OOP with Java

13. Debugging

Thomas Weise · 汤卫思

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

Outline

website

Introduction

- When writing programs, mistakes are impossible to avoid

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it
- Then there are errors which make your programs crash all the time.

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it
- Then there are errors which make your programs crash all the time.
- Harder to spot are errors that make your program crash only sometimes.

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it
- Then there are errors which make your programs crash all the time.
- Harder to spot are errors that make your program crash only sometimes.
- Harder are errors which do not make your program crash, but lead to wrong output, especially if the output is only sometimes wrong and if it is only a bit wrong.

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it
- Then there are errors which make your programs crash all the time.
- Harder to spot are errors that make your program crash only sometimes.
- Harder are errors which do not make your program crash, but lead to wrong output, especially if the output is only sometimes wrong and if it is only a bit wrong.
- Anyway, once we have recognized that there is a error/bug, we need to find it to fix it.

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Introduction

- When writing programs, mistakes are impossible to avoid
- There are a lot of different types of errors that can happen
- The easiest ones are syntax errors and stuff that the compiler picks up and warns you about it
- Then there are errors which make your programs crash all the time.
- Harder to spot are errors that make your program crash only sometimes.
- Harder are errors which do not make your program crash, but lead to wrong output, especially if the output is only sometimes wrong and if it is only a bit wrong.
- Anyway, once we have recognized that there is a error/bug, we need to find it to fix it.
- This process is called debugging.

In Lesson 25: *Exceptions*, we will learn to interpret the error print created by a crashing program.

Debugging

- A debugger is a very powerful tool.

Debugging

- A debugger is a very powerful tool.
- It helps us to run a program in a very special way

Debugging

- A debugger is a very powerful tool.
- It helps us to run a program in a very special way:
  - If we mark a line of code as "break point", the process will stop at this line

Debugging

- A debugger is a very powerful tool.
- It helps us to run a program in a very special way:
  - If we mark a line of code as "break point", the process will stop at this line
  - We can then see, e.g., the values of the variables at this point in time, to check if they are what they should be

Debugging

- A debugger is a very powerful tool.
- It helps us to run a program in a very special way:
  - If we mark a line of code as "break point", the process will stop at this line
  - We can then see, e.g., the values of the variables at this point in time, to check if they are what they should be
  - We can continue to execute the program step-by-step and see how the variables change

Debugging

- A debugger is a very powerful tool.
- It helps us to run a program in a very special way:
  - If we mark a line of code as "break point", the process will stop at this line
  - We can then see, e.g., the values of the variables at this point in time, to check if they are what they should be
  - We can continue to execute the program step-by-step and see how the variables change
- This way, we can find errors much easier than by reading the code alone

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)
  2. an `int` value `search`, which may or may not occur in `array`

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)
  2. an `int` value `search`, which may or may not occur in `array`
- The method returns

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)
  2. an `int` value `search`, which may or may not occur in `array`
- The method returns:
  1. the exact index of `search` in `array` if `array` contains `search`, i.e., in this case `search == array[binarySearch(array, search)]` holds

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)
  2. an `int` value `search`, which may or may not occur in `array`
- The method returns:
  1. the exact index of `search` in `array` if `array` contains `search`, i.e., in this case `search == array[binarySearch(array, search)]` holds
  2. `-1` if `search` does not occur inside `array`

Search in Sorted Array

- A colleague provides you with a method `binarySearch` for searching a value inside a sorted `int` array
- The method receives two parameters:
  1. an `int` array `array`, which must be sorted (in ascending order)
  2. an `int` value `search`, which may or may not occur in `array`
- The method returns:
  1. the exact index of `search` in `array` if `array` contains `search`, i.e., in this case `search == array[binarySearch(array, search)]` holds
  2. `-1` if `search` does not occur inside `array`
- If `array` is not sorted, the behavior of the method would be unspecified

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`
  - If `array[midIndex]<search`, then `search` must either come after `midIndex` or is not in `array` so I continue to search in the same way in the upper part of my division

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`
  - If `array[midIndex]<search`, then `search` must either come after `midIndex` or is not in `array` so I continue to search in the same way in the upper part of my division
  - If `array[midIndex]>search`, then `search` must either come before `midIndex` or is not in `array` so I continue to search in the same way in the lower part of my division

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`
  - If `array[midIndex]<search`, then `search` must either come after `midIndex` or is not in `array` so I continue to search in the same way in the upper part of my division
  - If `array[midIndex]>search`, then `search` must either come before `midIndex` or is not in `array` so I continue to search in the same way in the lower part of my division
  - Otherwise, it must be that `array[midIndex]==search` and I can return `midIndex`

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`
  - If `array[midIndex]<search`, then `search` must either come after `midIndex` or is not in `array` so I continue to search in the same way in the upper part of my division
  - If `array[midIndex]>search`, then `search` must either come before `midIndex` or is not in `array` so I continue to search in the same way in the lower part of my division
  - Otherwise, it must be that `array[midIndex]==search` and I can return `midIndex`
  - If I can no longer divide my array because my division is empty, then it means that `search` does not appear in it

Binary Search

- Your colleague implemented (or better, tried to implement) binary search for this purpose.
- Binary search makes use of the fact that `array` is sorted. The idea is:
  - If I know that `array[i]`, i.e., the element at index `i`, is greater than `search` (`array[i] > search`), then I know that `search` cannot appear at or after index `i`
  - If I know that `array[i] < search`, then I know that `search` cannot appear at or before index `i`
  - This means I can successively divide the array into three pieces: the element at the middle index `midIndex`, the elements before `midIndex`, and the elements after `midIndex`
  - If `array[midIndex]<search`, then `search` must either come after `midIndex` or is not in `array` so I continue to search in the same way in the upper part of my division
  - If `array[midIndex]>search`, then `search` must either come before `midIndex` or is not in `array` so I continue to search in the same way in the lower part of my division
  - Otherwise, it must be that `array[midIndex]==search` and I can return `midIndex`
  - If I can no longer divide my array because my division is empty, then it means that `search` does not appear in it
- The array size decreases to less than 50% in each step, the search will complete in at most $\left\lceil \log_2 \texttt{array.length} \right\rceil$ steps, i.e., much faster than searching from start to end

Wrong Implementation of Binary Search

Listing: Your Colleague's (wrong) Implementation of Binary Search

```
/** An erroneous implementation of Binary Search */
public class BinarySearchWrong {

  /** Find the index of value "search" in the sorted array "array".
   * @param array the array to search inside, must be sorted
   * @param search the value to search
   * @return the index of search, or -1 if it does not occur in "array"
   */
  static int binarySearch(int[] array, int search) {
    int lowerBound = 0; // the index of the first array element
    int upperBound = array.length - 1; // the index of the last array element

    while (lowerBound < upperBound) { // as long as current division is not empty
      int midIndex = (lowerBound + upperBound) / 2; // compute mid index

      if (array[midIndex] < search) { // if element in middle is smaller than search
        lowerBound = midIndex + 1; // search only above the middle from now on
      } else { // otherwise
        if (array[midIndex] > search) { // if element is bigger than search
          upperBound = midIndex - 1; // search only below the middle from now on
        } else { // ok, element is neither smaller or bigger, so it must be equal
          return midIndex; // then array[midIndex] == search must hold
        }
      }
    }
    return -1; // we did not find the element
  }
}
```

