Debugging

Brandon Krakowsky

Penn Engineering

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

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

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

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

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”

```
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;
```

Toggle Breakpoint ⇧⌘B
Disable Breakpoint ⇧Double Click
Run to Line ⌥⌘Click
Go to Annotation ⌘1
Validate

```
i++) {
Count is: " + count);
```

- 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;
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

(x)= Variables | Breakpoints | Expressions

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

- 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

(x)= Variables | Breakpoints | Expressions

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

```java
 */
public class Person {

    /**
     * Name of person.
     */
    private String name;

    /**
     * Set name of person with given name.
     * @param name for person
     */
    public void setName(String name) {
        this.name = name;
    }

    /**
     * Get name of person.
     * @return name
     */
    public String getName() {
        return this.name;
    }
```

Person Class

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

        //create list for people
        ArrayList<Person> people = new ArrayList<Person>();

        //Get first person's name
        String firstName = people.get(0).getName();

        //Get length of first person's name
        int firstNameLength = firstName.length();

        //Print length and name of first person
        System.out.println(firstName + " has a length of: " + firstNameLength);

    }
```

Person Class

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

```
Console   Problems   Debug Shell
<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

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

        //create list for people
        ArrayList<Person> people = new ArrayList<Person>();

        //create and add person
        Person person = new Person();
        people.add(person);

        //Get first person's name
        String firstName = people.get(0).getName();

        //Get length of first person's name
        int firstNameLength = firstName.length();

        //Print length and name of first person
        System.out.println(firstName + " has a length of: " + firstNameLength);

    }
}
```

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

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

        //create list for people
        ArrayList<Person> people = new ArrayList<Person>();

        //create and add person
        Person person = new Person();
        people.add(person);

        //set person's name
        person.setName("Brandon");

        //Get first person's name
        String firstName = people.get(0).getName();

        //Get length of first person's name
        int firstNameLength = firstName.length();

        //Print length and name of first person
        System.out.println(firstName + " has a length of: " + firstNameLength);

    }
```

Person Class

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

Word Class

SyllableCounter.java | Word.java

```java
import java.util.regex.Matcher;

/**
 * Represents a word.
 * @author lbrandon
 *
 */
public class Word {

    /**
     * Text for word.
     */
    private String text;
```

Word Class

```java
    /**
     * Creates a word with given String.
     * Trims non-letters from beginning and end of String.
     * @param s for word
     */
    public Word(String s) {

        //trim beginning of word
        int i = 0;
        while (i < s.length() && !Character.isLetter(s.charAt(i))) {
            i++;
        }

        //trim end of word
        int j = s.length() - 1;
        while (j > i && !Character.isLetter(s.charAt(j))) {
            j--;
        }

        //gets a substring of given s based on i and j
        //stores it as text for word
        //e.g. 123hello321 will be stored as hello
        this.text = s.substring(i, j);
    }
```

Word Class

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

Word Class

```java
    /**
     * Counts syllables in word.
     * @return syllable count
     */
    public int countSyllables() {

        //set initial count of syllables
        int count = 0;

        //get index of last character in word
        int end = this.text.length() - 1;

        //if word has 0 chars, consider 0 syllables
        if (end < 0) {
            return 0;
        }

        //An 'e' at the end of the word doesn't count as a vowel
        //So decrement end index
        char ch = this.text.charAt(end);
        if (ch == 'e' || ch == 'E') {
            end--;
        }
```

Word Class

```java
        //set flag for being inside a syllable
        boolean insideSyllable = false;

        //iterate over characters in word and look for vowels
        for (int i = 0; i <= end; i++) {

            //get each character
            ch = this.text.charAt(i);

            //determine if character is a vowel
            //create a "character class" using regular expression,
            //containing every vowel we're looking for (lower and upper) in word
            String vowelRegex = "[aeiouAEIOU]";

            //create pattern to match with character
            Pattern p = Pattern.compile(vowelRegex);

            //find matches in char (casted to a String)
            Matcher m = p.matcher(ch + "");

            //if it is a vowel, enter syllable
            if (m.matches()) {
                if (!insideSyllable) {
                    count++;
                    insideSyllable = true;
                }
            }
        }
```

Word Class

```
        //every word has at least one syllable
        if (count == 0) {
            count = 1;
        }

        //return the count
        return count;
    }
```

