

---

Software Testing Laboratory  
(CS6474)  
Assignment 07 :FindBugs

---

**Tapas Manna**  
**223CS3152**  
Master of Technology  
223cs3152@nitrkl.ac.in

**Department of Computer Science and Engineering**  
**NIT, Rourkela**

April 7, 2024

## Contents

<b>1 Find Bugs</b>	<b>3</b>
1.1 Write a program to generate a Factorial of numbers. . . . .	3
1.2 Write a program to generate Fibonacci numbers. . . . .	3
1.3 Write a program that performs sorting of a group of integer values using the quick sort technique. . . . .	4
1.4 Write a program that accepts elements of a matrix and displays its transpose. . . . .	4
1.5 Write a program to add two matrices and display the sum matrix . . . . .	5
1.6 Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop. . . . .	5
1.7 Write a program to generate a palindrome of numbers. . . . .	6
1.8 Write a program to find out the sum of two arrays. . . . .	6
1.9 Write a program to check whether the number is even or odd. . . . .	7
1.10 Write a program for binary to hexadecimal conversion. . . . .	7

# 1 Find Bugs

## 1.1 Write a program to generate a Factorial of numbers.

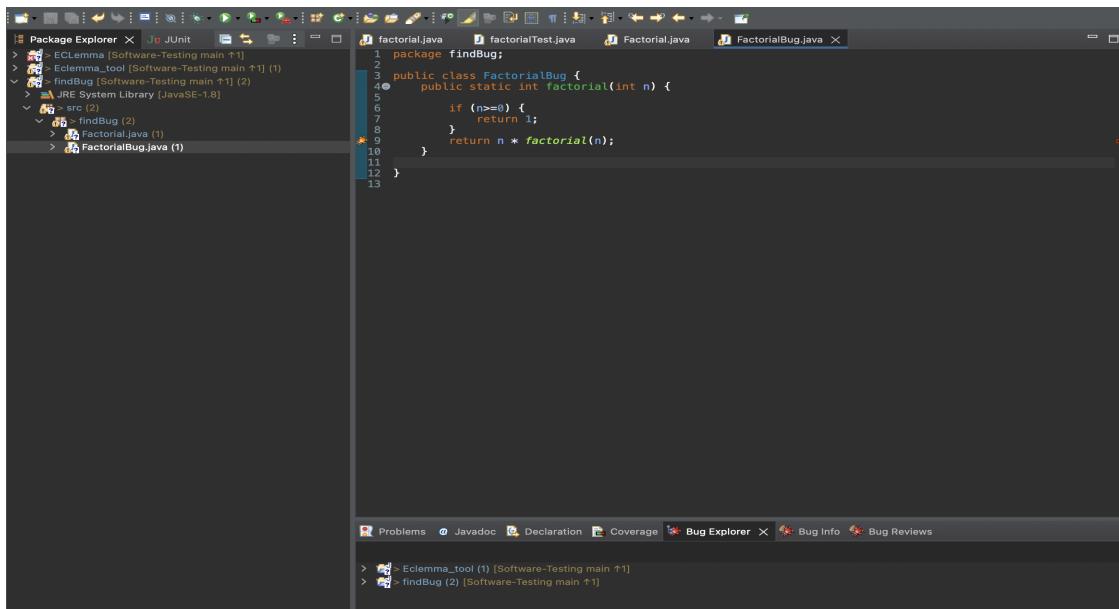


Figure 1: Findbug Screenshot

## 1.2 Write a program to generate Fibonacci numbers.

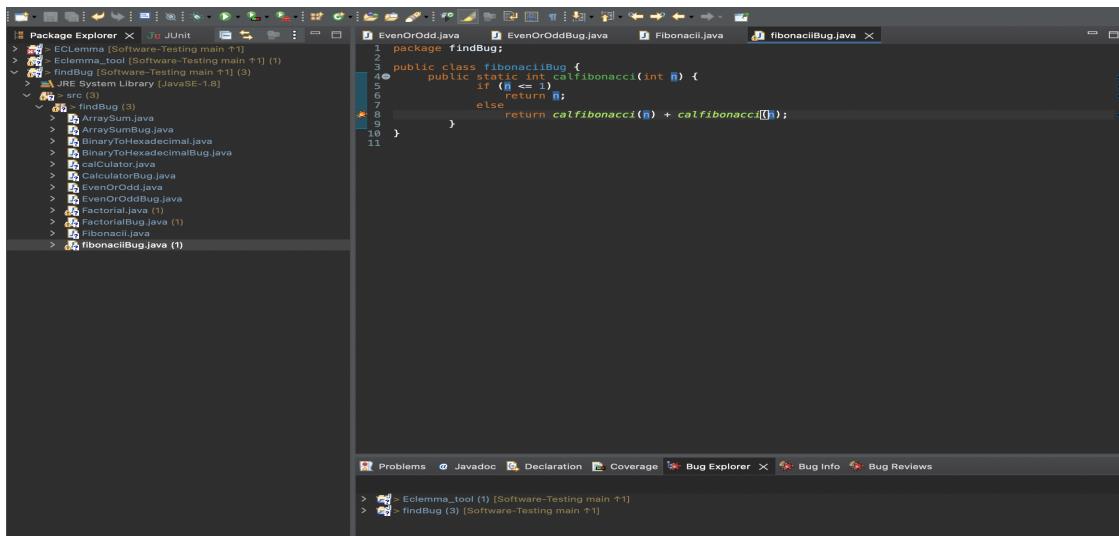
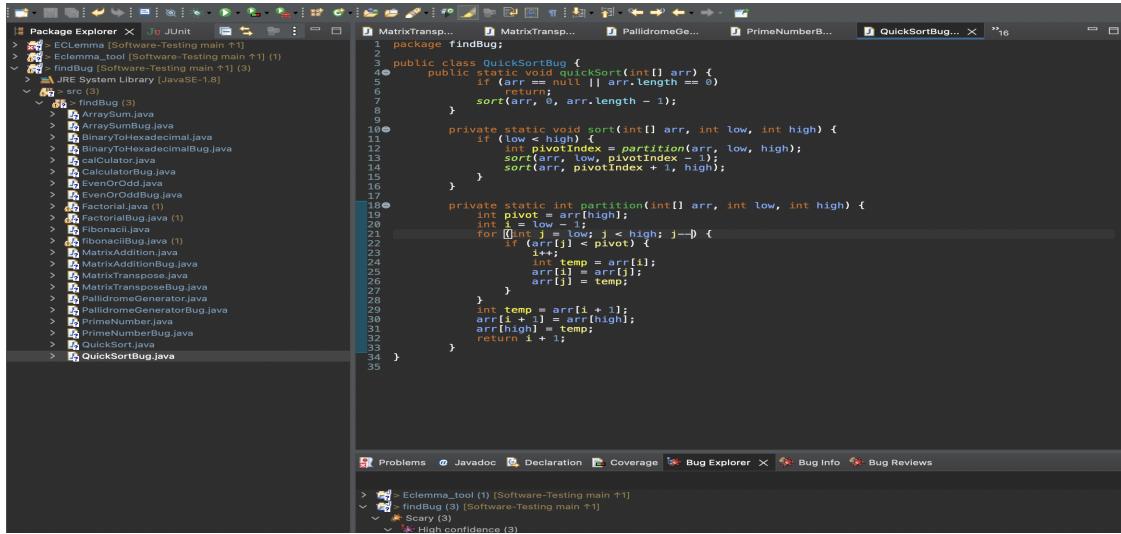