Your Test of the Wrong Implementation of Binary Search

Listing: Your Test of this (wrong) Implementation of Binary Search

```
/** A program testing BinarySearchWrong. */
public class BinarySearchWrongTest {

  /** The main routine
   * @param args
   *          we ignore this parameter */
  public static final void main(String[] args) {

    int[] array = {0, 1, 2, 3, 5, 6};

    System.out.println("index␣of␣0:␣" + BinarySearchWrong.binarySearch(array, 0)); //$NON-NLS-1$
    System.out.println("index␣of␣1:␣" + BinarySearchWrong.binarySearch(array, 1)); //$NON-NLS-1$
    System.out.println("index␣of␣2:␣" + BinarySearchWrong.binarySearch(array, 2)); //$NON-NLS-1$
    System.out.println("index␣of␣3:␣" + BinarySearchWrong.binarySearch(array, 3)); //$NON-NLS-1$
    System.out.println("index␣of␣5:␣" + BinarySearchWrong.binarySearch(array, 5)); //$NON-NLS-1$
    System.out.println("index␣of␣6:␣" + BinarySearchWrong.binarySearch(array, 6)); //$NON-NLS-1$

    System.out.println("index␣of␣-1:␣" + BinarySearchWrong.binarySearch(array, -1)); //$NON-NLS-1$
    System.out.println("index␣of␣4:␣" + BinarySearchWrong.binarySearch(array, 4)); //$NON-NLS-1$
    System.out.println("index␣of␣7:␣" + BinarySearchWrong.binarySearch(array, 7)); //$NON-NLS-1$
  }
}
```

The Output

Listing: The expected output

```
index of 0: 0
index of 1: 1
index of 2: 2
index of 3: 3
index of 5: 4
index of 6: 5
index of -1: -1
index of 4: -1
index of 7: -1
```

Listing: The actual output

```
index of 0: 0
index of 1: -1
index of 2: 2
index of 3: -1
index of 5: 4
index of 6: -1
index of -1: -1
index of 4: -1
index of 7: -1
```

What now?

- We found that the method does not behave as specified

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`
- Let's use the Eclipse debugger

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`
- Let's use the Eclipse debugger:
  1. put a break point into `BinarySearchWrongTest.java` at the corresponding line

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`
- Let's use the Eclipse debugger:
  1. put a break point into `BinarySearchWrongTest.java` at the corresponding line
  2. execute the program step by step, tracing into the call to `binarySearch`

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`
- Let's use the Eclipse debugger:
  1. put a break point into `BinarySearchWrongTest.java` at the corresponding line
  2. execute the program step by step, tracing into the call to `binarySearch`
  3. check where and why it fails

What now?

- We found that the method does not behave as specified
- It fails for `array = 0, 1, 2, 3, 5, 6;` and `search=1`
- Let's use the Eclipse debugger:
  1. put a break point into `BinarySearchWrongTest.java` at the corresponding line
  2. execute the program step by step, tracing into the call to `binarySearch`
  3. check where and why it fails
- Based on the findings, fix the code

The Debugging Process in Screenshots

javaExamples - Java - 13_debugging/src/BinarySearchWrongTest.java - Eclipse

Package Explorer

- 13_debugging [javaExamples
  - src
    - (default package)
      - BinarySearchRight.java
      - BinarySearchRightTest.jav
      - BinarySearchWrong.java
      - BinarySearchWrongTest.ja
  - JRE System Library [java-8-op
  - .classpath
  - .gitignore
  - .project
  - make_linux.sh
  - README.md
- javaExamples [javaExamples

BinarySearchWrong.java | BinarySearchWrongTest.java

```
1  /** A program testing BinarySearchWrong. */
2  public class BinarySearchWrongTest {
3
4    /**
5     * The main routine
6     *
7     * @param args
8     *          we ignore this parameter for now
9     */
10   public static final void main(String[] args) {
11
12     int[] array = {0, 1, 2, 3, 5, 6};
13
14     System.out.println("index of 0: " + BinarySearchWrong.binarySearch(array, 0)); /
                                1: " + BinarySearchWrong.binarySearch(array, 1)); /
                                2: " + BinarySearchWrong.binarySearch(array, 2)); /
                                3: " + BinarySearchWrong.binarySearch(array, 3)); /
                                5: " + BinarySearchWrong.binarySearch(array, 5)); /
                                6: " + BinarySearchWrong.binarySearch(array, 6)); /

                                -1: " + BinarySearchWrong.binarySearch(array, -1));
                                4: " + BinarySearchWrong.binarySearch(array, 4)); /
                                7: " + BinarySearchWrong.binarySearch(array, 7)); /
```

Toggle Breakpoint Shift+Ctrl+B
Disable Breakpoint
Go to Annotation Ctrl+1
Team
Add Bookmark...
Add Task...
✓ Show Quick Diff Shift+Ctrl+Q
Show Annotations
✓ Show Line Numbers
Folding
Preferences...
Breakpoint Properties...

Console | Progress

```
<terminated> BinarySearchRightTest [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4
index of 0: 0
index of 1: 1
index of 2: 2
```

Writable | Smart Insert | 17 : 30

The Debugging Process in Screenshots

javaExamples - Java - 13_debugging/src/BinarySearchWrongTest.java - Eclipse

1 BinarySearchWrongTest
2 BinarySearchRightTest
3 BinarySearchWrong
Debug As
Debug Configurations...
Organize Favorites...

Package Explorer
- 13_debugging [javaExamples
  - src
    - (default package)
      - BinarySearchRight.java
      - BinarySearchRightTest.jav
      - BinarySearchWrong.java
      - BinarySearchWrongTest.ja
  - JRE System Library [java-8-op
  - .classpath
  - .gitignore
  - .project
  - make_linux.sh
  - README.md
- javaExamples [javaExamples

BinarySearchWrong.ja

```
1  /** A program tes
2  public class Bina
3
4    /**
5     * The main routine
6     *
7     * @param args
8     *          we ignore this parameter for now
9     */
10   public static final void main(String[] args) {
11
12     int[] array = {0, 1, 2, 3, 5, 6};
13
14     System.out.println("index of 0: " + BinarySearchWrong.binarySearch(array, 0)); /
15     System.out.println("index of 1: " + BinarySearchWrong.binarySearch(array, 1)); /
16     System.out.println("index of 2: " + BinarySearchWrong.binarySearch(array, 2)); /
17     System.out.println("index of 3: " + BinarySearchWrong.binarySearch(array, 3)); /
18     System.out.println("index of 5: " + BinarySearchWrong.binarySearch(array, 5)); /
19     System.out.println("index of 6: " + BinarySearchWrong.binarySearch(array, 6)); /
20
21     System.out.println("index of -1: " + BinarySearchWrong.binarySearch(array, -1));
22     System.out.println("index of 4: " + BinarySearchWrong.binarySearch(array, 4)); /
23     System.out.println("index of 7: " + BinarySearchWrong.binarySearch(array, 7)); /
24   }
25 }
26
```

Problems | Javadoc | Declaration | Console | Progress | Debug

<terminated>BinarySearchWrongTest [Java Application]
<terminated>BinarySearchWrongTest at localhost:36969
<terminated, exit value: 1>/usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:20:02 AM)

Writable | Smart Insert | 14 : 8

The Debugging Process in Screenshots

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrongTest.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended (breakpoint at line 15 in BinarySearch
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints**

Name	Value
args	String[0] (id=15)
array	(id=16)
[0]	0
[1]	1
[2]	2
[3]	3
[4]	5

[0, 1, 2, 3, 5, 6]

BinarySearchWrong.java | BinarySearchWrongTest.java

```
1  /** A program testing BinarySearchWrong. */
2  public class BinarySearchWrongTest {
3
4    /**
5     * The main routine
6     *
7     * @param args
8     *          we ignore this parameter for now
9     */
10   public static final void main(String[] args) {
11
12     int[] array = {0, 1, 2, 3, 5, 6};
13
14     System.out.println("index of 0: " + BinarySearchWrong.binarySearch(array, 0)); //$NON-NLS-1$
15     System.out.println("index of 1: " + BinarySearchWrong.binarySearch(array, 1)); //$NON-NLS-1$
16     System.out.println("index of 2: " + BinarySearchWrong.binarySearch(array, 2)); //$NON-NLS-1$
17     System.out.println("index of 3: " + BinarySearchWrong.binarySearch(array, 3)); //$NON-NLS-1$
18     System.out.println("index of 5: " + BinarySearchWrong.binarySearch(array, 5)); //$NON-NLS-1$
19     System.out.println("index of 6: " + BinarySearchWrong.binarySearch(array, 6)); //$NON-NLS-1$
20
       System.out.println("index of -1: " + BinarySearchWrong.binarySearch(array, -1)); //$NON-NLS-1$
```

