Reverse Words in a Given String

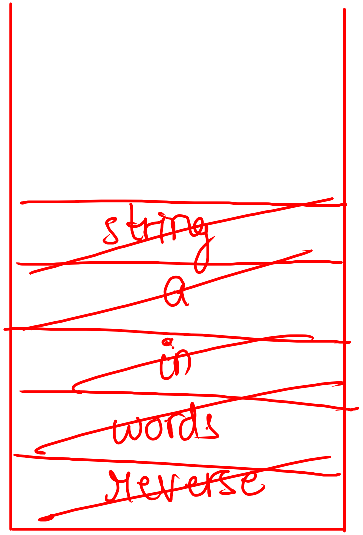
str = "reverse_words_in_a_string" ;

Inbuilt function

String[] arr = str.split("_") ;

arr =

0	1	2	3	4
"reverse"	"words"	"in"	"a"	"string"



ans = ans + st.pop()

0 1 2 0 1

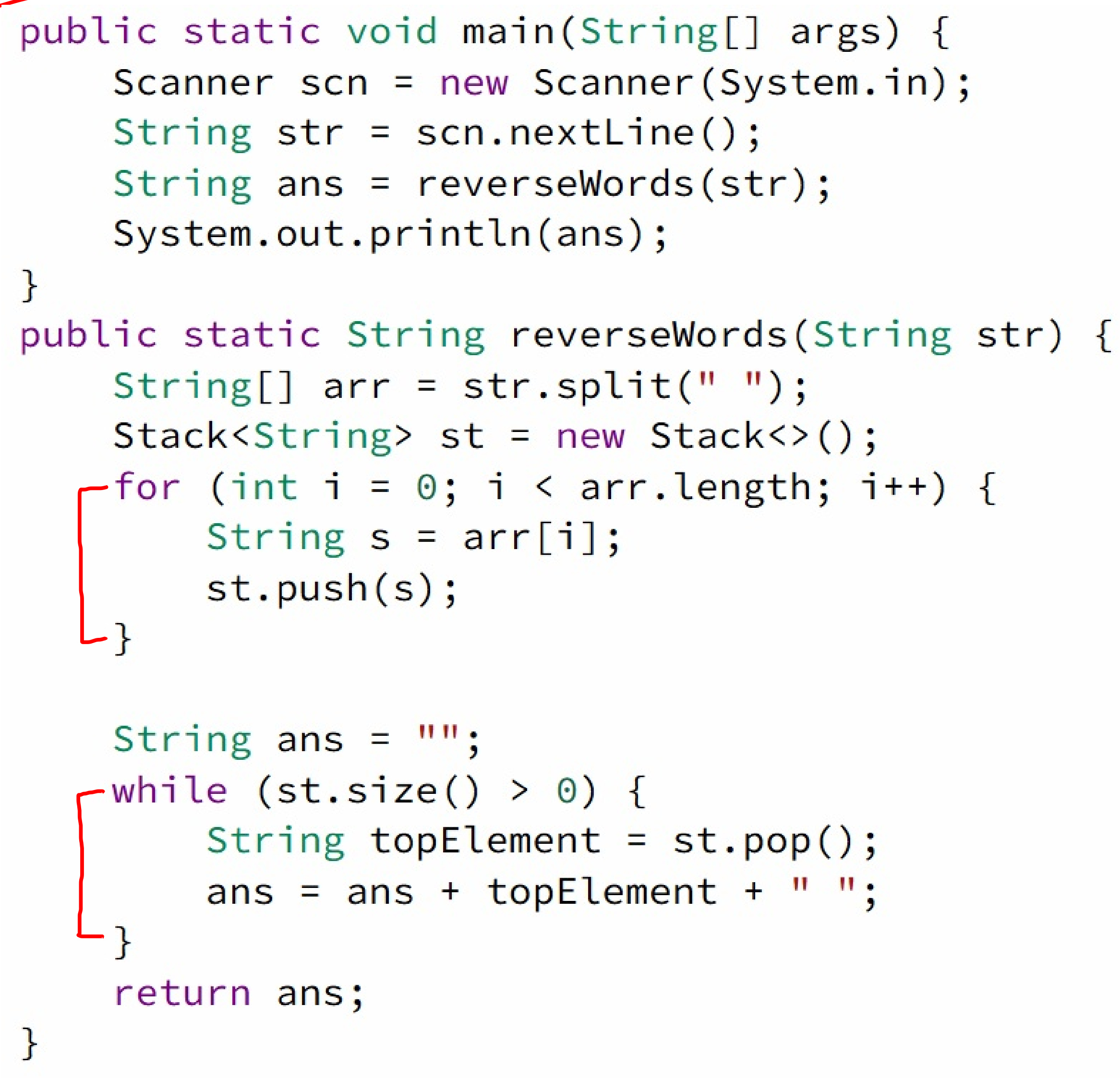
str = "reverse_words_in_a_string";

