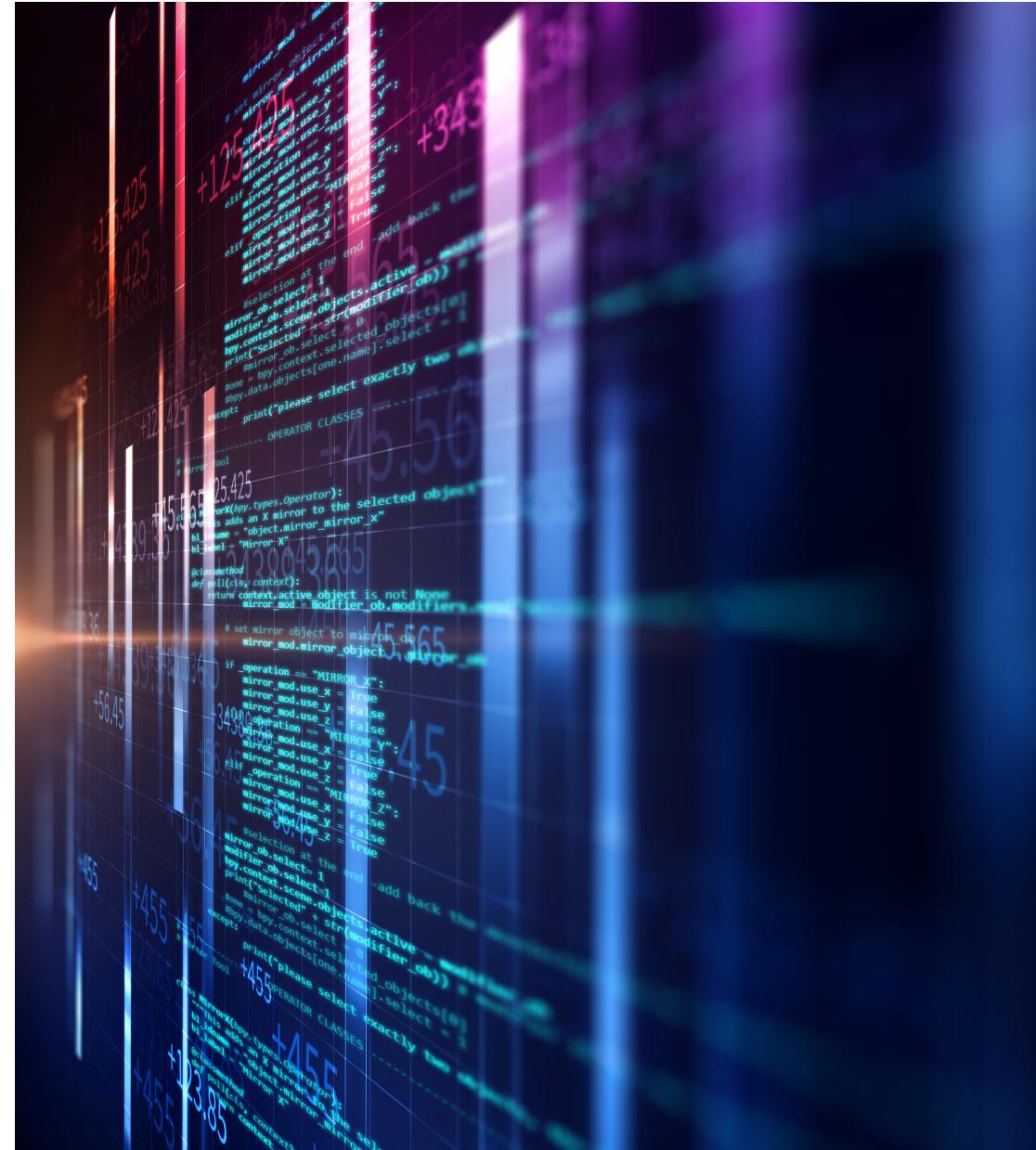


Debugging

Brandon Krakowsky





Debugging in Eclipse



What is Debugging?

- Debugging allows you to run a program interactively while watching the source code and the variables during the execution
- A breakpoint in the source code specifies where the execution of the program should stop during debugging
- Once a program is stopped, you can investigate variables (and even change their content)



