**Longest Palindrome**

**Description:**

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

**Code:**

**public** **class** Solution {

**private** **int** lo, maxLen;

**public** [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) longestPalindrome([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) s) {

**int** len = s.length();

**if** (len < 2)

**return** s;

**for** (**int** i = 0; i < len-1; i++) {

            extendPalindrome(s, i, i);  //assume odd length, try to extend Palindrome as possible

            extendPalindrome(s, i, i+1); //assume even length. Center of palindrome null

}

**return** s.substring(lo, lo + maxLen); //The sum inclusive + difference = exclusive

}

**private** **void** extendPalindrome([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) s, **int** j, **int** k) {

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

                        j--; //This cycle checks over the center of palindrome

                        k++;

            }

**if** (maxLen < k - j - 1) {

                        lo = j + 1; //This number is inclusive for the substring;

                        maxLen = k - j - 1; //Number of elements between two borders;

            }

}

/\* A palindrome can have (2\*n)-1 centers

Where n is the length of the palindrome. \*/