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

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

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

- This means that Step Into MIGHT first enter another method like the *System.out.println* method or a method in the *String* or *StringBuilder* class. You'll have to Step Return out of this 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

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

        System.out.println("Enter a sentence: ");

        //get user input of sentence
        Scanner in = new Scanner(System.in);

        String input;
        Word w;
        int syllables;

        //while there is a word to scan
        while(in.hasNext()) {

            //get next token (word)
            input = in.next();

            //create instance of Word class
            w = new Word(input);

            //get number of syllables
            syllables = w.countSyllables();

            //print out word and count of syllables
            System.out.println("Syllables in " + w.getText() + ": " + syllables);

            //break out of while loop at period
            if (input.endsWith(".")) break;
        }

        in.close();

    }
```

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

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

Word Class

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

```java
    /**
     * Creates a word with given String.
     * Trims non-letters from beginning and end of String.
     * @param s for word
     */
    public Word(String s) {

        //trim beginning of word
        int i = 0;
        while (i < s.length() && !Character.isLetter(s.charAt(i))) {
            i++;
        }

        //trim end of word
        int j = s.length() - 1;
        while (j > i && !Character.isLetter(s.charAt(j))) {
            j--;
        }

        //gets a substring of given s based on i and j
        //stores it as text for word
        //e.g. 123hello321 will be stored as hello
        this.text = s.substring(i, j);
    }

```

Word Class

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

```java
    /**
     * Creates a word with given String.
     * Trims non-letters from beginning and end of String.
     * @param s for word
     */
    public Word(String s) {

        //trim beginning of word
        int i = 0;
        while (i < s.length() && !Character.isLetter(s.charAt(i))) {
            i++;
        }

        //trim end of word
        int j = s.length() - 1;
        while (j > i && !Character.isLetter(s.charAt(j))) {
            j--;
        }

        //gets a substring of given s based on i and j
        //stores it as text for word
        //e.g. 123hello321 will be stored as hello
        this.text = s.substring(i, j + 1);
    }
```

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

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

```java
        //set flag for being inside a syllable
        boolean insideSyllable = false;

        //iterate over characters in word and look for vowels
        for (int i = 0; i <= end; i++) {

            //get each character
            ch = this.text.charAt(i);

            //determine if character is a vowel
            //create a "character class" using regular expression,
            //containing every vowel we're looking for (lower and upper) in word
            String vowelRegex = "[aeiouAEIOU]";

            //create pattern to match with character
            Pattern p = Pattern.compile(vowelRegex);

            //find matches in char (casted to a String)
            Matcher m = p.matcher(ch + "");

            //if it is a vowel, enter syllable
            if (m.matches()) {
                if (!insideSyllable) {
                    count++;
                    insideSyllable = true;
                }
            //otherwise, exit syllable
            } else {
                insideSyllable = false;
            }
        }
```

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!

Debugging Examples - Battleship

Ocean Class

- To watch ships being placed in the ocean …
  - You can place breakpoints accordingly and Step Into calls to the *placeShipAt* method

BattleshipGame.java | Ocean.java | Ship.java

```java
76      void placeAllShipsRandomly() {
77
78          Random rand = new Random();
79          int row;
80          int column;
81          boolean horizontal;
82
83          //place battleships
84          for (int i = 0; i < Ocean.NUM_BATTLESHIPS; i++) {
85              Ship battleship = new Battleship();
86              row = rand.nextInt(10);
87              column = rand.nextInt(10);
88              horizontal = rand.nextInt(2) == 0 ? false : true;
89              while(!battleship.okToPlaceShipAt(row, column, horizontal, this)) {
90                  row = rand.nextInt(10);
91                  column = rand.nextInt(10);
92                  horizontal = rand.nextInt(2) == 0 ? false : true;
93              }
94              battleship.placeShipAt(row, column, horizontal, this);
95          }
```

Ship Class

- Then, Step Over each line in your placeShipAt method

BattleshipGame.java | Ocean.java | Ship.java

```
402     void placeShipAt(int row, int column, boolean horizontal, Ocean ocean) {
403         this.setBowRow(row);
404         this.setBowColumn(column);
405         this.setHorizontal(horizontal);
406
407         int shipLength = this.getLength();
408         if (this.isHorizontal()) {
409             int stern = column - (shipLength - 1);
410             for (int i = column; i >= stern; i--) {
411                 ocean.getShipArray()[row][i] = this;
412             }
```