Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar, Gujarat

Name : Ghori Zeel Jivrajbhai

Student ID : 202201287

Course : IT 314 Software Engineering

Professor : Saurabh Tiwari

Semester : Autumn 2024

Lab : Program Inspection, Debugging and Static Analysis

Merge Sort

Given code :

```
// This program implements the merge sort algorithm for
// arrays of integers.

import java.util.*;

public class MergeSort {
   public static void main(String[] args) {
      int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
      System.out.println("before: " + Arrays.toString(list));
      mergeSort(list);
      System.out.println("after:  " + Arrays.toString(list));
   }

   // Places the elements of the given array into sorted order
   // using the merge sort algorithm.
   // post: array is in sorted (nondecreasing) order
   public static void mergeSort(int[] array) {
      if (array.length > 1) {
         // split array into two halves
         int[] left = leftHalf(array+1);
         int[] right = rightHalf(array-1);

         // recursively sort the two halves
         mergeSort(left);
         mergeSort(right);

         // merge the sorted halves into a sorted whole
```

```
        merge(array, left++, right--);
    }
}

// Returns the first half of the given array.
public static int[] leftHalf(int[] array) {
    int size1 = array.length / 2;
    int[] left = new int[size1];
    for (int i = 0; i < size1; i++) {
        left[i] = array[i];
    }
    return left;
}

// Returns the second half of the given array.
public static int[] rightHalf(int[] array) {
    int size1 = array.length / 2;
    int size2 = array.length - size1;
    int[] right = new int[size2];
    for (int i = 0; i < size2; i++) {
        right[i] = array[i + size1];
    }
    return right;
}

// Merges the given left and right arrays into the given
// result array.  Second, working version.
// pre : result is empty; left/right are sorted
// post: result contains result of merging sorted lists;
public static void merge(int[] result,
                         int[] left, int[] right) {
```

```
        int i1 = 0;   // index into left array
        int i2 = 0;   // index into right array

        for (int i = 0; i < result.length; i++) {
          if (i2 >= right.length || (i1 < left.length &&
                left[i1] <= right[i2])) {
            result[i] = left[i1];    // take from left
            i1++;
          } else {
            result[i] = right[i2];   // take from right
            i2++;
          }
        }
      }
    }
```

Input: before 14 32 67 76 23 41 58 85

after 14 23 32 41 58 67 76 85

Program Inspection for Merge Sort

1. **How many errors are there in the program? Mention the errors you have identified.**

- **Errors Identified: 3**
    - **Incorrect Array Manipulation:**
        - In the mergeSort method, the lines int[] left = leftHalf(array+1); and int[] right = rightHalf(array-1); are incorrect. You cannot add or subtract from an array directly in Java. Instead, the code should pass a subarray of the original array to these methods.
    - **Incorrect Merge Method Call:**
        - The line merge(array, left++, right--); contains incorrect usage of the ++ and -- operators. These operators cannot be used on array references in this context. Instead, it should just be merge(array, left, right);.

- **Merge Method Logic:**
    - The merge function should operate on a temporary array (the result array) to hold the merged values. The merge function is defined correctly, but it's not utilized properly in the mergeSort method.

2. Which category of program inspection would you find more effective?

- **Effective Category:**
    - **Control-Flow Errors**: This category is particularly effective here since it deals with the logical flow of the program, especially in how the recursive calls and array handling are managed.

