

The diagram illustrates the flow of execution for the `removeDoubleCharacters` method. It uses colored circles as nodes and lines as edges to show the sequence of operations and loops.

- Purple Nodes and Lines:** Represent the initial state and the first conditional check.
  - Starts at the top of the method definition.
  - Flows to the `if (str.isEmpty())` condition.
  - One path leads to the `return str;` statement.
  - Another path leads to the `StringBuilder` initialization.
- Orange Nodes and Lines:** Represent the initialization and the start of the loop.
  - Flows to `StringBuilder result = new StringBuilder();`.
  - Flows to `char prev = str.charAt(0);`.
  - Flows to `result.append(prev);`.
  - Flows to the start of the `for` loop: `for (int i = 1; i < str.length(); i++)`.
- Yellow Nodes and Lines:** Represent the inner loop logic and the final return.
  - Flows to the inner `if` condition: `if (prev != cur)`.
  - One path leads to `result.append(str.charAt(i));`.
  - Another path leads to `prev = cur;`.
  - Both paths merge and flow to the `return result.toString();` statement.
  - Finally, the flow ends at the closing brace of the method.

```
public static String removeDoubleCharacters(String str) {  
    if (str.isEmpty()) {  
        return str;  
    }  
  
    StringBuilder result = new StringBuilder();  
  
    char prev = str.charAt(0);  
    result.append(prev);  
    for (int i = 1; i < str.length(); i++) {  
        char cur = str.charAt(i);  
        if (prev != cur) {  
            result.append(str.charAt(i));  
        }  
        prev = cur;  
    }  
  
    return result.toString();  
}
```