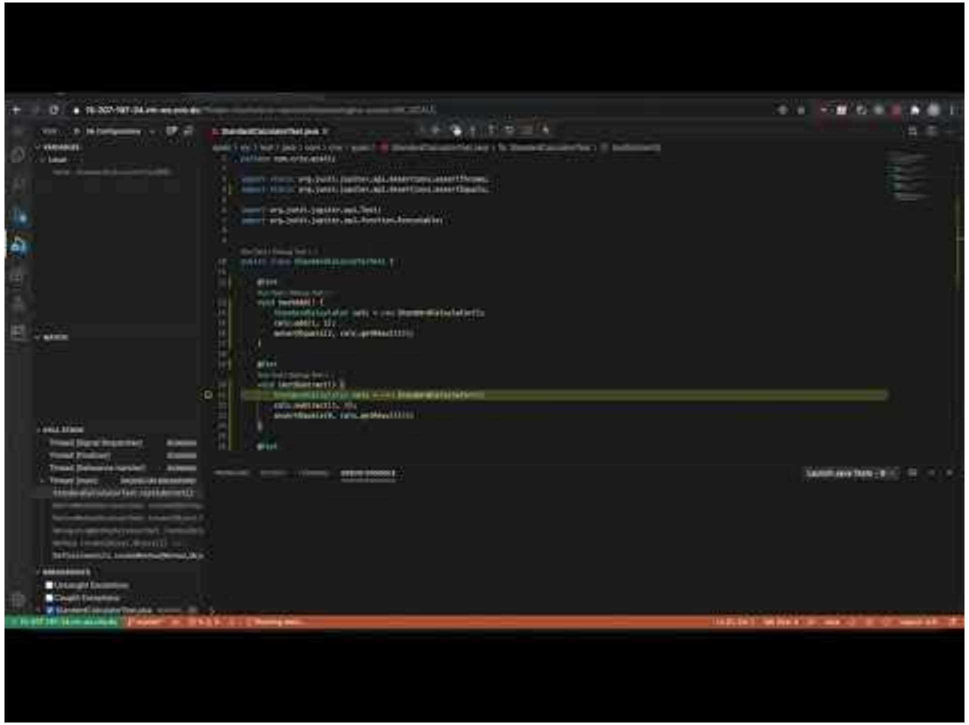
- You have to complete the below modules of QCalc Micro-Experience:
  - Module 6 - Java Debugging
- You have to complete the Buildout
  - Buildout - Short URL (XUrl)

Feedback

Thank you for joining in today.

We’d love to hear your thoughts and feedback - Feedback for JAVA-2 Session

Video Reference



StandardCalculatorTest.java

```
package [illegible];

import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.function.Executable;


public class StandardCalculatorTest {

    @Test
    void testAdd() {
        StandardCalculator calc = new StandardCalculator();
        calc.add(1, 1);
        assertEquals(2, calc.getResult());
    }

    @Test
    void testSubtract() {
        StandardCalculator calc = new StandardCalculator();
        calc.subtract(1, 1);
        assertEquals(0, calc.getResult());
    }

    @Test
```

Thank you