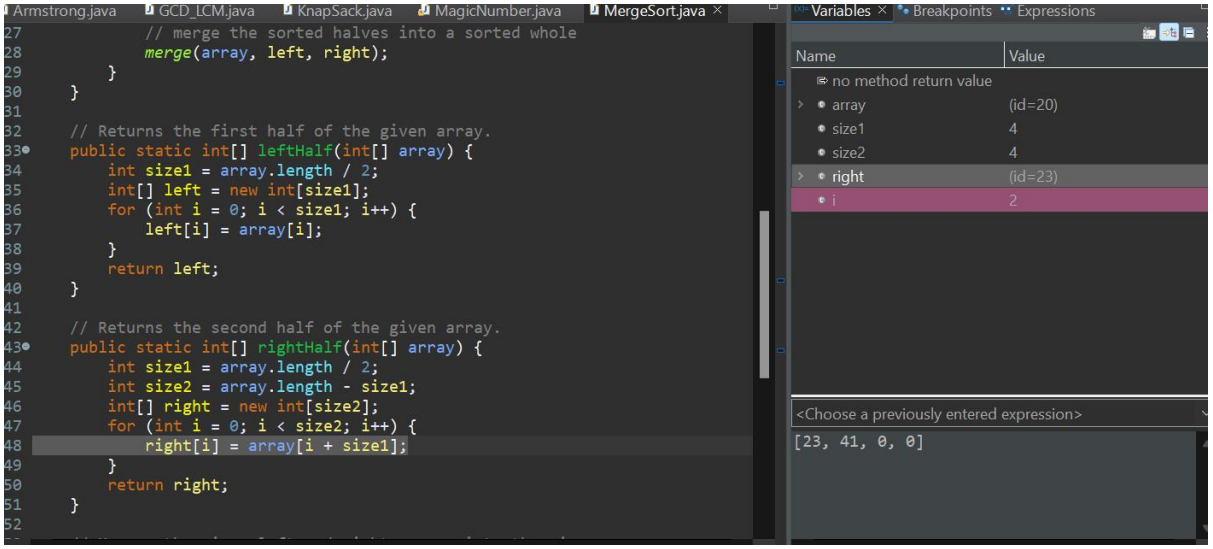
3. Which type of error you are not able to identify using the program inspection?

- **Errors Not Identified:**
    - **Semantic Errors in Array Handling**: Program inspections can miss semantic errors like those arising from incorrect manipulation or interpretation of array indices, especially when it comes to passing array sections or creating subarrays.

4. Is the program inspection technique worth applicable?

- **Applicability of Program Inspection:**
    - Yes, the program inspection technique is worth applying. It aids in identifying structural issues and potential run-time exceptions, which ultimately contributes to higher code quality. However, it's important to supplement it with other testing methods to catch logical and semantic errors.

Code Debugging



```
27            // merge the sorted halves into a sorted whole
28            merge(array, left, right);
29        }
30    }
31
32    // Returns the first half of the given array.
33    public static int[] leftHalf(int[] array) {
34        int size1 = array.length / 2;
35        int[] left = new int[size1];
36        for (int i = 0; i < size1; i++) {
37            left[i] = array[i];
38        }
39        return left;
40    }
41
42    // Returns the second half of the given array.
43    public static int[] rightHalf(int[] array) {
44        int size1 = array.length / 2;
45        int size2 = array.length - size1;
46        int[] right = new int[size2];
47        for (int i = 0; i < size2; i++) {
48            right[i] = array[i + size1];
49        }
50        return right;
51    }
52
```

```
51    }
52
53    // Merges the given left and right arrays into the given
54    // result array.  Second, working version.
55    // pre : result is empty; left/right are sorted
56    // post: result contains result of merging sorted lists;
57    public static void merge(int[] result,
58                             int[] left, int[] right) {
59        int i1 = 0;   // index into left array
60        int i2 = 0;   // index into right array
61
62        for (int i = 0; i < result.length; i++) {
63            if (i2 >= right.length || (i1 < left.length &&
64                    left[i1] <= right[i2])) {
65                result[i] = left[i1];    // take from left
66                i1++;
67            } else {
68                result[i] = right[i2];   // take from right
69                i2++;
70            }
71        }
72    }
73 }
74
```

```
Console × Problems Debug Shell
<terminated> MergeSort [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20 Oct 2024, 6:27:47 pm – 6:27:47 pm) [pid: 22200
before: [14, 32, 67, 76, 23, 41, 58, 85]
after:  [14, 23, 32, 41, 58, 67, 76, 85]
```

Errors Identified

1. **Incorrect Method Calls for Splitting the Array:**
    - **Original Line:** int[] left = leftHalf(array + 1);
    - **Correction:** Change to int[] left = leftHalf(array); (You should pass the entire array without modifying it.)
    - **Original Line:** int[] right = rightHalf(array - 1);

- **Correction:** Change to int[] right = rightHalf(array); (Again, pass the entire array.)

2. **Invalid Merge Function Call:**
    - **Original Line:** merge(array, left++, right--);
    - **Correction:** Change to merge(array, left, right); (Post-increment and post-decrement do not apply to arrays.)

Breakpoints Needed

You can set breakpoints at the following lines for effective debugging:

- **Line 15:** To check how the left array is created.
- **Line 16:** To check how the right array is created.
- **Line 21:** To verify if the merge is done correctly.

Steps to Fix the Errors

1. **Change the method calls** to leftHalf(array) and rightHalf(array) to avoid modifying the input array.
2. **Update the merge function call** to merge(array, left, right); instead of using post-increment and post-decrement.

Fixed Code

```
// This program implements the merge sort algorithm for arrays of integers.
import java.util.*;
public class MergeSort {
  public static void main(String[] args) {
    int[] list = {14, 32, 67, 76, 23, 41, 58, 85};
    System.out.println("before: " + Arrays.toString(list));
    mergeSort(list);
    System.out.println("after: " + Arrays.toString(list));
  }

  // Places the elements of the given array into sorted order using the merge sort algorithm.
  // post: array is in sorted (nondecreasing) order
  public static void mergeSort(int[] array) {
    if (array.length > 1) {
      // split array into two halves
      int[] left = leftHalf(array); // Fixed
```

```
        int[] right = rightHalf(array); // Fixed
        // recursively sort the two halves
        mergeSort(left);
        mergeSort(right);
        // merge the sorted halves into a sorted whole
        merge(array, left, right); // Fixed
    }
}

// Returns the first half of the given array.
public static int[] leftHalf(int[] array) {
    int size1 = array.length / 2;
    int[] left = new int[size1];
    for (int i = 0; i < size1; i++) {
        left[i] = array[i];
    }
    return left;
}

// Returns the second half of the given array.
public static int[] rightHalf(int[] array) {
    int size1 = array.length / 2;
    int size2 = array.length - size1;
    int[] right = new int[size2];
    for (int i = 0; i < size2; i++) {
        right[i] = array[i + size1];
    }
    return right;
}

// Merges the given left and right arrays into the given result array.
```

```
  // pre : result is empty; left/right are sorted
  // post: result contains result of merging sorted lists
  public static void merge(int[] result, int[] left, int[] right) {
    int i1 = 0; // index into left array
    int i2 = 0; // index into right array
    for (int i = 0; i < result.length; i++) {
      if (i2 >= right.length || (i1 < left.length && left[i1] <= right[i2])) {
        result[i] = left[i1]; // take from left
        i1++;
      } else {
        result[i] = right[i2]; // take from right
        i2++;
      }
    }
  }
}
```

Input and Output

- **Input:**
    - Before sorting: {14, 32, 67, 76, 23, 41, 58, 85}
- **Output:**

After sorting: {14, 23, 32, 41, 58, 67, 76, 85}

Static Analysis Tools

Choose a static analysis tool (in Java, Python, C, C++) in any programming language of your interest and identify the defects. You can also choose your own code fragment from GitHub (more than 2000 LOC) in any programming language to perform static analysis. Submit your results in the .xls or .jpg format only.

Link of the excel file(.xls):

https://docs.google.com/spreadsheets/d/1ETFxQdw1_RNXufBPwhlt7OumxA7lJCjh/edit?usp=sharing&ouid=117105520491645507929&rtpof=true&sd=true