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrongTest.java - Eclipse

Step Into (F5)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints

Name	Value
args	String[0] (id=15)
array	(id=16)

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
1  /** A program testing BinarySearchWrong. */
2  public class BinarySearchWrongTest {
3
4    /**
5     * The main routine
6     *
7     * @param args
8     *          we ignore this parameter for now
9     */
10   public static final void main(String[] args) {
11
12     int[] array = {0, 1, 2, 3, 5, 6};
13
14     System.out.println("index of 0: " + BinarySearchWrong.binarySearch(array, 0)); //$NON-NLS-1$
15     System.out.println("index of 1: " + BinarySearchWrong.binarySearch(array, 1)); //$NON-NLS-1$
16     System.out.println("index of 2: " + BinarySearchWrong.binarySearch(array, 2)); //$NON-NLS-1$
17     System.out.println("index of 3: " + BinarySearchWrong.binarySearch(array, 3)); //$NON-NLS-1$
18     System.out.println("index of 5: " + BinarySearchWrong.binarySearch(array, 5)); //$NON-NLS-1$
19     System.out.println("index of 6: " + BinarySearchWrong.binarySearch(array, 6)); //$NON-NLS-1$
20
       System.out.println("index of -1: " + BinarySearchWrong.binarySearch(array, -1)); //$NON-NLS-1$
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - java.lang.StringBuilder - Eclipse

Step Return (F7)

Quick Access

Debug

BinarySearchWrongTest [Java Application]
BinarySearchWrongTest at localhost:40614
Thread [main] (Suspended)
StringBuilder.<init>(String) line: 112
BinarySearchWrongTest.main(String[]) line: 15
/usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints

Name	Value
this	StringBuilder (id=21)
arg0	"index of 1: " (id=25)

