**[Longest Palindromic Substring](https://leetcode.com/problems/longest-palindromic-substring/)**

**public** **class** LongestPalindromicSubstring {

**static** **int** *left* = 0;

**static** **int** *maxLength* = 0;

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

System.***out***.println(*longestPalindrome*("abcbba"));

}

**public** **static** String longestPalindrome(String s) {

**if**(s == **null** || s.length() == 0) {

**return** "";

}

**for**(**int** i = 0 ; i < s.length() ; i++) {

*isPalindrome*(s , i , i);

*isPalindrome*(s , i , i + 1);

}

**return** s.substring(*left* , *left* + *maxLength*);

}

**public** **static** **void** isPalindrome(String s , **int** i , **int** j) {

**while**(i >= 0 && j < s.length() && s.charAt(i) == s.charAt(j)) {

i--;

j++;

}

**if**(j - i - 1 > *maxLength*) {

*maxLength* = j - i - 1;

*left* = i + 1;

}

}

}

Time complexity : O(n^2) , n is length of given word

Space Complexity : O(1) , constant space