

```
public static boolean compute(String word) {  
    boolean result = true;  
    for (int i = 0, j = word.length() - 1; i < word.length() / 2; i++, j--) {  
        if (word.charAt(i) != word.charAt(j)) {  
            result = false;  
            break;  
        }  
    }  
    return result;  
}
```

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

- Black nodes:** Located at the start of the `compute` method and at the end of the `return result;` statement.
- Purple nodes:** Represent the initial state of the `result` variable being set to `true`.
- Orange nodes:** Represent the loop's entry point and the state where the loop continues to the next iteration.
- Yellow nodes:** Represent the state where the loop breaks due to a mismatch in characters.

The flow starts at the black node, moves to the purple nodes for the `result = true;` assignment, then to the orange nodes for the `for` loop. From the orange nodes, the flow branches: one path leads to yellow nodes for the `if` condition and `break` statement, while the other path continues through orange nodes for the next iteration of the `for` loop. Both paths eventually converge back to the black node at the end of the method.