

```
public static boolean containsSubstring(String word, String substring) {  
    boolean containsSubstring = false;  
  
    for (int i = 0; i < word.length(); i++) {  
        for (int j = 0; j < substring.length(); j++) {  
            if (i + j > word.length()) {  
                break;  
            }  
            if (word.charAt(i + j) != substring.charAt(j)) {  
                break;  
            } else {  
                if (j == substring.length() - 1) {  
                    containsSubstring = true;  
                    break;  
                }  
            }  
        }  
    }  
}  
  
return containsSubstring;  
}
```

The diagram illustrates the control flow of the provided Java code. It uses colored circular nodes and lines to represent the execution path. Yellow nodes and lines represent the initial state and the entry into the loops. Red nodes and lines represent the inner loop's logic, including the break condition when the indices exceed the word's length. Purple nodes and lines represent the outer loop's logic, including the final return statement. The flow starts at the function signature, moves to the initialization of 'containsSubstring', then enters the 'for' loops. It follows the inner loop's 'if' conditions, branching to a break or the 'else' block. The 'else' block leads to the final 'return' statement. The flow ends at the closing brace of the function.