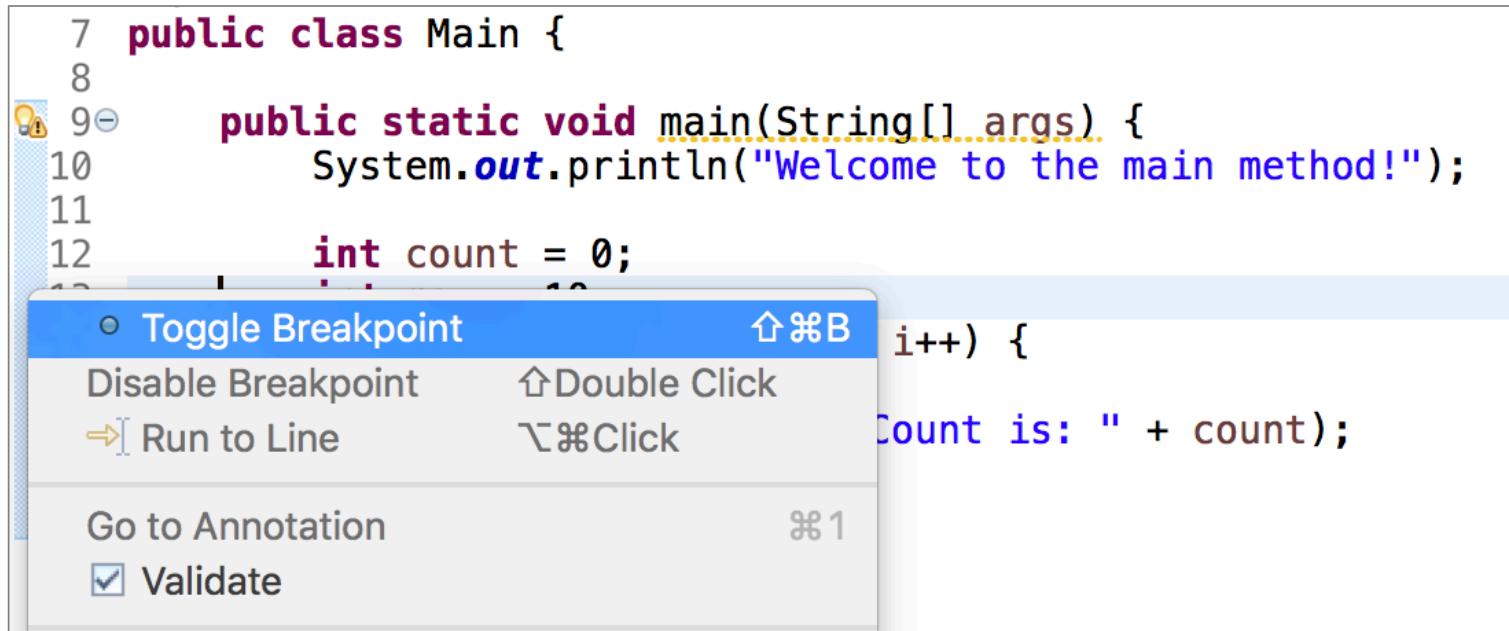
Debugging Support in Eclipse

- Eclipse allows you to start a Java program in Debug mode
- Eclipse provides a [Debug perspective](#), which gives you a pre-configured set of views
- Eclipse allows you to control the execution flow via [debug commands](#)



Setting Breakpoints

- To define a breakpoint in your source code, right-click in the left margin in the Java editor and select “Toggle Breakpoint”



- You can also double-click on this position in the margin to toggle a breakpoint



Setting Breakpoints

- For example, here we've set a breakpoint on the line:

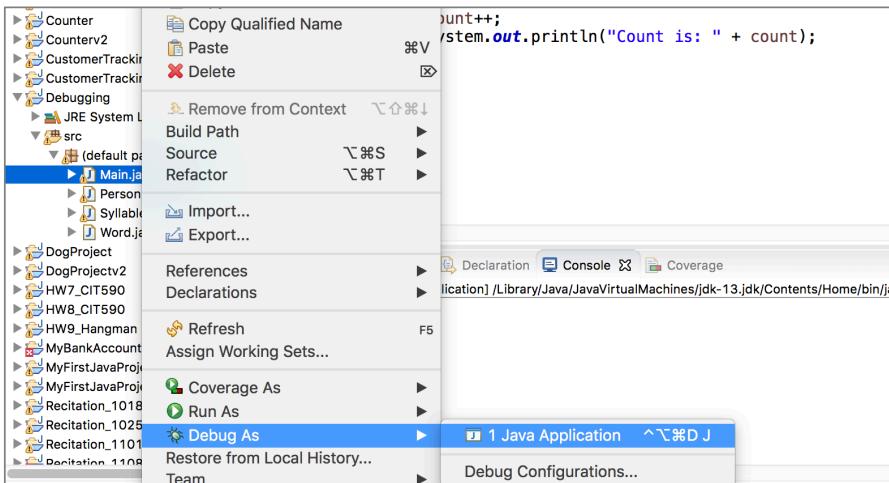
```
int max = 10;
```

```
7 public class Main {  
8  
9     public static void main(String[] args) {  
10         System.out.println("Welcome to the main method!");  
11  
12         int count = 0;  
13         int max = 10; // Line 13 is highlighted with a red box  
14         for (int i = 0; i < max; i++) {  
15             count++;  
16             System.out.println("Count is: " + count);  
17         }  
18     }  
19  
20 }  
21
```



Starting the Debugger

- To debug your application, do the following:
 - Select your program (Java file) in the Package Explorer
 - Go to “Debug As” → “Java Application”



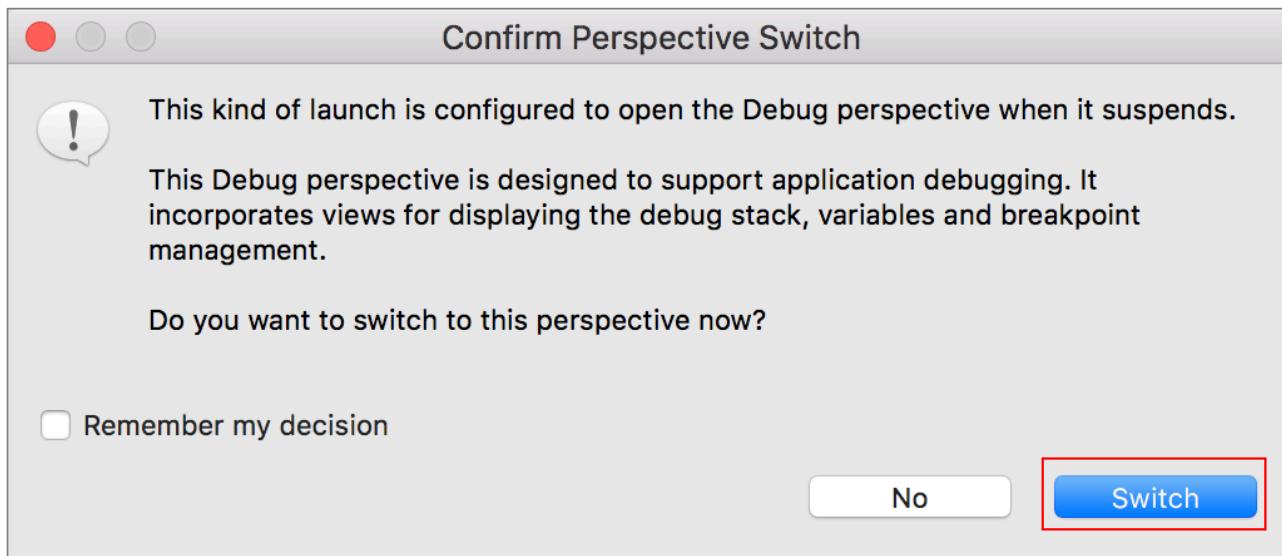
- You can also use the Debug button in the Eclipse toolbar