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
101         super(capacity);
102     }
103
104     /**
105      * Constructs a string builder initialized to the contents of the
106      * specified string. The initial capacity of the string builder is
107      * {@code 16} plus the length of the string argument.
108      *
109      * @param   str   the initial contents of the buffer.
110      */
111     public StringBuilder(String str) {
112         super(str.length() + 16);
113         append(str);
114     }
115
116     /**
117      * Constructs a string builder that contains the same characters
118      * as the specified {@code CharSequence}. The initial capacity of
119      * the string builder is {@code 16} plus the length of the
120      * {@code CharSequence} argument.
      *
```

OK, we stepped into some Java native code. We are not interested in it and can step right out.

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrongTest.java - Eclipse

Step Into (F5)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints

Name	Value
args	String[0] (id=15)
array	(id=16)

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
1  /** A program testing BinarySearchWrong. */
2  public class BinarySearchWrongTest {
3
4    /**
5     * The main routine
6     *
7     * @param args
8     *          we ignore this parameter for now
9     */
10   public static final void main(String[] args) {
11
12     int[] array = {0, 1, 2, 3, 5, 6};
13
14     System.out.println("index of 0: " + BinarySearchWrong.binarySearch(array, 0)); //$NON-NLS-1$
15     System.out.println("index of 1: " + BinarySearchWrong.binarySearch(array, 1)); //$NON-NLS-1$
16     System.out.println("index of 2: " + BinarySearchWrong.binarySearch(array, 2)); //$NON-NLS-1$
17     System.out.println("index of 3: " + BinarySearchWrong.binarySearch(array, 3)); //$NON-NLS-1$
18     System.out.println("index of 5: " + BinarySearchWrong.binarySearch(array, 5)); //$NON-NLS-1$
19     System.out.println("index of 6: " + BinarySearchWrong.binarySearch(array, 6)); //$NON-NLS-1$
20
       System.out.println("index of -1: " + BinarySearchWrong.binarySearch(array, -1)); //$NON-NLS-1$
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 10
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints**

Name	Value
array	(id=16)
[0]	0
[1]	1
[2]	2
[3]	3
[4]	5
[5]	6
search	1

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 10
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints**

Name	Value
array	(id=16)
[0]	0
[1]	1
[2]	2
[3]	3
[4]	5
[5]	6
search	1

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 11
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints**

Name	Value
array	(id=16)
search	1
lowerBound	0

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 13