Figure 2: Findbug Screenshot

### 1.3 Write a program that performs sorting of a group of integer values using the quick sort technique.



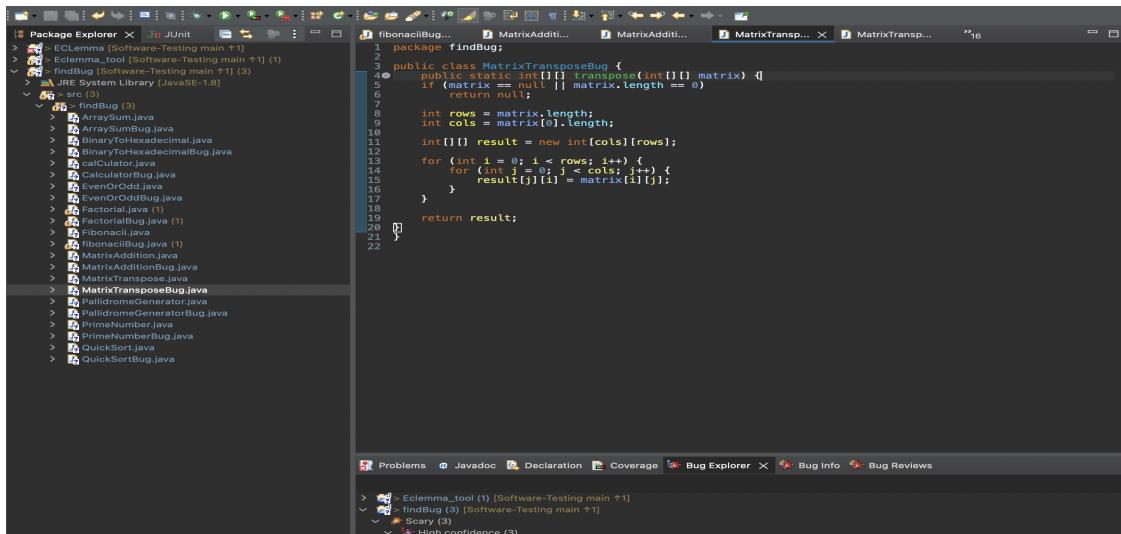
```

1 package findBug;
2
3 public class QuickSortBug {
4     public static void quickSort(int[] arr) {
5         if (arr == null || arr.length == 0)
6             return;
7         sort(arr, 0, arr.length - 1);
8     }
9
10    private static void sort(int[] arr, int low, int high) {
11        if (low > high)
12            int pivotIndex = partition(arr, low, high);
13            sort(arr, low, pivotIndex - 1);
14            sort(arr, pivotIndex + 1, high);
15    }
16
17    private static int partition(int[] arr, int low, int high) {
18        int pivot = arr[high];
19        int i = low; j = high;
20        for (; i < j; j--) {
21            if (arr[j] < pivot) {
22                int temp = arr[i];
23                arr[i] = arr[j];
24                arr[j] = temp;
25            }
26        }
27        int temp = arr[i + 1];
28        arr[i + 1] = arr[high];
29        arr[high] = temp;
30        return i + 1;
31    }
32
33    }
34 }