Starting the Debugger

- Eclipse might ask you if you want to open the Debug perspective once a stop point is reached
 - Click “Switch”





Controlling the Program Execution

- Eclipse provides buttons in the toolbar for controlling the execution of the program you are debugging

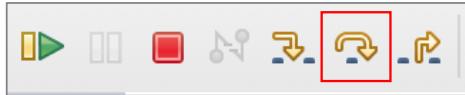


- The Step Into (F5) button executes the currently selected line and goes to the next line in your program
 - If the selected line is a method call, the debugger *steps into* the associated code



Controlling the Program Execution

- Eclipse provides buttons in the toolbar for controlling the execution of the program you are debugging

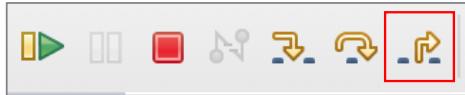


- The Step Over (F6) button *steps over* the call
 - This means it executes a method without *stepping into* it in the debugger



Controlling the Program Execution

- Eclipse provides buttons in the toolbar for controlling the execution of the program you are debugging

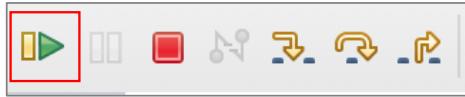


- The Step Return (F7) button *steps out* to the caller of the currently executed method
 - This finishes the execution of the current method and returns to the caller of this method



Controlling the Program Execution

- Eclipse provides buttons in the toolbar for controlling the execution of the program you are debugging

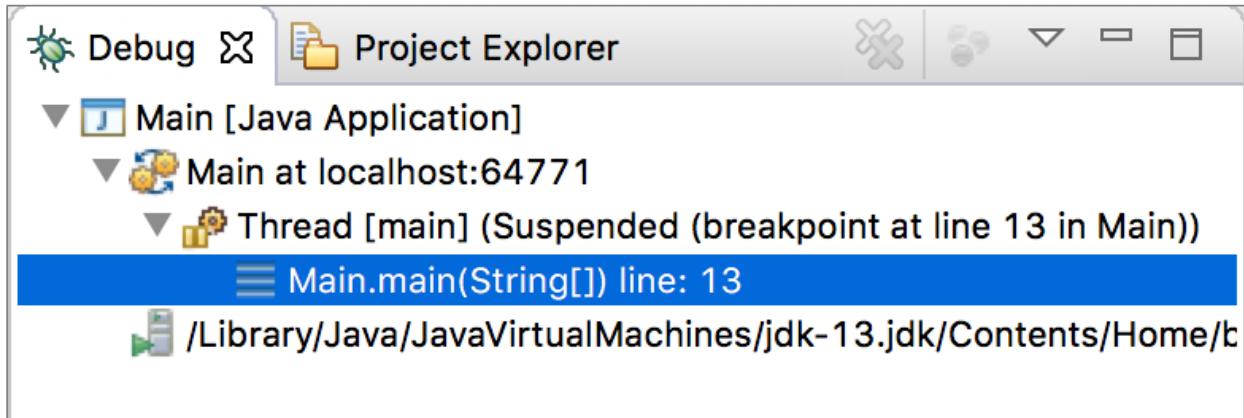


- The [Resume \(F8\)](#) button tells the Eclipse debugger to resume the execution of the program code until it reaches the next breakpoint



Controlling the Program Execution

- The Debug view shows the parts of the program which are currently executed and how they relate to each other





Evaluating Variables in the Debugger

- The **Variables** view displays variables with their current values

Name	Value
no method return value	
args	String[0] (id=18)
count	4
max	10
i	3

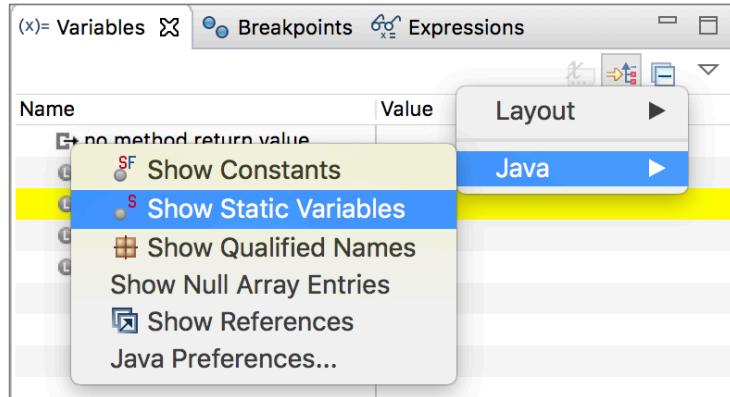


Evaluating Variables in the Debugger

- The **Variables** view displays variables with their current values

Name	Value
+ no method return value	
args	String[0] (id=18)
count	4
max	10
i	3

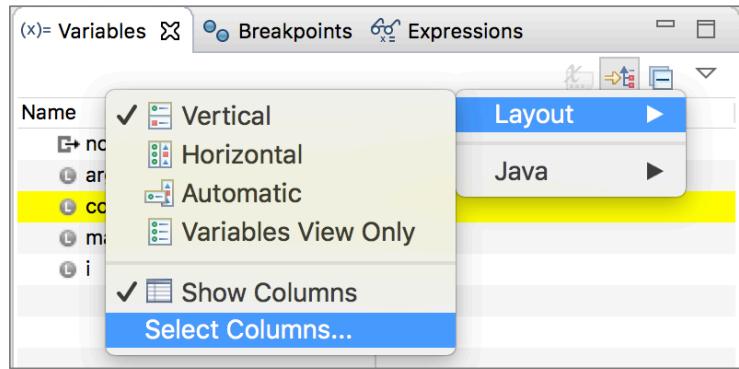
- Use the drop-down menu in the top-right of the **Variables** view to display static variables





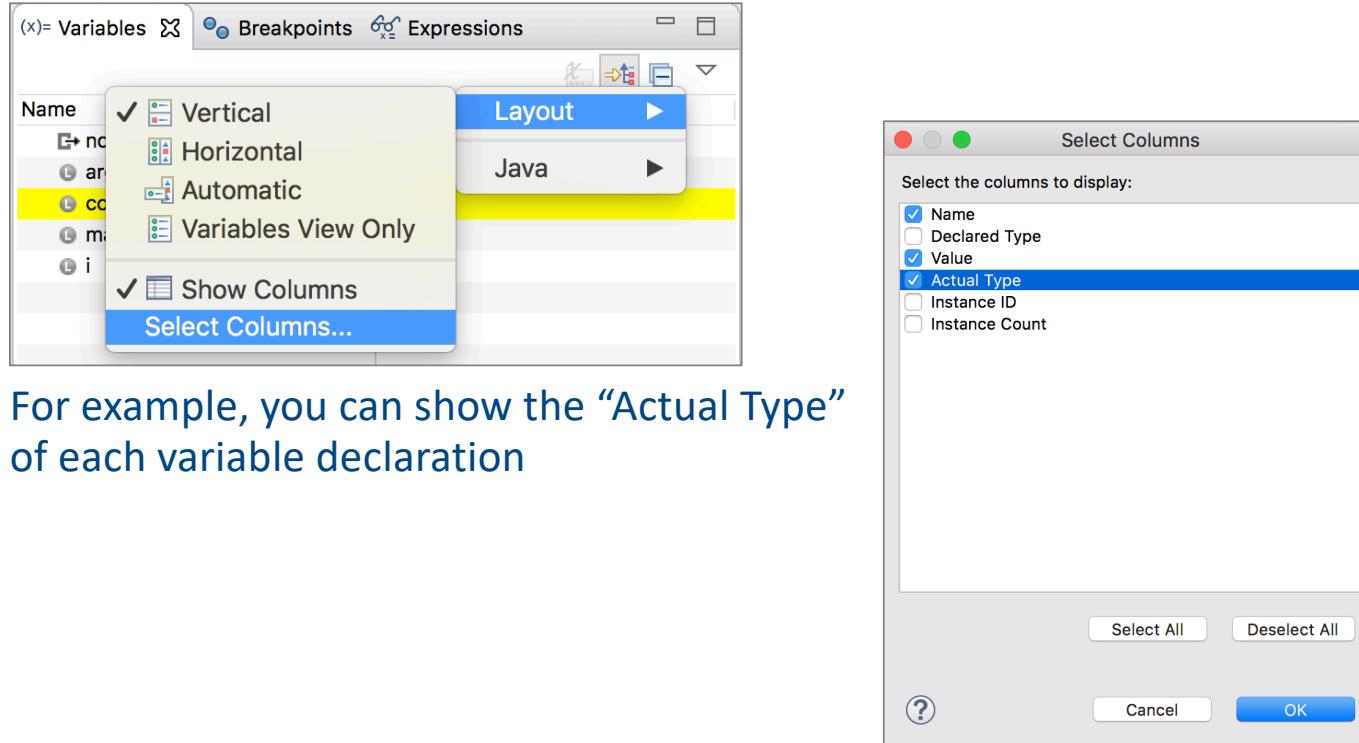
Evaluating Variables in the Debugger

- You can customize the displayed columns



Evaluating Variables in the Debugger

- You can customize the displayed columns

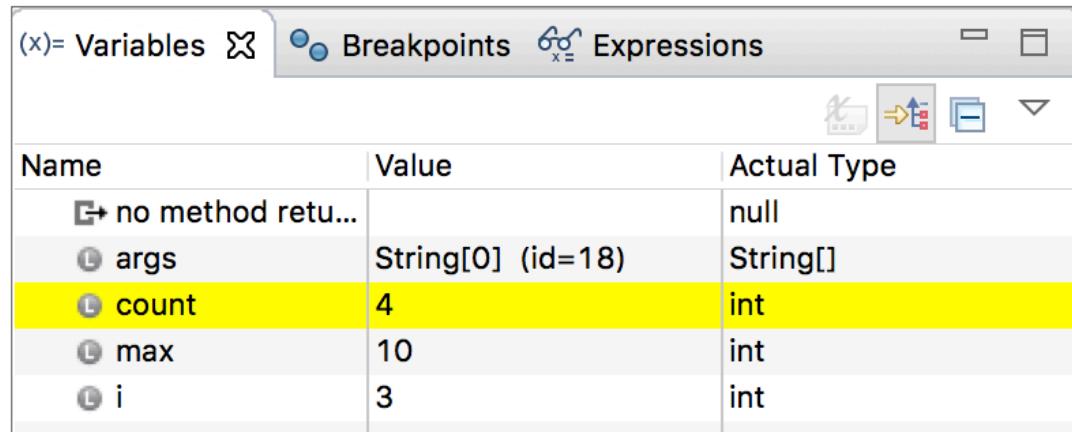


- For example, you can show the “Actual Type” of each variable declaration



Evaluating Variables in the Debugger

- The Variables view now displays the “Actual Type” for each variable

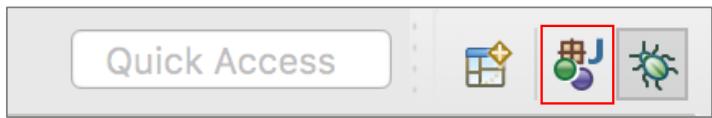


Name	Value	Actual Type
↳ no method retu...		null
↳ args	String[0] (id=18)	String[]
↳ count	4	int
↳ max	10	int
↳ i	3	int