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints**

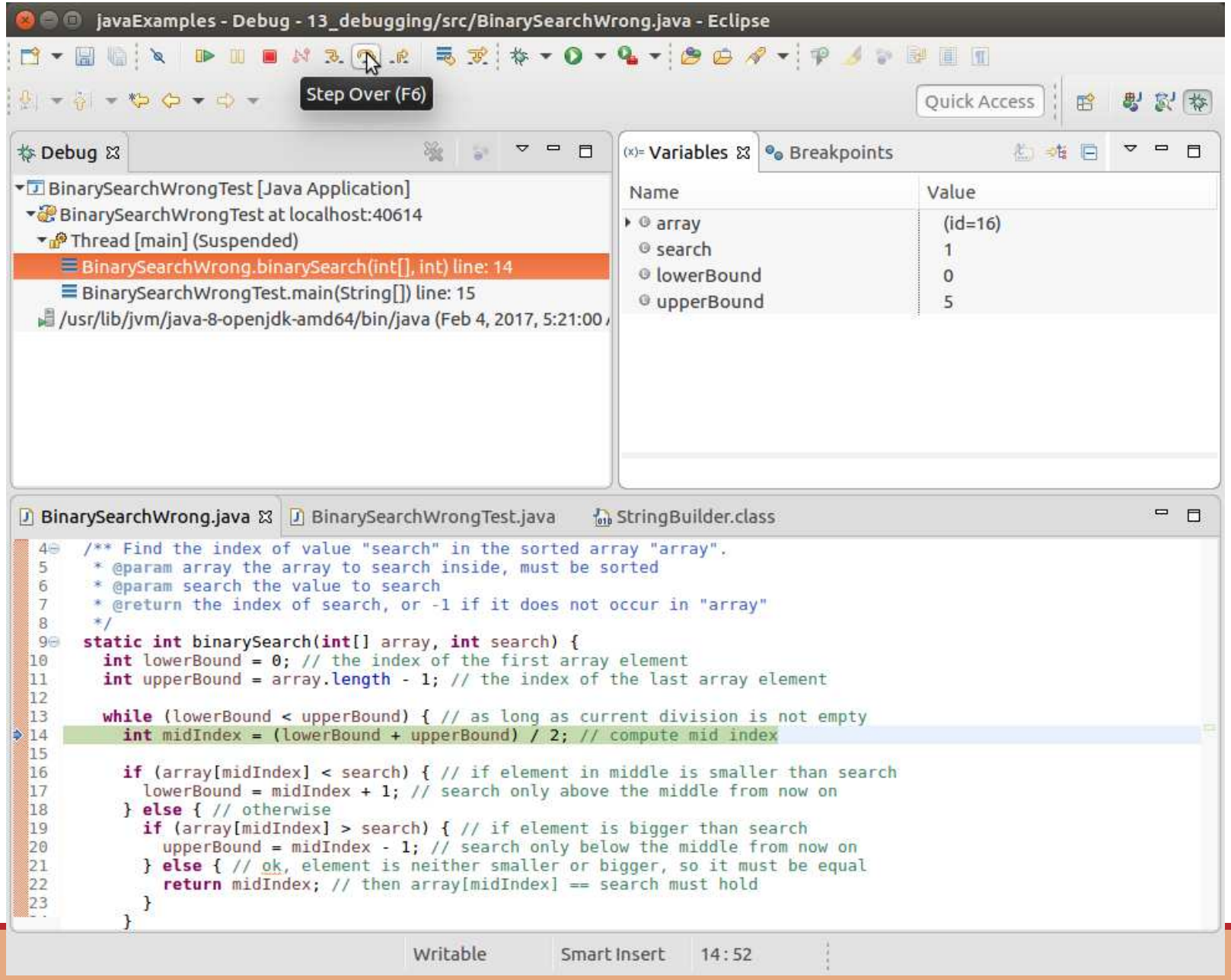
Name	Value
array	(id=16)
search	1
lowerBound	0
upperBound	5

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots



The Debugging Process in Screenshots

Source Refactor Navigate Search Project Run Window Help

javaExamples - Debug - 13_debugging/s

Eclipse

New Window
Editor
Appearance
Show View
Perspective
Navigation
Preferences

Ant
Breakpoints
Console
Debug
Display
Error Log
Expressions
Outline
Tasks
Variables
Other...

Quick Access

Debug

BinarySearchWrongTest [Java Application]
BinarySearchWrongTest at localhost:40614
Thread [main] (Suspended)
BinarySearchWrong.binarySearch(int[], int) line: 16
BinarySearchWrongTest.main(String[]) line: 15
/usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

	Value
	(id=16)
searc	1
lower	0
upper	5
midIn	2

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array e
12
13    while (lowerBound < upperBound) { // as long as current division is
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid inde
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

We can not just check variables, but complete expressions!

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
Add new expression	

No details to display for the current selection.

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array e
12
13    while (lowerBound < upperBound) { // as long as current division is
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid inde
15
16      if (array[midIndex] < search) { // if element in middle is smalle
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

Such as the value of `array` in the middle of the current selection

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
array[midIndex]	

No details to display for the current selection.

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints** | **Expressions**

Name	Value
"array[midIndex]"	2
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

Writable | Smart Insert | 14 : 52

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	2
"search"	1
"lowerBound"	0
"upperBound"	5
"midIndex"	2
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array e
12
13    while (lowerBound < upperBound) { // as long as current division is
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid inde
15
16      if (array[midIndex] < search) { // if element in middle is smalle
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

For the sake of simplicity, we also add the interesting variables

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

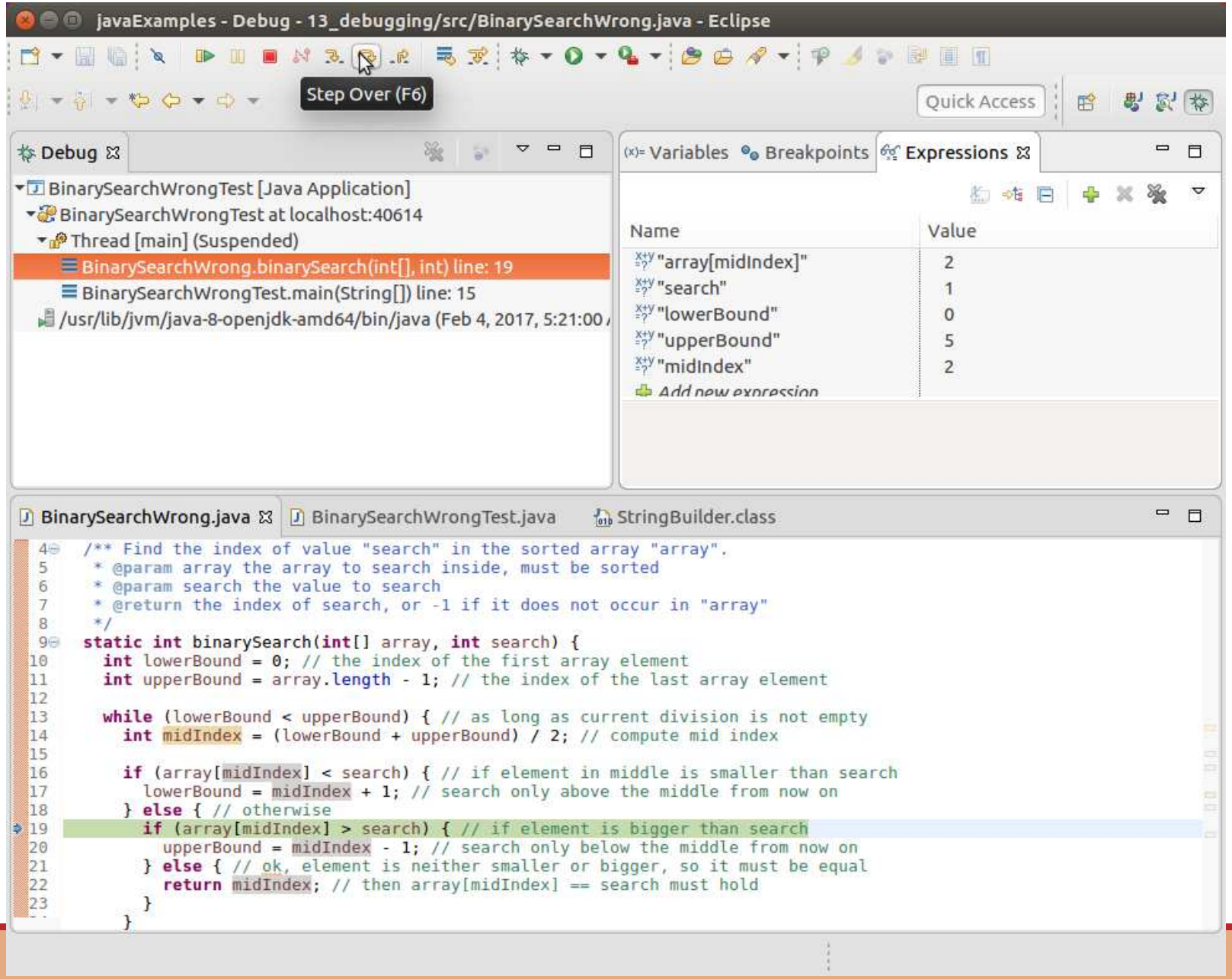
Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	2
"search"	1
"lowerBound"	0
"upperBound"	5
"midIndex"	2
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