```

Figure 3: Findbug Screenshot

### 1.4 Write a program that accepts elements of a matrix and displays its transpose.



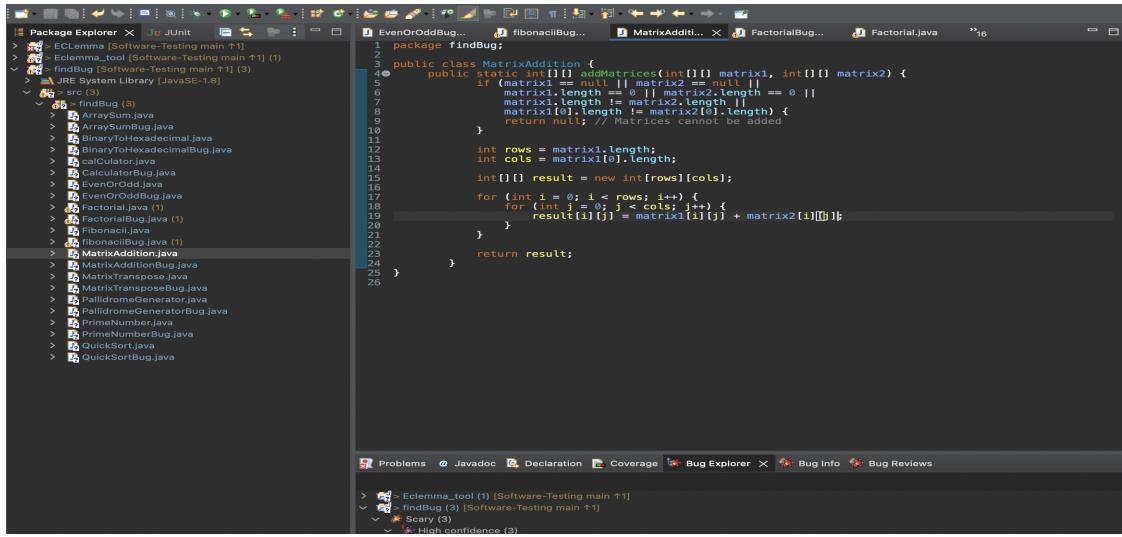
```

1 package findBug;
2
3 public class MatrixTransposeBug {
4     public static int[][] transpose(int[][] matrix) {
5         if (matrix == null || matrix.length == 0)
6             return null;
7
8         int rows = matrix.length;
9         int cols = matrix[0].length;
10
11        int[][] result = new int[cols][rows];
12
13        for (int i = 0; i < rows; i++) {
14            for (int j = 0; j < cols; j++) {
15                result[j][i] = matrix[i][j];
16            }
17        }
18
19        return result;
20    }
21
22

```

Figure 4: Findbug Screenshot

## 1.5 Write a program to add two matrices and display the sum matrix

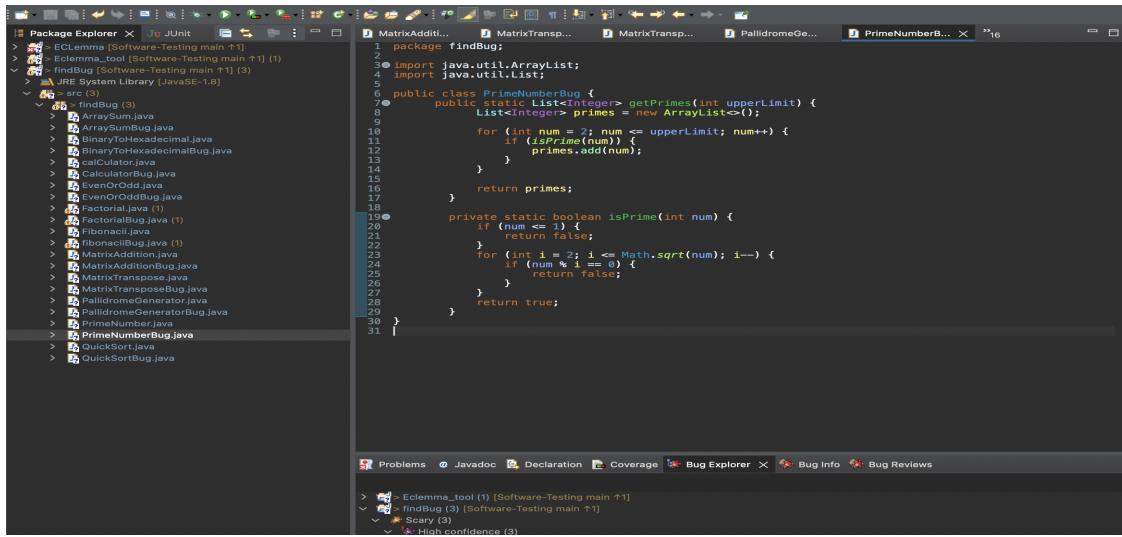


The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view lists several Java projects and source files, including `MatrixAddition.java` and `MatrixAdditionBug.java`. The central workspace shows the Java code for `MatrixAddition`. The code defines a static method `addMatrices` that takes two `int[][]` matrices as parameters and returns a new matrix representing their sum. The code includes checks for null matrices and matrix dimensions. The right side of the interface shows the FindBugs analysis results in the Bug Explorer view, indicating 3 bugs: 1 Scary and 2 High confidence.

```
1 package findBug;
2
3 public class MatrixAddition {
4     public static int[][] addMatrices(int[][] matrix1, int[][] matrix2) {
5         if (matrix1 == null || matrix2 == null || matrix1.length == 0 || matrix1[0].length != matrix2.length || matrix1[0].length != matrix2[0].length) {
6             return null; // Matrices cannot be added
7         }
8         int rows = matrix1.length;
9         int cols = matrix1[0].length;
10
11        int[][] result = new int[rows][cols];
12
13        for (int i = 0; i < rows; i++) {
14            for (int j = 0; j < cols; j++) {
15                result[i][j] = matrix1[i][j] + matrix2[i][j];
16            }
17        }
18        return result;
19    }
20 }
```

Figure 5: Findbug Screenshot

## 1.6 Write a program to Print Prime Numbers from 1 to 100 using Scanner Class and For Loop.

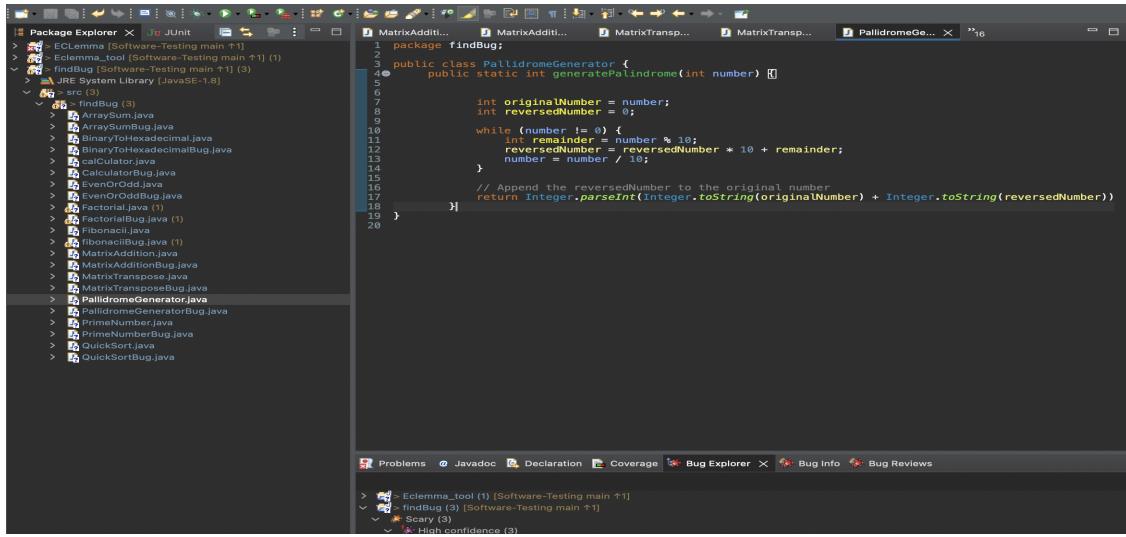


The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view lists several Java projects and source files, including `PrimeNumberBug.java`. The central workspace shows the Java code for `PrimeNumberBug`. The code defines a static method `getPrimes` that takes an `int` `upperLimit` and returns a `List<Integer>` of prime numbers. The code includes a helper method `isPrime` that checks if a number is prime using a for loop and the `Math.sqrt` method. The right side of the interface shows the FindBugs analysis results in the Bug Explorer view, indicating 3 bugs: 1 Scary and 2 High confidence.

```
1 package findbug;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class PrimeNumberBug {
7     public static List<Integer> getPrimes(int upperLimit) {
8         List<Integer> primes = new ArrayList<>();
9
10        for (int num = 2; num <= upperLimit; num++) {
11            if (isPrime(num)) {
12                primes.add(num);
13            }
14        }
15
16        return primes;
17    }
18
19    private static boolean isPrime(int num) {
20        if (num <= 1) {
21            return false;
22        }
23        for (int i = 2; i <= Math.sqrt(num); i++) {
24            if (num % i == 0) {
25                return false;
26            }
27        }
28        return true;
29    }
30 }
```

Figure 6: Findbug Screenshot

## 1.7 Write a program to generate a palindrome of numbers.



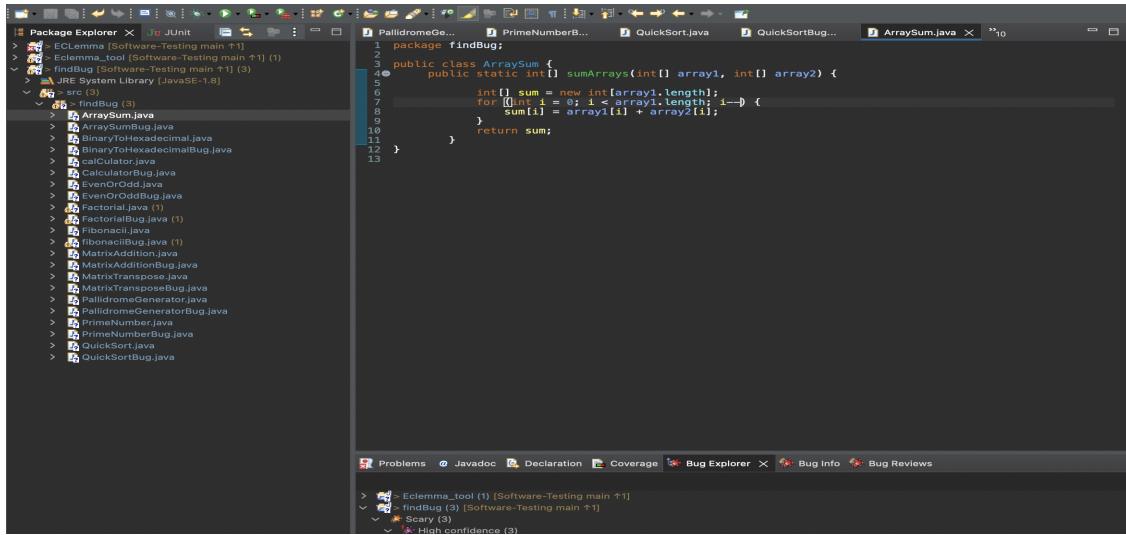
The screenshot shows the Eclipse IDE interface. The left pane is the Package Explorer, displaying a project structure with several Java files under a 'findBug' package. The right pane is the code editor, showing the following Java code:

```
1 package findBug;
2
3 public class PalindromeGenerator {
4     public static int generatePalindrome(int number) {
5
6         int originalNumber = number;
7         int reversedNumber = 0;
8
9         while (number != 0) {
10             int remainder = number % 10;
11             reversedNumber = reversedNumber * 10 + remainder;
12             number = number / 10;
13         }
14
15         // Append the reversedNumber to the original number
16         return Integer.parseInt(Integer.toString(originalNumber) + Integer.toString(reversedNumber));
17     }
18 }
19
20
```

Below the code editor is the 'Bug Explorer' view, which lists a single bug entry: 'Scary (3)' under the 'findBug' package, with 'High confidence (3)'.

Figure 7: Findbug Screenshot

## 1.8 Write a program to find out the sum of two arrays.



The screenshot shows the Eclipse IDE interface. The left pane is the Package Explorer, displaying a project structure with several Java files under a 'findBug' package. The right pane is the code editor, showing the following Java code:

```
1 package findBug;
2
3 public class ArraySum {
4     public static int[] sumArrays(int[] array1, int[] array2) {
5
6         int[] sum = new int[array1.length];
7         for (int i = 0; i < array1.length; i--) {
8             sum[i] = array1[i] + array2[i];
9         }
10     }
11 }
12
13
```

Below the code editor is the 'Bug Explorer' view, which lists a single bug entry: 'Scary (3)' under the 'findBug' package, with 'High confidence (3)'.

Figure 8: Findbug Screenshot

## 1.9 Write a program to check whether the number is even or odd.

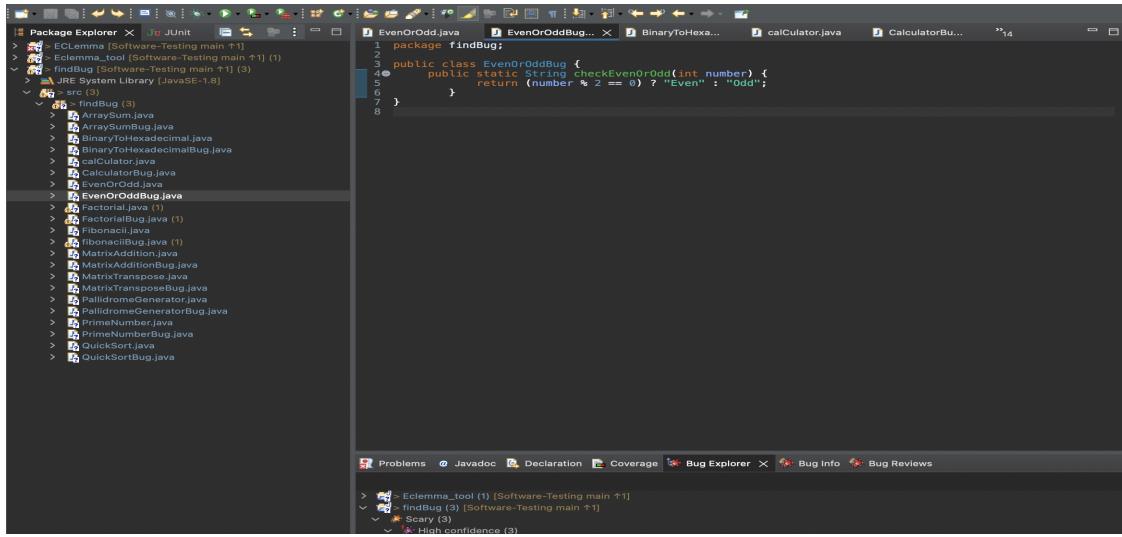


Figure 9: Findbug Screenshot

## 1.10 Write a program for binary to hexadecimal conversion.

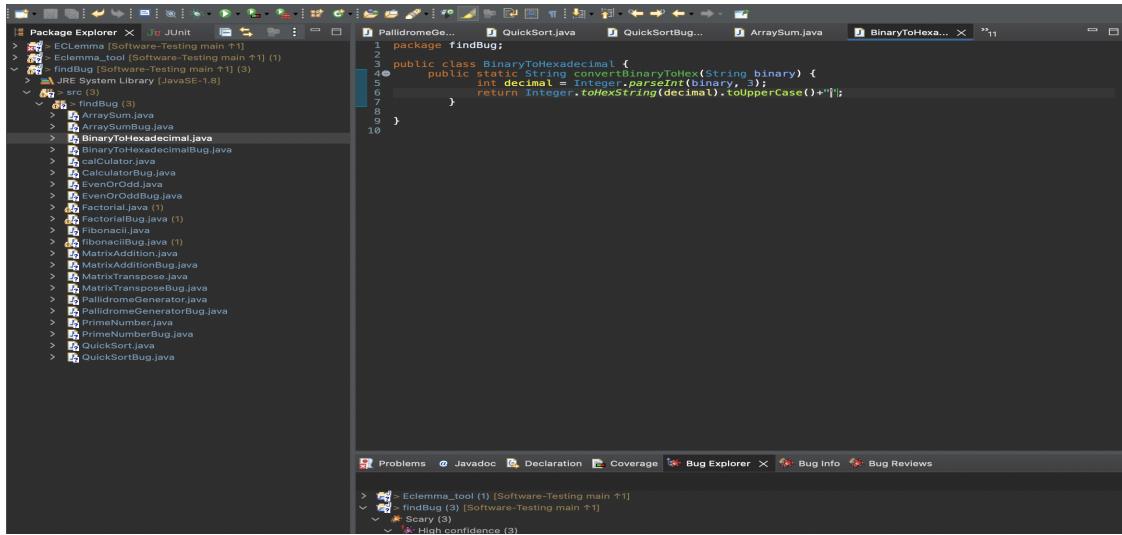


Figure 10: Findbug Screenshot