Returning to Java Perspective

- To go back to the Java perspective (and exit the Debug perspective), click the Java button on the right of the Eclipse toolbar





Debugging Examples - People

Person Class



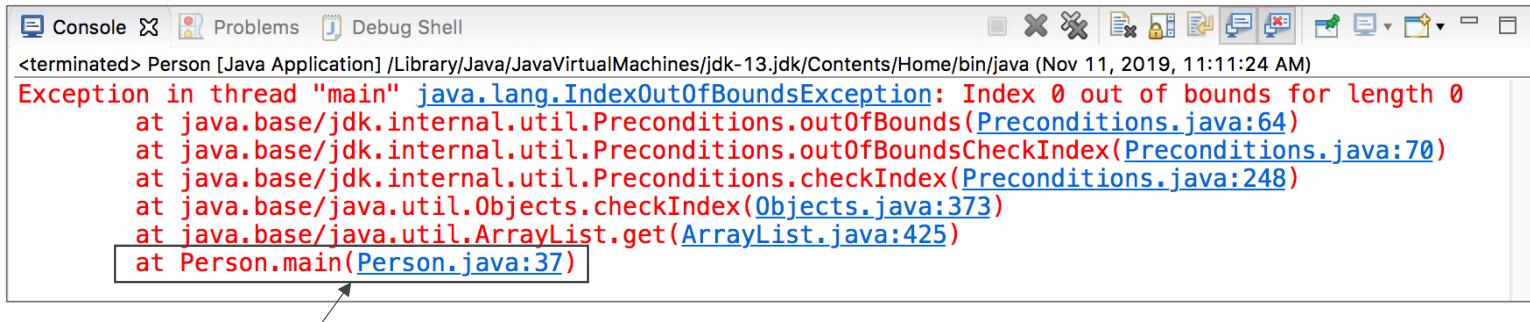
```
Person.java
7  /*
8  public class Person {
9
10     /**
11      * Name of person.
12      */
13     private String name;
14
15     /**
16      * Set name of person with given name.
17      * @param name for person
18      */
19     public void setName(String name) {
20         this.name = name;
21     }
22
23     /**
24      * Get name of person.
25      * @return name
26      */
27     public String getName() {
28         return this.name;
29     }
}
```

Person Class

```
31  public static void main(String[] args) {
32
33     //create list for people
34     ArrayList<Person> people = new ArrayList<Person>();
35
36     //Get first person's name
37     String firstName = people.get(0).getName();
38
39     //Get length of first person's name
40     int firstNameLength = firstName.length();
41
42     //Print length and name of first person
43     System.out.println(firstName + " has a length of: " + firstNameLength);
44
45 }
```

Person Class

- Start debugging using the Java stack trace displayed in the console
 - It contains hyperlinks for jumping to particular source code locations



The screenshot shows a Java application console window. The title bar includes tabs for 'Console', 'Problems', and 'Debug Shell'. The main area displays a stack trace for an `IndexOutOfBoundsException`. The text is color-coded, with red highlighting the exception type and file names, and blue highlighting line numbers. A mouse cursor is visible, pointing at the line 'at Person.main(Person.java:37)'. This line is also highlighted with a white rectangle. An arrow points from the text 'Execution of code leading to an Exception starts here' to this highlighted line.

```
<terminated> Person [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (Nov 11, 2019, 11:11:24 AM)
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:64)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:70)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:248)
    at java.base/java.util.Objects.checkIndex(Objects.java:373)
    at java.base/java.util.ArrayList.get(ArrayList.java:425)
    at Person.main(Person.java:37)
```

Execution of code leading to an Exception starts here

Person Class

- Add a breakpoint and debug in the Debug perspective ...



```
31  public static void main(String[] args) {
32
33     //create list for people
34     ArrayList<Person> people = new ArrayList<Person>();
35
36     //Get first person's name
37     String firstName = people.get(0).getName(); ●
38
39     //Get length of first person's name
40     int firstNameLength = firstName.length();
41
42     //Print length and name of first person
43     System.out.println(firstName + " has a length of: " + firstNameLength);
44
45 }
```

Person Class

- ... then fix the bug by creating and adding at least one person

```
31  public static void main(String[] args) {  
32  
33      //create list for people  
34      ArrayList<Person> people = new ArrayList<Person>();  
35  
36      //create and add person  
37      Person person = new Person();  
38      people.add(person);  
39  
40      //Get first person's name  
41      String firstName = people.get(0).getName();  
42  
43      //Get length of first person's name  
44      int firstNameLength = firstName.length();  
45  
46      //Print length and name of first person  
47      System.out.println(firstName + " has a length of: " + firstNameLength);  
48  
49  }  
50
```