The Debugging Process in Screenshots



The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 20
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	2
"search"	1
"lowerBound"	0
"upperBound"	5
"midIndex"	2
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 21
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	2
"search"	1
"lowerBound"	0
"upperBound"	1
"midIndex"	2
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 13
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | **Breakpoints** | **Expressions**

Name	Value
"array[midIndex]"	<error(s)_during_the_evaluatio
"search"	1
"lowerBound"	0
"upperBound"	1
"midIndex"	<error(s)_during_the_evaluatio
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 13
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

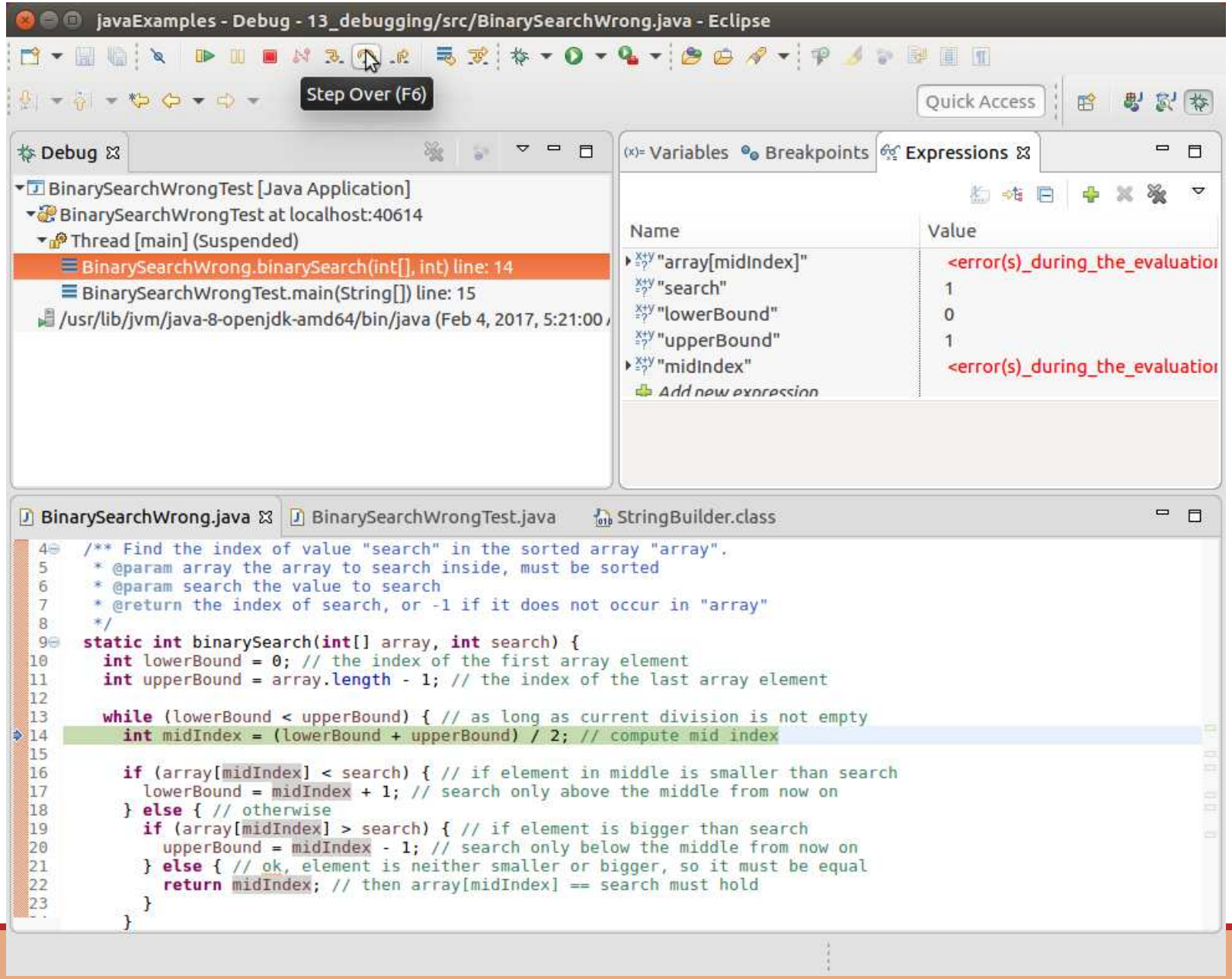
Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	<error(s)_during_the_evaluatio
"search"	1
"lowerBound"	0
"upperBound"	1
"midIndex"	<error(s)_during_the_evaluatio
Add new expression	

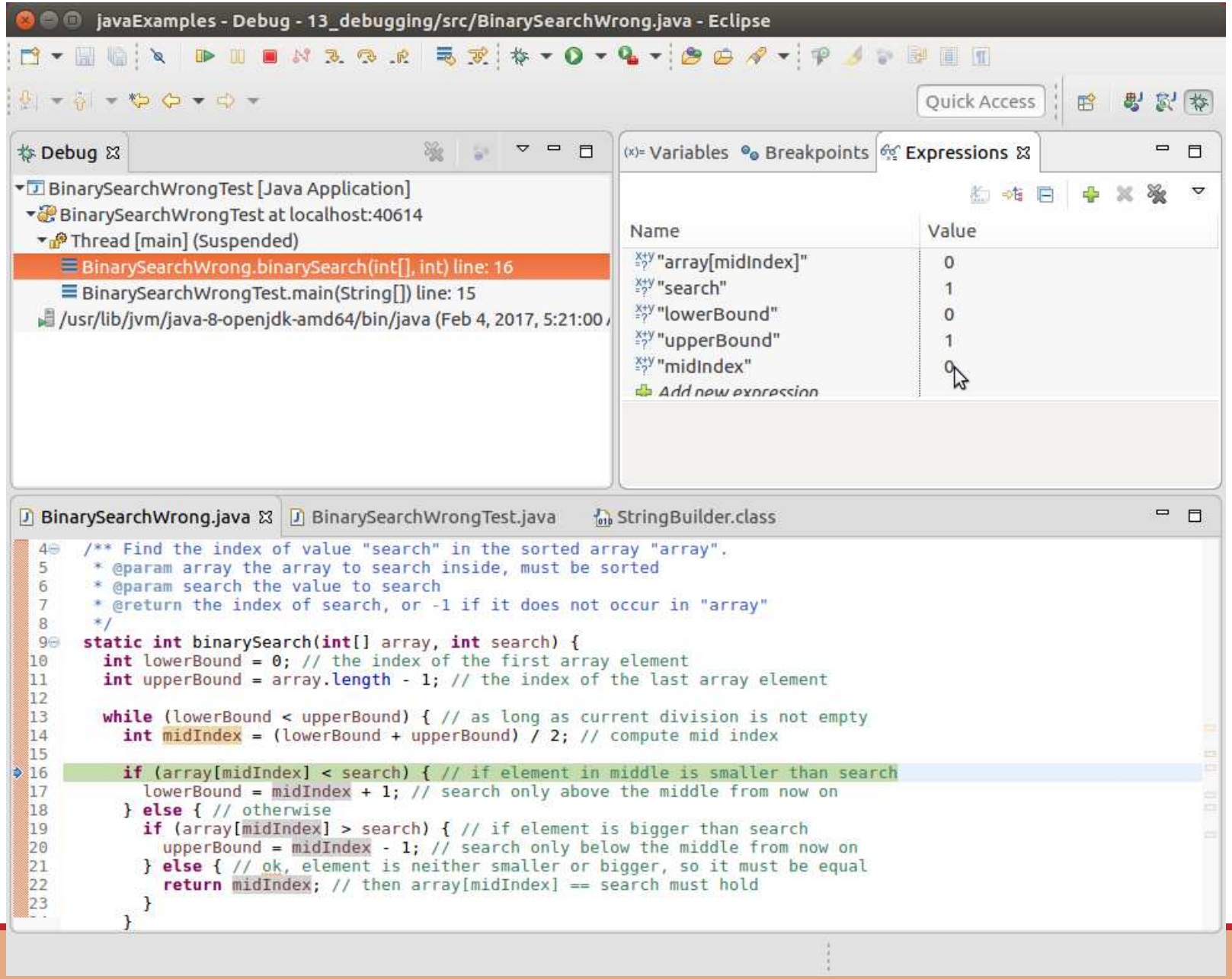
BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

The Debugging Process in Screenshots



The Debugging Process in Screenshots



The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Step Over (F6)

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 16
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

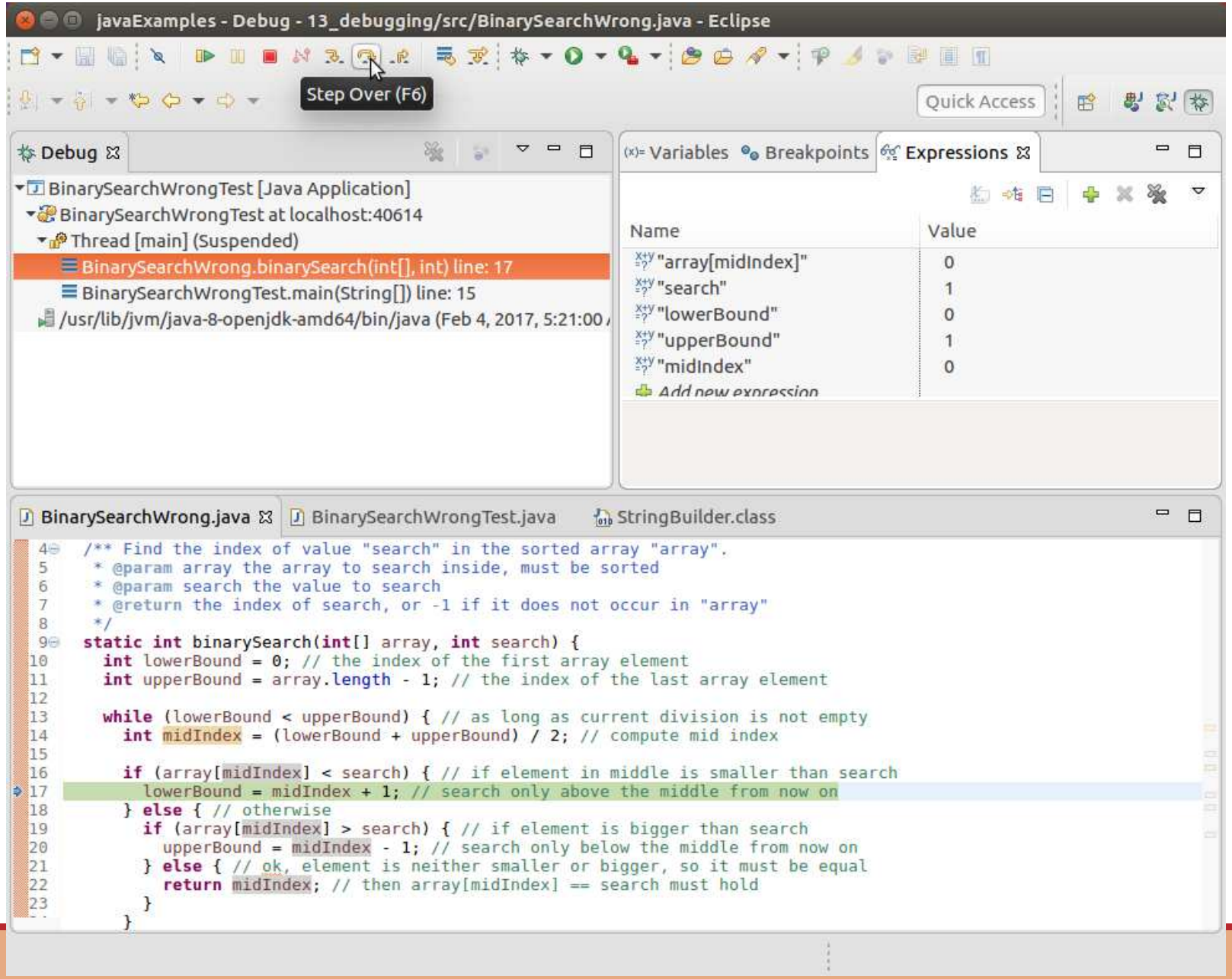
Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	0
"search"	1
"lowerBound"	0
"upperBound"	1
"midIndex"	0
Add new expression	

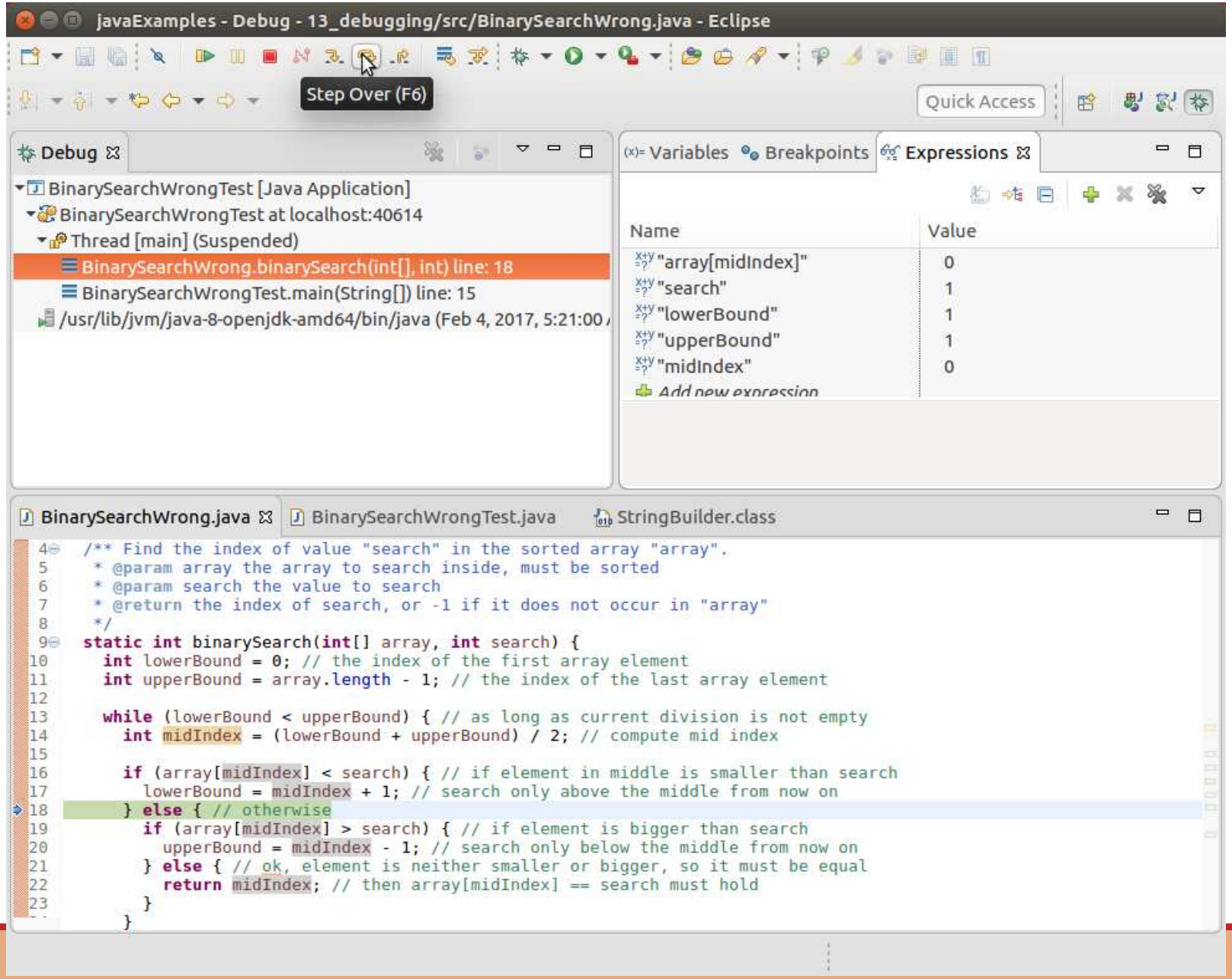
BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
4   /** Find the index of value "search" in the sorted array "array".
5    * @param array the array to search inside, must be sorted
6    * @param search the value to search
7    * @return the index of search, or -1 if it does not occur in "array"
8    */
9   static int binarySearch(int[] array, int search) {
10    int lowerBound = 0; // the index of the first array element
11    int upperBound = array.length - 1; // the index of the last array element
12
13    while (lowerBound < upperBound) { // as long as current division is not empty
14      int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16      if (array[midIndex] < search) { // if element in middle is smaller than search
17        lowerBound = midIndex + 1; // search only above the middle from now on
18      } else { // otherwise
19        if (array[midIndex] > search) { // if element is bigger than search
20          upperBound = midIndex - 1; // search only below the middle from now on
21        } else { // ok, element is neither smaller or bigger, so it must be equal
22          return midIndex; // then array[midIndex] == search must hold
23        }
      }
```

