Longest Palindromic Substring

Given a string s, find the longest palindromic substring in s. You may assume that the maximum length of s is 1000.

Example 1:

```
Input: "babad"
Output: "bab"
```

Note: "aba" is also a valid answer.

Example 2:

```
Input: "cbbd"
Output: "bb"
```

Solution 1 in C++

```
#include <iostream>
#include <algorithm>


using namespace std;

class Solution {
public:
    string longestPalindrome(string s) {
        int str_len = s.length();
        string myPalindrome;
        string ret;
        int currentPalindromeLen;

        int str_remaining=str_len;
        for(int i=0; i<str_len; i++)
        {
            myPalindrome="";
            currentPalindromeLen=0;
            for(int j=i; j<str_len; j++)
            {
                myPalindrome += s[j];
                currentPalindromeLen++;
                int idx1=0, idx2=currentPalindromeLen-1;
                while(idx1<idx2)
                {
                    if (myPalindrome[idx1]!=myPalindrome[idx2]) break;
                    idx1++;
                    idx2--;
                }
                if (idx1>=idx2)
                {
                    if (myPalindrome.length()>=ret.length()) ret = myPalindrome;
                }
            }
            str_remaining--;
            if (str_remaining<ret.length()) break;
        }
        return ret;
    }
};

int main(void)
{
    Solution s;

    string testcase1 = "babad"; // the answer should be "bab" or "aba"
    cout << testcase1 << " -> " << s.longestPalindrome(testcase1) << endl;

    string testcase2 = "cbbd"; // the answer should be "bb"
    cout << testcase2 << " -> " << s.longestPalindrome(testcase2) << endl;
}
```

Solution 2 in Java - Time Complexity = O(n^2)

```
public String longestPalindrome(String s) {
    if (s == null || s.length() < 1) return "";
    int start = 0, end = 0;
    for (int i = 0; i < s.length(); i++) {
        int len1 = expandAroundCenter(s, i, i);
        int len2 = expandAroundCenter(s, i, i + 1);
        int len = Math.max(len1, len2);
        if (len > end - start) {
            start = i - (len - 1) / 2;
            end = i + len / 2;
        }
    }
    return s.substring(start, end + 1);
}

private int expandAroundCenter(String s, int left, int right) {
    int L = left, R = right;
    while (L >= 0 && R < s.length() && s.charAt(L) == s.charAt(R)) {
        L--;
        R++;
    }
    return R - L - 1;
}
```