Person Class

- Start debugging again using the Java stack trace displayed in the console

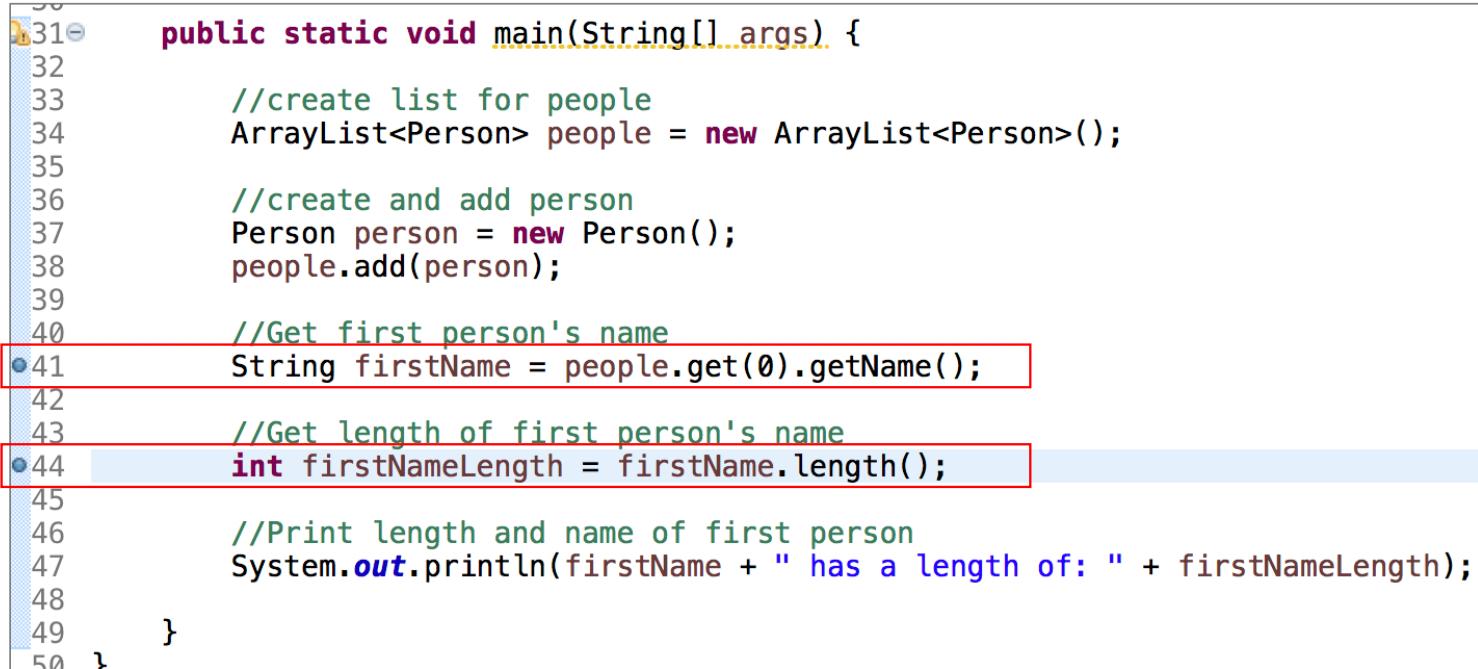


```
Console Problems Debug Shell
<terminated> Person [Java Application] /Library/Java/JavaVirtualMachines/jdk-13.jdk/Contents/Home/bin/java (Nov 11, 2019, 11:13:25 AM)
Exception in thread "main" java.lang.NullPointerException
    at Person.main(Person.java:44)
```

Execution of code leading to an Exception starts here

Person Class

- Add a 2nd breakpoint and debug in the Debug perspective ...



```
31  public static void main(String[] args) {
32
33     //create list for people
34     ArrayList<Person> people = new ArrayList<Person>();
35
36     //create and add person
37     Person person = new Person();
38     people.add(person);
39
40     //Get first person's name
41     String firstName = people.get(0).getName();
42
43     //Get length of first person's name
44     int firstNameLength = firstName.length();
45
46     //Print length and name of first person
47     System.out.println(firstName + " has a length of: " + firstNameLength);
48
49 }
50 }
```

Person Class

- ... then fix the bug by setting the person's name

```
31  public static void main(String[] args) {  
32  
33      //create list for people  
34      ArrayList<Person> people = new ArrayList<Person>();  
35  
36      //create and add person  
37      Person person = new Person();  
38      people.add(person);  
39  
40      //set person's name  
41      person.setName("Brandon");  
42  
43      //Get first person's name  
44      String firstName = people.get(0).getName();  
45  
46      //Get length of first person's name  
47      int firstNameLength = firstName.length();  
48  
49      //Print length and name of first person  
50      System.out.println(firstName + " has a length of: " + firstNameLength);  
51  
52  }
```

Person Class



```
Console X
<terminated> ExceptionExample [Java Application] /Library/Java/JavaVirtualMachines/jdk
Brandon has a length of: 7
```



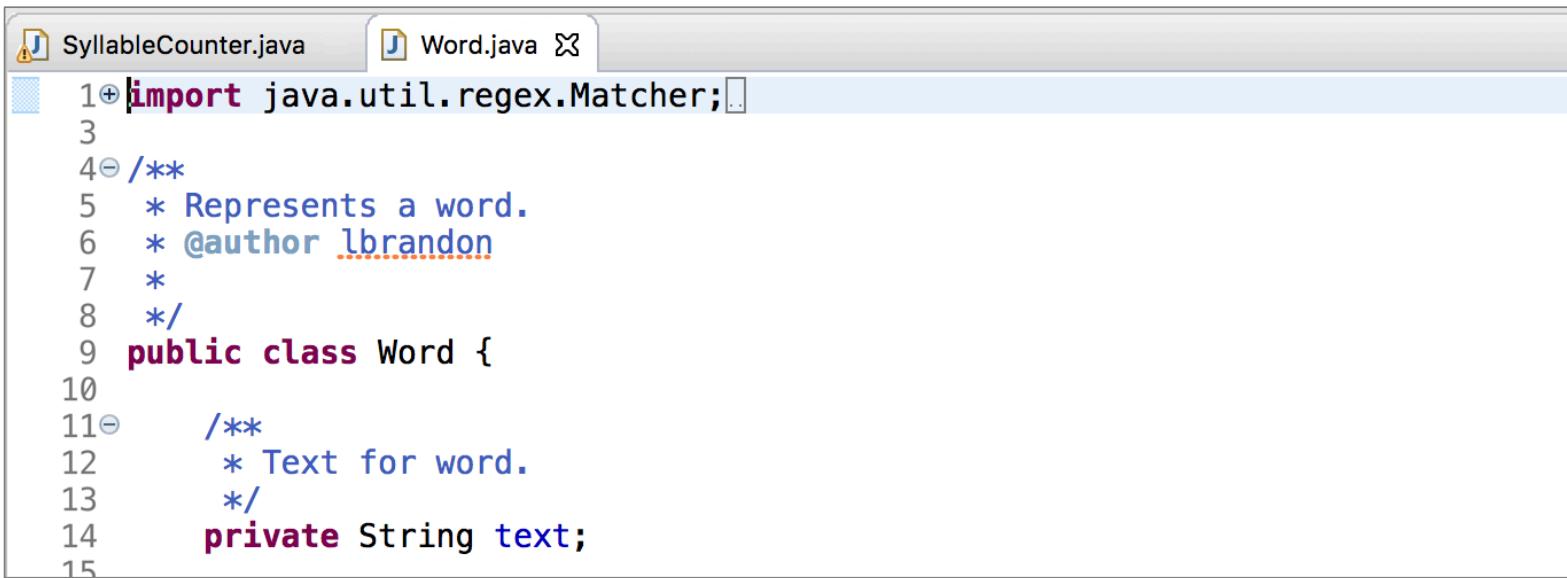
Debugging Examples – Syllable Counter

SyllableCounter Class

```
8 public class SyllableCounter {  
9  
10    public static void main(String[] args) {  
11        System.out.println("Enter a sentence: ");  
12  
13        //get user input of sentence  
14        Scanner in = new Scanner(System.in);  
15  
16        String input;  
17        Word w;  
18        int syllables;  
19  
20        //while there is a word to scan  
21        while(in.hasNext()) {  
22  
23            //get next token (word)  
24            input = in.next();  
25  
26            //create instance of Word class  
27            w = new Word(input);  
28  
29            //get number of syllables  
30            syllables = w.countSyllables();  
31  
32            //print out word and count of syllables  
33            System.out.println("Syllables in " + w.getText() + ": " + syllables);  
34  
35            //break out of while loop at period  
36            if (input.endsWith(".")) break;  
37        }  
38  
39        in.close();  
40  
41    }  
42  
43}  
44
```



Word Class



```
1+import java.util.regex.Matcher;[]
3
4+/*
5 * Represents a word.
6 * @author lbrandon
7 *
8 */
9public class Word {
10
11+    /**
12     * Text for word.
13     */
14    private String text;
15}
```

Word Class

```
15
16⊕  /**
17   * Creates a word with given String.
18   * Trims non-letters from beginning and end of String.
19   * @param s for word
20   */
21⊕  public Word(String s) {
22
23     //trim beginning of word
24     int i = 0;
25     while (i < s.length() && !Character.isLetter(s.charAt(i))) {
26       i++;
27     }
28
29     //trim end of word
30     int j = s.length() - 1;
31     while (j > i && !Character.isLetter(s.charAt(j))) {
32       j--;
33     }
34
35     //gets a substring of given s based on i and j
36     //stores it as text for word
37     //e.g. 123hello321 will be stored as hello
38     this.text = s.substring(i, j);
39
40
```

Word Class

```
40
41 ⊕  /**
42   * Get word text.
43   * @return text
44   */
45 ⊕  public String getText() {
46   return this.text;
47 }
48
```



Word Class

```
49④    /**
50     * Counts syllables in word.
51     * @return syllable count
52     */
53④    public int countSyllables() {
54
55        //set initial count of syllables
56        int count = 0;
57
58        //get index of last character in word
59        int end = this.text.length() - 1;
60
61        //if word has 0 chars, consider 0 syllables
62        if (end < 0) {
63            return 0;
64        }
65
66        //An 'e' at the end of the word doesn't count as a vowel
67        //So decrement end index
68        char ch = this.text.charAt(end);
69        if (ch == 'e' || ch == 'E') {
70            end--;
71        }
72    }
```



Word Class

```
72
73     //set flag for being inside a syllable
74     boolean insideSyllable = false;
75
76     //iterate over characters in word and look for vowels
77     for (int i = 0; i <= end; i++) {
78
79         //get each character
80         ch = this.text.charAt(i);
81
82         //determine if character is a vowel
83         //create a "character class" using regular expression,
84         //containing every vowel we're looking for (lower and upper) in word
85         String vowelRegex = "[aeiouAEIOU]";
86
87         //create pattern to match with character
88         Pattern p = Pattern.compile(vowelRegex);
89
90         //find matches in char (casted to a String)
91         Matcher m = p.matcher(ch + "");
92
93         //if it is a vowel, enter syllable
94         if (m.matches()) {
95             if (!insideSyllable) {
96                 count++;
97                 insideSyllable = true;
98             }
99         }
100    }
```

Word Class

```
102
103     //every word has at least one syllable
104     if (count == 0) {
105         count = 1;
106     }
107
108     //return the count
109     return count;
110 }
111
```



SyllableCounter Class

- Run the program and type a sentence ending with a period

```
Enter a sentence ending in a period.  
this class is amazing, although the homework can be difficult, everybody loves it.  
Syllables in thi: 1  
Syllables in clas: 1  
Syllables in i: 1  
Syllables in amazin: 1  
Syllables in althoug: 1  
Syllables in th: 1  
Syllables in homewor: 1  
Syllables in ca: 1  
Syllables in b: 1  
Syllables in difficul: 1  
Syllables in everybod: 1  
Syllables in love: 1  
Syllables in i: 1
```

- Note the output:
 - Every word lost its last character

SyllableCounter Class

- Add a **breakpoint** and debug in the **Debug perspective** ...
 - Step Into the *getText* method to see what's happening

```
10 public static void main(String[] args) {  
11     System.out.println("Enter a sentence: ");  
12  
13     //get user input of sentence  
14     Scanner in = new Scanner(System.in);  
15  
16     String input;  
17     Word w;  
18     int syllables;  
19  
20     //while there is a word to scan  
21     while(in.hasNext()) {  
22  
23         //get next token (word)  
24         input = in.next();  
25  
26         //create instance of Word class  
27         w = new Word(input);  
28  
29         //get number of syllables  
30         syllables = w.countSyllables();  
31  
32         //print out word and count of syllables  
33         System.out.println("Syllables in " + w.getText() + ": " + syllables);  
34     }  
35     //break out of while loop at period  
36     if (input.endsWith(".")) break;  
37     }  
38  
39     in.close();  
40  
41 }  
42 }
```

- Note: By default, Step Into will enter the first method it finds on the line

- Step Into will first enter the *println* method. You'll have to Step Return out of the *println* method, then Step Into the *getText* method.



SyllableCounter Class

- Now, add a new breakpoint where the instance of Word is created ...

- Step Into the *Word* constructor

```
10  public static void main(String[] args) {  
11      System.out.println("Enter a sentence: ");  
12  
13      //get user input of sentence  
14      Scanner in = new Scanner(System.in);  
15  
16      String input;  
17      Word w;  
18      int syllables;  
19  
20      //while there is a word to scan  
21      while(in.hasNext()) {  
22  
23          //get next token (word)  
24          input = in.next();  
25  
26          //create instance of Word class  
27          w = new Word(input);  
28  
29          //get number of syllables  
30          syllables = w.countSyllables();  
31  
32          //print out word and count of syllables  
33          System.out.println("Syllables in " + w.getText() + ": " + syllables);  
34  
35          //break out of while loop at period  
36          if (input.endsWith(".")) break;  
37      }  
38  
39      in.close();  
40  }  
41  
42 }
```

- Step Into will first enter a *loadClass* method in the *ClassLoader*. You'll have to Step Return out of the *loadClass* method, then Step Into the *Word* constructor.

- Note: By default, Step Into will enter the first method it finds on the line



Word Class

- Step Over each line in the *Word* class constructor



Word Class

- ... then fix the bug by correctly setting i and j

```
16⑩  /**
17   * Creates a word with given String.
18   * Trims non-letters from beginning and end of String.
19   * @param s for word
20   */
21⑪  public Word(String s) {
22
23      //trim beginning of word
24      int i = 0;
25      while (i < s.length() && !Character.isLetter(s.charAt(i))) {
26          i++;
27      }
28
29      //trim end of word
30      int j = s.length() - 1;
31      while (j > i && !Character.isLetter(s.charAt(j))) {
32          j--;
33      }
34
35      //gets a substring of given s based on i and j
36      //stores it as text for word
37      //e.g. 123hello321 will be stored as hello
38      this.text = s.substring(i, j + 1);
39  }
```



SyllableCounter Class

- Run the program and type a sentence ending with a period

```
Enter a sentence ending in a period.  
this class is amazing, although the homework can be difficult, everybody loves it.  
Syllables in this: 1  
Syllables in class: 1  
Syllables in is: 1  
Syllables in amazing: 1  
Syllables in although: 1  
Syllables in the: 1  
Syllables in homework: 1  
Syllables in can: 1  
Syllables in be: 1  
Syllables in difficult: 1  
Syllables in everybody: 1  
Syllables in loves: 1  
Syllables in it: 1
```

- Note the output:
 - The syllables are counted incorrectly



SyllableCounter Class

- Add a breakpoint and debug in the Debug perspective ...
 - Step Into the *countSyllables* method to see what's happening

```
10  public static void main(String[] args) {  
11      System.out.println("Enter a sentence: ");  
12  
13      //get user input of sentence  
14      Scanner in = new Scanner(System.in);  
15  
16      String input;  
17      Word w;  
18      int syllables;  
19  
20      //while there is a word to scan  
21      while(in.hasNext()) {  
22  
23          //get next token (word)  
24          input = in.next();  
25  
26          //create instance of Word class  
27          w = new Word(input);  
28  
29          //get number of syllables  
30          syllables = w.countSyllables();  
31  
32          //print out word and count of syllables  
33          System.out.println("Syllables in " + w.getText() + ": " + syllables);  
34      }  
35  
36          //break out of while loop at period  
37          if (input.endsWith(".")) break;  
38      }  
39  
40      in.close();  
41  }  
42 }
```



Word Class

- ... then fix the bug by correctly resetting the *insideSyllable* variable

```
72
73     //set flag for being inside a syllable
74     boolean insideSyllable = false;
75
76     //iterate over characters in word and look for vowels
77     for (int i = 0; i <= end; i++) {
78
79         //get each character
80         ch = this.text.charAt(i);
81
82         //determine if character is a vowel
83         //create a "character class" using regular expression,
84         //containing every vowel we're looking for (lower and upper) in word
85         String vowelRegex = "[aeiouAEIOU]";
86
87         //create pattern to match with character
88         Pattern p = Pattern.compile(vowelRegex);
89
90         //find matches in char (casted to a String)
91         Matcher m = p.matcher(ch + "");
92
93         //if it is a vowel, enter syllable
94         if (m.matches()) {
95             if (!insideSyllable) {
96                 count++;
97                 insideSyllable = true;
98             }
99         } //otherwise, exit syllable
100        } else {
101            insideSyllable = false;
102        }
103    }
104 }
```



SyllableCounter Class

- Run the program and type a sentence ending with a period

```
Enter a sentence ending in a period.  
this class is amazing, although the homework can be difficult, everybody loves it.  
Syllables in this: 1  
Syllables in class: 1  
Syllables in is: 1  
Syllables in amazing: 3  
Syllables in although: 2  
Syllables in the: 1  
Syllables in homework: 3  
Syllables in can: 1  
Syllables in be: 1  
Syllables in difficult: 3  
Syllables in everybody: 5  
Syllables in loves: 2  
Syllables in it: 1
```

- Note the output:
 - It's not perfect, but it's getting there!