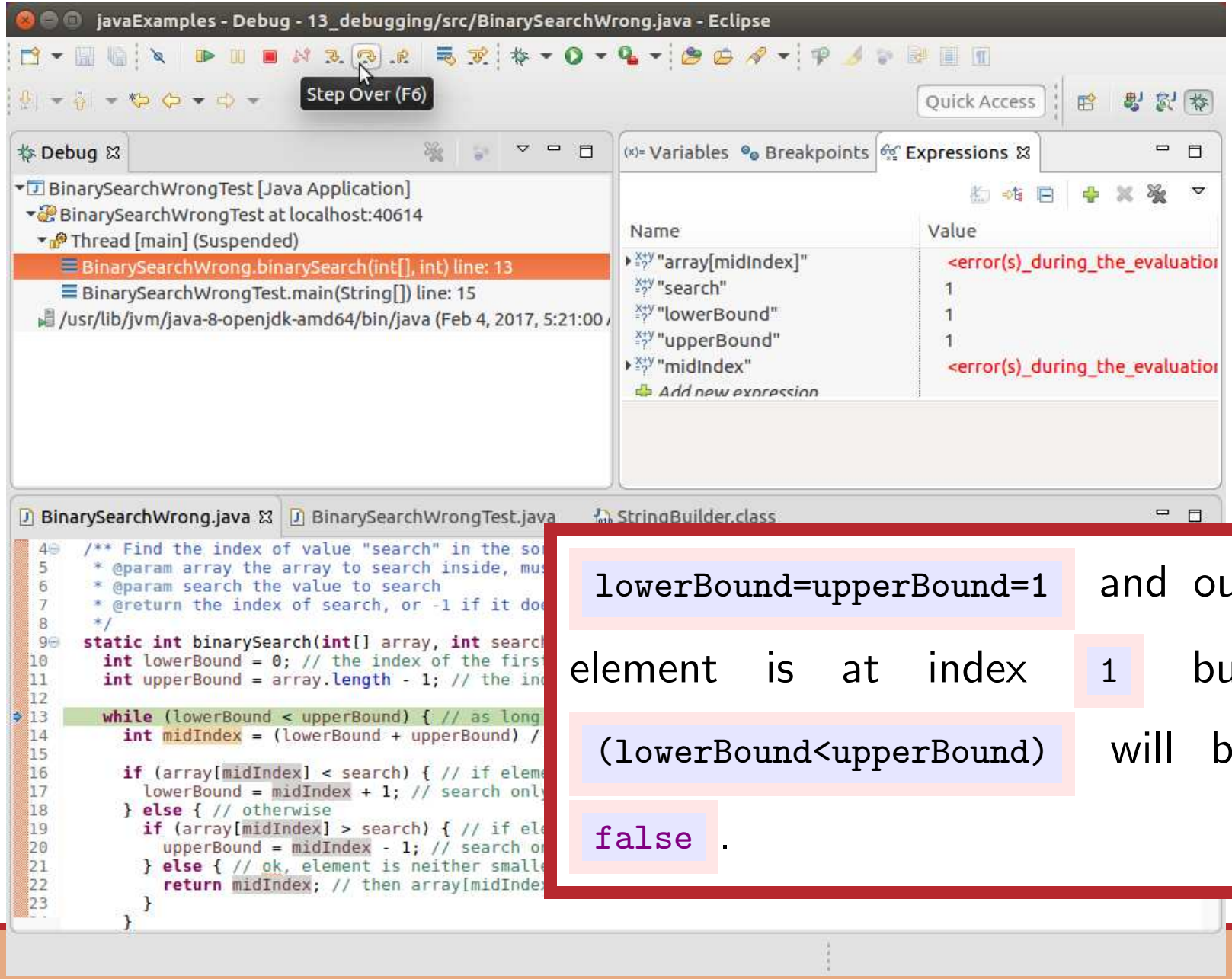
The Debugging Process in Screenshots



The Debugging Process in Screenshots



The Debugging Process in Screenshots



`lowerBound=upperBound=1` and our element is at index `1` but `(lowerBound<upperBound)` will be `false`.

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Quick Access

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 26
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	<error(s)_during_the_evaluatio
"search"	1
"lowerBound"	1
"upperBound"	1
"midIndex"	<error(s)_during_the_evaluatio
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
10      int lowerBound = 0; // the index of the first array element
11      int upperBound = array.length - 1; // the index of the last array element
12
13      while (lowerBound < upperBound) { // as long as current division is
14        int midIndex = (lowerBound + upperBound) / 2; // compute mid inde
15
16        if (array[midIndex] < search) { // if element in middle is smalle
17          lowerBound = midIndex + 1; // search only above the middle from
18        } else { // otherwise
19          if (array[midIndex] > search) { // if element is bigger than se
20            upperBound = midIndex - 1; // search only below the middle from now on
21          } else { // ok, element is neither smaller or bigger, so it must be equal
22            return midIndex; // then array[midIndex] == search must hold
23          }
24        }
25      }
26      return -1; // we did not find the element
27    }
28  }
29
```

Debug Current Instruction Pointer

We found the error, we can stop debugging.

The Debugging Process in Screenshots

javaExamples - Debug - 13_debugging/src/BinarySearchWrong.java - Eclipse

Debug

- BinarySearchWrongTest [Java Application]
  - BinarySearchWrongTest at localhost:40614
    - Thread [main] (Suspended)
      - BinarySearchWrong.binarySearch(int[], int) line: 26
      - BinarySearchWrongTest.main(String[]) line: 15
    - /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb 4, 2017, 5:21:00

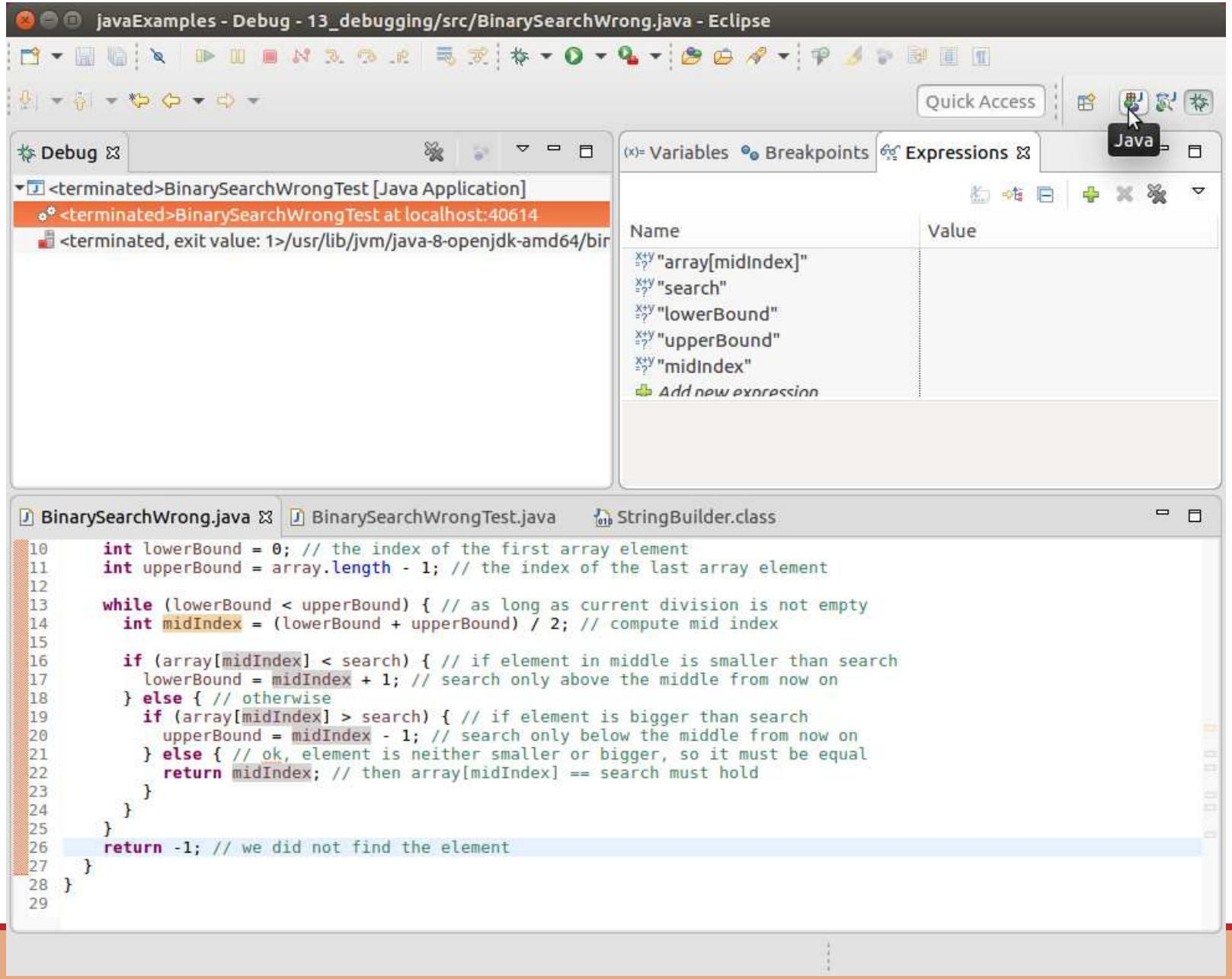
Variables | Breakpoints | Expressions

Name	Value
"array[midIndex]"	<error(s)_during_the_evaluatio
"search"	1
"lowerBound"	1
"upperBound"	1
"midIndex"	<error(s)_during_the_evaluatio
Add new expression	

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
10      int lowerBound = 0; // the index of the first array element
11      int upperBound = array.length - 1; // the index of the last array element
12
13      while (lowerBound < upperBound) { // as long as current division is not empty
14        int midIndex = (lowerBound + upperBound) / 2; // compute mid index
15
16        if (array[midIndex] < search) { // if element in middle is smaller than search
17          lowerBound = midIndex + 1; // search only above the middle from now on
18        } else { // otherwise
19          if (array[midIndex] > search) { // if element is bigger than search
20            upperBound = midIndex - 1; // search only below the middle from now on
21          } else { // ok, element is neither smaller or bigger, so it must be equal
22            return midIndex; // then array[midIndex] == search must hold
23          }
24        }
25      }
26      return -1; // we did not find the element
27    }
28  }
29
```

