

```
public static String compute(int[] numbers) {  
    double[] result = new double[numbers.length];  
  
    for (int i = 0; i < numbers.length; i++) {  
        if (numbers[i] == 0) {  
            result[i] = 0;  
            continue;  
        }  
  
        if (numbers[i] < 0) {  
            result[i] = Math.sqrt(-1 * numbers[i]);  
        } else {  
            result[i] = Math.sqrt(numbers[i]);  
        }  
    }  
  
    return Arrays.toString(result);  
}
```

The diagram illustrates the control flow of the provided Java code. It uses colored nodes and lines to represent different execution paths:

- Red nodes and lines:** Represent the path where `numbers[i] == 0`. This path starts at the `for` loop, goes to the `if (numbers[i] == 0)` condition, then to `result[i] = 0;`, `continue;`, and loops back to the `for` loop.
- Purple nodes and lines:** Represent the path where `numbers[i] < 0`. This path starts at the `for` loop, goes to the `if (numbers[i] < 0)` condition, then to `result[i] = Math.sqrt(-1 * numbers[i]);`, and loops back to the `for` loop.
- Orange nodes and lines:** Represent the path where `numbers[i] >= 0` and `numbers[i] != 0`. This path starts at the `for` loop, goes to the `if (numbers[i] < 0)` condition, then to the `else` block (`result[i] = Math.sqrt(numbers[i]);`), and loops back to the `for` loop.
- Final return:** After the `for` loop, the code reaches `return Arrays.toString(result);`, which is represented by orange nodes and lines.