=> str.split("r");

	0	1	2	3	4
=	""	"eve"	"se wo"	"ds in a st"	"ing"

""

code



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    String ans = reverseWords(str);
    System.out.println(ans);
}
public static String reverseWords(String str) {
    String[] arr = str.split(" ");
    Stack<String> st = new Stack<>();
    for (int i = 0; i < arr.length; i++) {
        String s = arr[i];
        st.push(s);
    }

    String ans = "";
    while (st.size() > 0) {
        String topElement = st.pop();
        ans = ans + topElement + " ";
    }
    return ans;
}
```

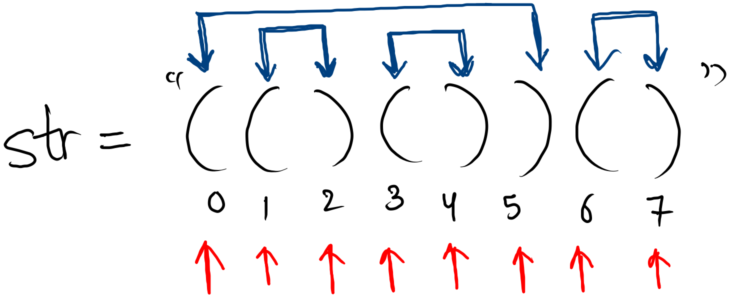
T.C = O(n)

n = stack size

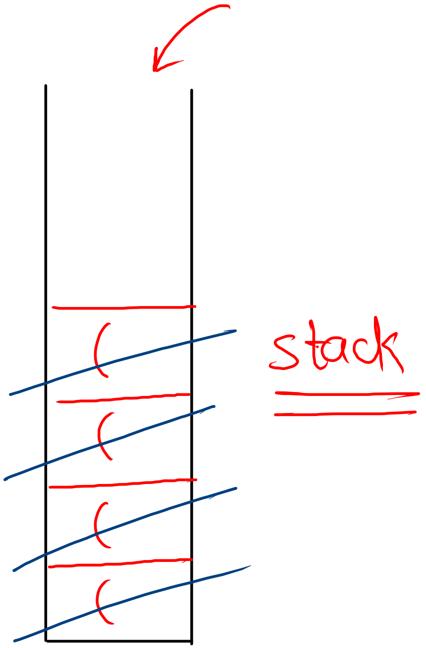
Valid Parentheses (V.V.V. famous)

str = "(()())()"   ans = true

only contains opening & closing para.



faith:- our stack will contain only invalid para. at the end



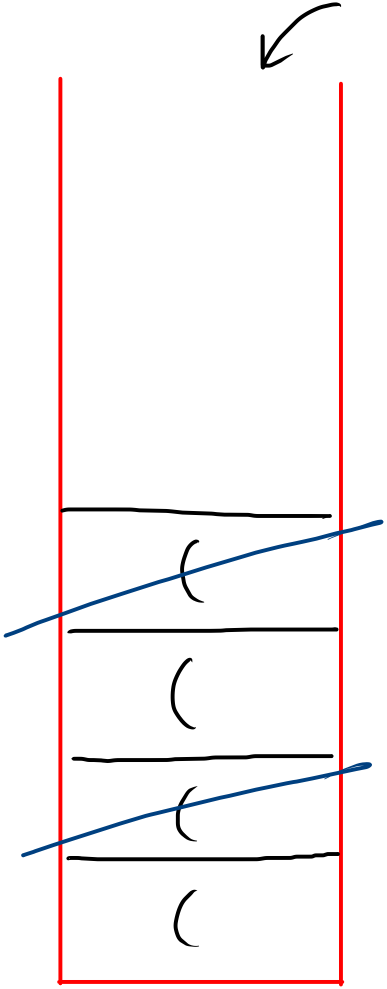
str = "))) ((("

↑ ↑ ↑ ↑ ↑ ↑

str = " (( ) (( ) "

0 1 2 3 4 5

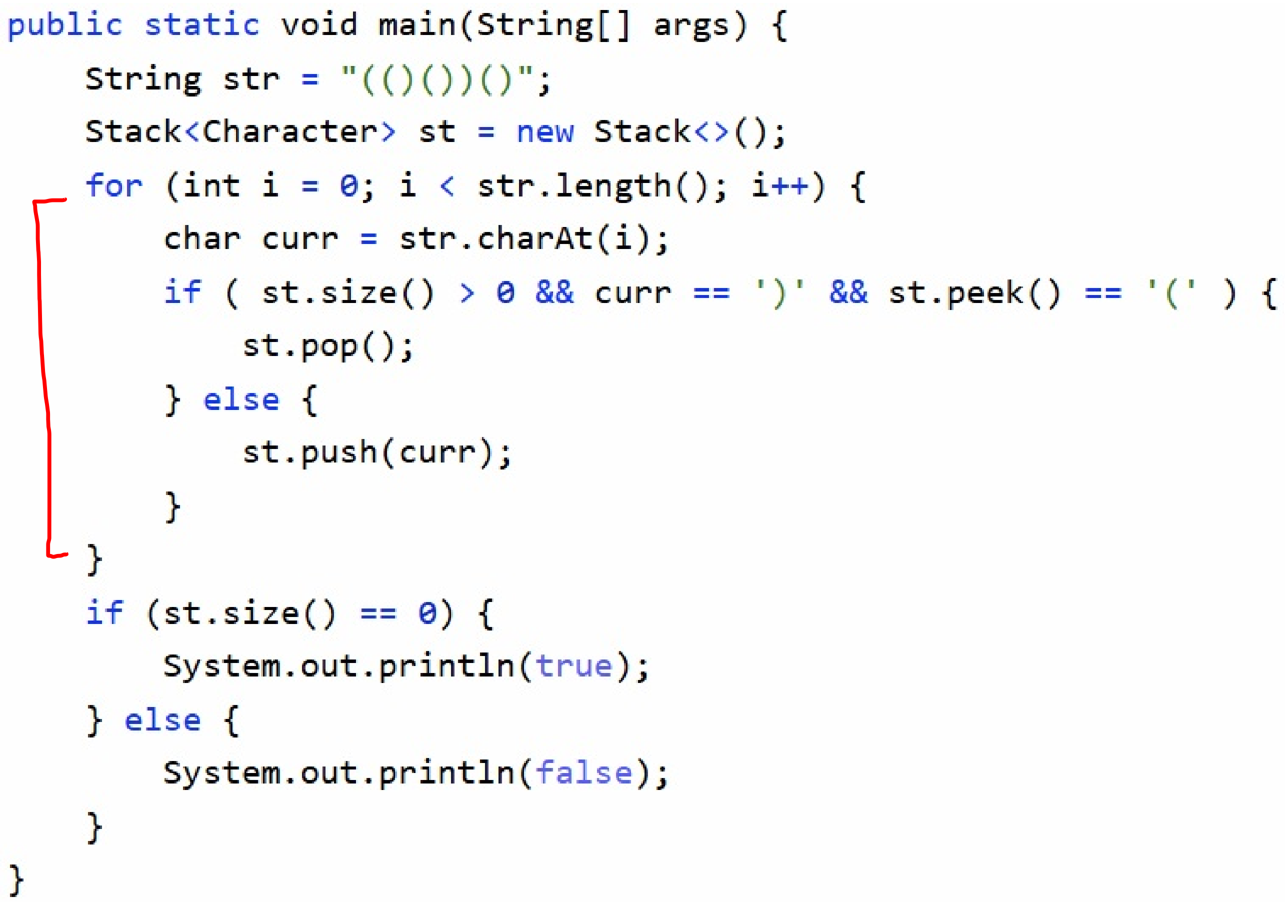
↑ ↑ ↑ ↑ ↑ ↑



psudo code

```
1) Stack
2) traverse over string
    2.1) fetch each character
    2.2) check if   ch == '('
            push
    2.3) else   ch == ')' && st.peek == '('
            pop
3) if st.size() == 0     retue true
   else                  return false
```

code



```
public static void main(String[] args) {
    String str = "(()())()";
    Stack<Character> st = new Stack<>();
    for (int i = 0; i < str.length(); i++) {
        char curr = str.charAt(i);
        if ( st.size() > 0 && curr == ')' && st.peek() == '(' ) {
            st.pop();
        } else {
            st.push(curr);
        }
    }
    if (st.size() == 0) {
        System.out.println(true);
    } else {
        System.out.println(false);
    }
}
```

Valid Parentheses 4

Ex1 str = "([ ]) { }"

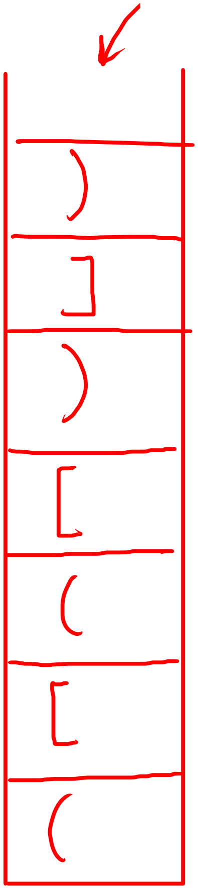
0 1 2 3 4 5

↑ ↑ ↑ ↑ ↑ ↑

Ex2 str = "([([ )])"

0 1 2 3 4 5 6

↑ ↑ ↑ ↑ ↑ ↑ ↑



→ current element

false

faith:- stack will contain only invalid brackets at the end

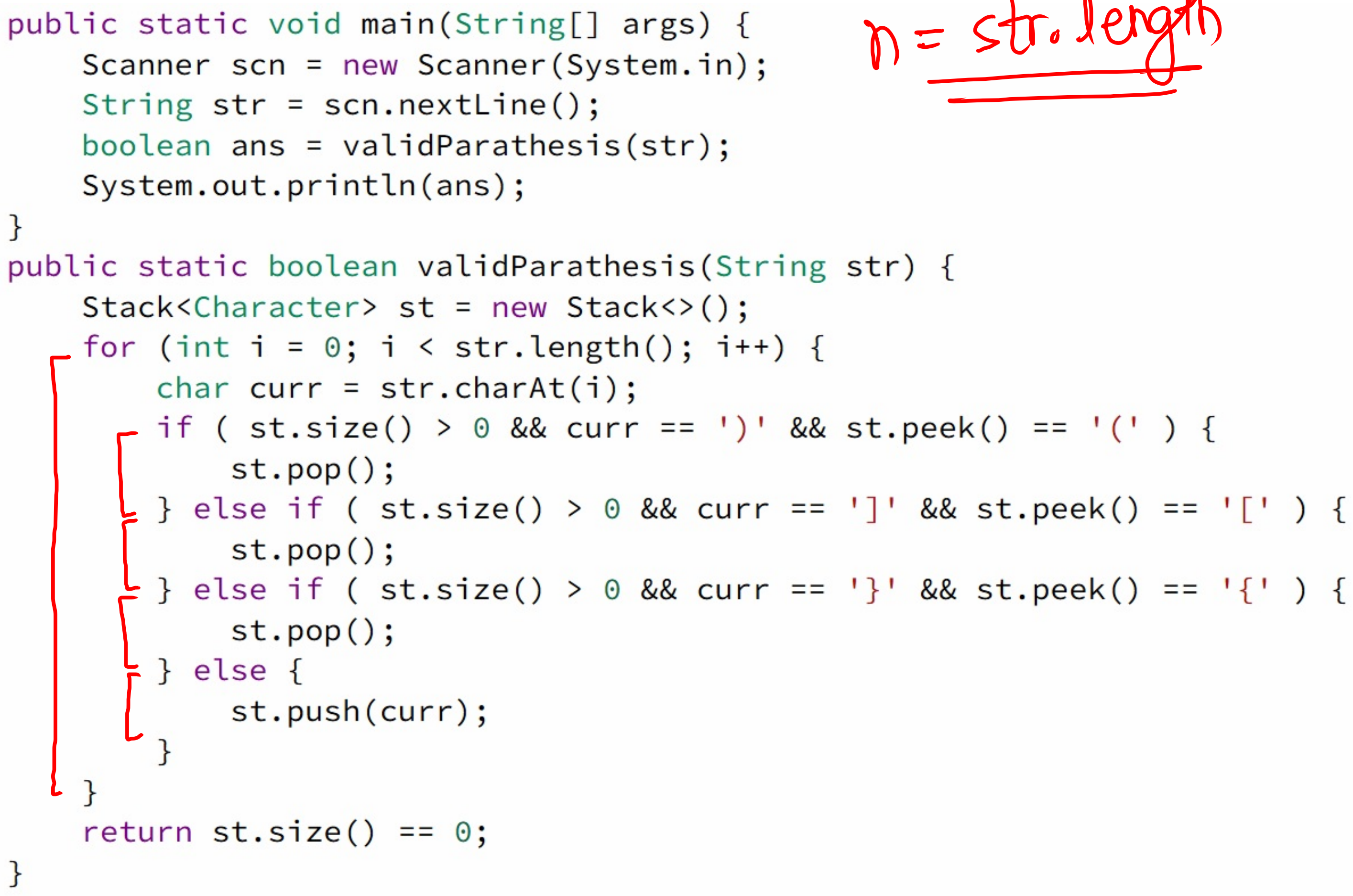
observation

```
1) curr == ']' && peek == '['
        pop
2) curr == ')' && peek == '('
        pop
3) curr == '}' && peek == '{'
        pop
4) else
        push blindly
```

code

T.C = O(n)

n = str.length



```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();
    boolean ans = validParathesis(str);
    System.out.println(ans);
}
public static boolean validParathesis(String str) {
    Stack<Character> st = new Stack<>();
    for (int i = 0; i < str.length(); i++) {
        char curr = str.charAt(i);
        if ( st.size() > 0 && curr == ')' && st.peek() == '(' ) {
            st.pop();
        } else if ( st.size() > 0 && curr == ']' && st.peek() == '[' ) {
            st.pop();
        } else if ( st.size() > 0 && curr == '}' && st.peek() == '{' ) {
            st.pop();
        } else {
            st.push(curr);
        }
    }
    return st.size() == 0;
}
```