The Debugging Process in Screenshots



The Debugging Process in Screenshots

javaExamples - Java - 13_debugging/src/BinarySearchWrong.java - Eclipse

Package Explorer

- 13_debugging [javaExamples
  - src
    - (default package)
      - BinarySearchRight.java
      - BinarySearchRightTest.jav
      - BinarySearchWrong.java
      - BinarySearchWrongTest.ja
  - JRE System Library [java-8-op
  - .classpath
  - .gitignore
  - .project
  - make_linux.sh
  - README.md
- javaExamples [javaExamples

BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
1  /** An erroneous implementation of Binary Search */
2  public class BinarySearchWrong {
3
4    /** Find the index of value "search" in the sorted array "array".
5     * @param array the array to search inside, must be sorted
6     * @param search the value to search
7     * @return the index of search, or -1 if it does not occur in "array"
8     */
9    static int binarySearch(int[] array, int search) {
10     int lowerBound = 0; // the index of
11     int upperBound = array.length - 1; /
12
13     while (lowerBound upperBound) { //
14       int midIndex = lowerBound + upper
15
16       if (array[midIndex] < search) { //
17         lowerBound = midIndex + 1; // se
18       } else { // otherwise
19         if (array[midIndex] > search) {
20           upperBound = midIndex - 1; // search only below the middle from now on
21         } else { // ok, element is neither smaller or bigger, so it must be equal
22           return midIndex; // then array[midIndex] == search must hold
23         }
24       }
25     }
26     return -1; // we did not find the element
27   }
28 }
29
```

it should not be

`(lowerBound<upperBound)`

Problems | Javadoc | Declaration | Console | Progress | Debug

```
<terminated> BinarySearchWrongTest [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb
index of 0: 0
```

Writable | Smart Insert | 13 : 23

The Debugging Process in Screenshots

javaExamples - Java - 13_debugging/src/BinarySearchWrong.java - Eclipse

Package Explorer

- 13_debugging [javaExamples
  - src
    - (default package)
      - BinarySearchRight.java
      - BinarySearchRightTest.jav
      - BinarySearchWrong.java
      - BinarySearchWrongTest.ja
  - JRE System Library [java-8-op
  - .classpath
  - .gitignore
  - .project
  - make_linux.sh
  - README.md
- javaExamples [javaExamples

*BinarySearchWrong.java | BinarySearchWrongTest.java | StringBuilder.class

```
1  /** An erroneous implementation of Binary Search */
2  public class BinarySearchWrong {
3
4    /** Find the index of value "search" in the sorted array "array".
5     * @param array the array to search inside, must be sorted
6     * @param search the value to search
7     * @return the index of search, or -1 if it does not occur in "array"
8     */
9    static int binarySearch(int[] array, int search) {
10     int lowerBound = 0; // the index of
11     int upperBound = array.length - 1; /
12
13     while (lowerBound <= upperBound) { /
14       int midIndex = (lowerBound + upper
15
16       if (array[midIndex] < search) { //
17         lowerBound = midIndex + 1; // se
18       } else { // otherwise
19         if (array[midIndex] > search) {
20           upperBound = midIndex - 1; // search only below the middle from now on
21         } else { // ok, element is neither smaller or bigger, so it must be equal
22           return midIndex; // then array[midIndex] == search must hold
23         }
24       }
25     }
26     return -1; // we did not find the element
27   }
28 }
29
```

but

`(lowerBound<=upperBound)`

Problems | Javadoc | Declaration | Console | Progress | Debug

```
<terminated> BinarySearchWrongTest [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Feb
index of 0: 0
```

Writable | Smart Insert | 13 : 23

Fixed Implementation of Binary Search

Listing: Your BugFix of your Colleague's Implementation of Binary Search

```
/** Examples for using System.in and Scanner */
public class BinarySearchRight {

  /** Find the index of value "search" in the sorted array "array".
   * @param array the array to search inside, must be sorted
   * @param search the value to search
   * @return the index of search, or -1 if it does not occur in "array"
   */
  static int binarySearch(int[] array, int search) {
    int lowerBound = 0; // the index of the first array element
    int upperBound = array.length - 1; // the index of the last array element

    while (lowerBound <= upperBound) { // as long as current division is not empty
      int midIndex = (lowerBound + upperBound) / 2; // compute mid index

      if (array[midIndex] < search) { // if element in middle is smaller than search
        lowerBound = midIndex + 1; // search only above the middle from now on
      } else { // otherwise
        if (array[midIndex] > search) { // if element is bigger than search
          upperBound = midIndex - 1; // search only below the middle from now on
        } else { // ok, element is neither smaller or bigger, so it must be equal
          return midIndex; // then array[midIndex] == search must hold
        }
      }
    }
    return -1; // we did not find the element
  }
}
```

Listing: Your Test of the fixed Implementation of Binary Search

```
/** A program testing BinarySearchRight. */
public class BinarySearchRightTest {

  /** The main routine
   * @param args
   *          we ignore this parameter */
  public static final void main(String[] args) {

    int[] array = {0, 1, 2, 3, 5, 6};

    System.out.println("index␣of␣0:␣" + BinarySearchRight.binarySearch(array, 0)); //$NON-NLS-1$
    System.out.println("index␣of␣1:␣" + BinarySearchRight.binarySearch(array, 1)); //$NON-NLS-1$
    System.out.println("index␣of␣2:␣" + BinarySearchRight.binarySearch(array, 2)); //$NON-NLS-1$
    System.out.println("index␣of␣3:␣" + BinarySearchRight.binarySearch(array, 3)); //$NON-NLS-1$
    System.out.println("index␣of␣5:␣" + BinarySearchRight.binarySearch(array, 5)); //$NON-NLS-1$
    System.out.println("index␣of␣6:␣" + BinarySearchRight.binarySearch(array, 6)); //$NON-NLS-1$

    System.out.println("index␣of␣-1:␣" + BinarySearchRight.binarySearch(array, -1)); //$NON-NLS-1$
    System.out.println("index␣of␣4:␣" + BinarySearchRight.binarySearch(array, 4)); //$NON-NLS-1$
    System.out.println("index␣of␣7:␣" + BinarySearchRight.binarySearch(array, 7)); //$NON-NLS-1$
  }
}
```

The Output

Listing: The expected output

```
index of 0: 0
index of 1: 1
index of 2: 2
index of 3: 3
index of 5: 4
index of 6: 5
index of -1: -1
index of 4: -1
index of 7: -1
```

Listing: The actual output

```
index of 0: 0
index of 1: 1
index of 2: 2
index of 3: 3
index of 5: 4
index of 6: 5
index of -1: -1
index of 4: -1
index of 7: -1
```

Summary

- We have learned what debugging is.
- We have learned how to use a debugger.
- We have learned about break points, stepping into functions, step-by-step program execution, checking variables, and checking expressions
- We have learned how to find known errors in programs and fix them
- We also have learned that *testing* is important, because if we do not know that there is an error, we cannot find it
- If we had not tested `binarySearch`, maybe we would have shipped a wrong program...

谢谢

Thank you

Thomas Weise [汤卫思]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui,
China

Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog