MATCH PARENTHESIS IN AN EXPRESSION

CHECK WHETHER AN EXPRESSION HAS WELL-FORMED PARENTHESIS I.E. THE OPENING AND THE CLOSING BRACKETS MATCH

USE A STACK TO STORE THE OPENING BRACKETS EACH TIME YOU ENCOUNTER ONE

FOR EVERY CLOSING BRACKET COMPARE TO THE LAST ELEMENT PUSHED ONTO THE STACK

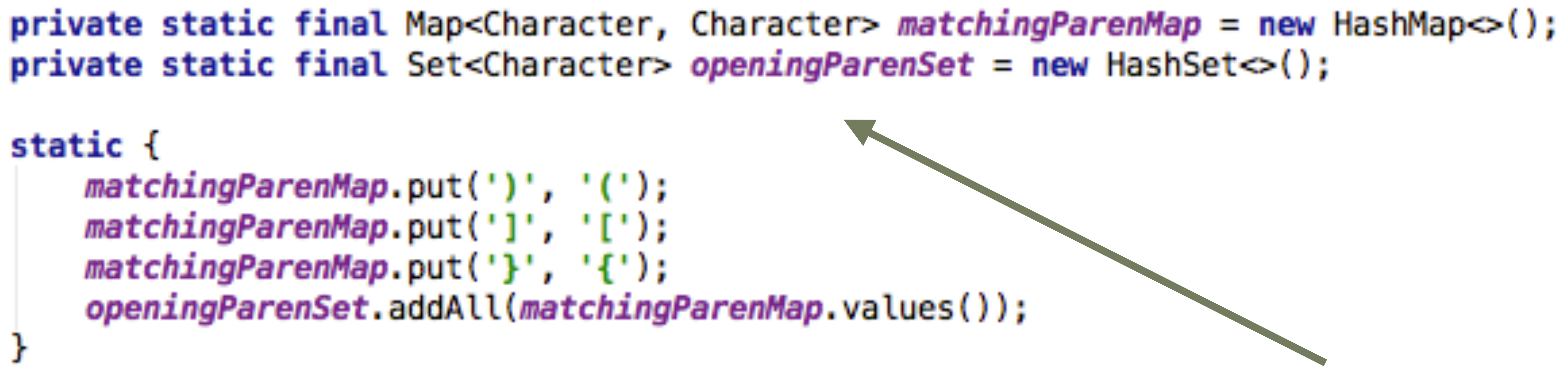
MATCH PARENTHESIS IN AN EXPRESSION

LET'S SEE SOME EXAMPLES:

(ABC) {DEF} [XYZ (LMN)]	MATCH
(ABC {DEF} [XYZ (LMN)]	MISMATCH
(ABC) {DEF} [XYZ (LMN)]}	MISMATCH

FIND MATCHING PARENTHESIS

MAP THE CLOSING BRACKETS WITH THE CORRESPONDING OPENING BRACKETS

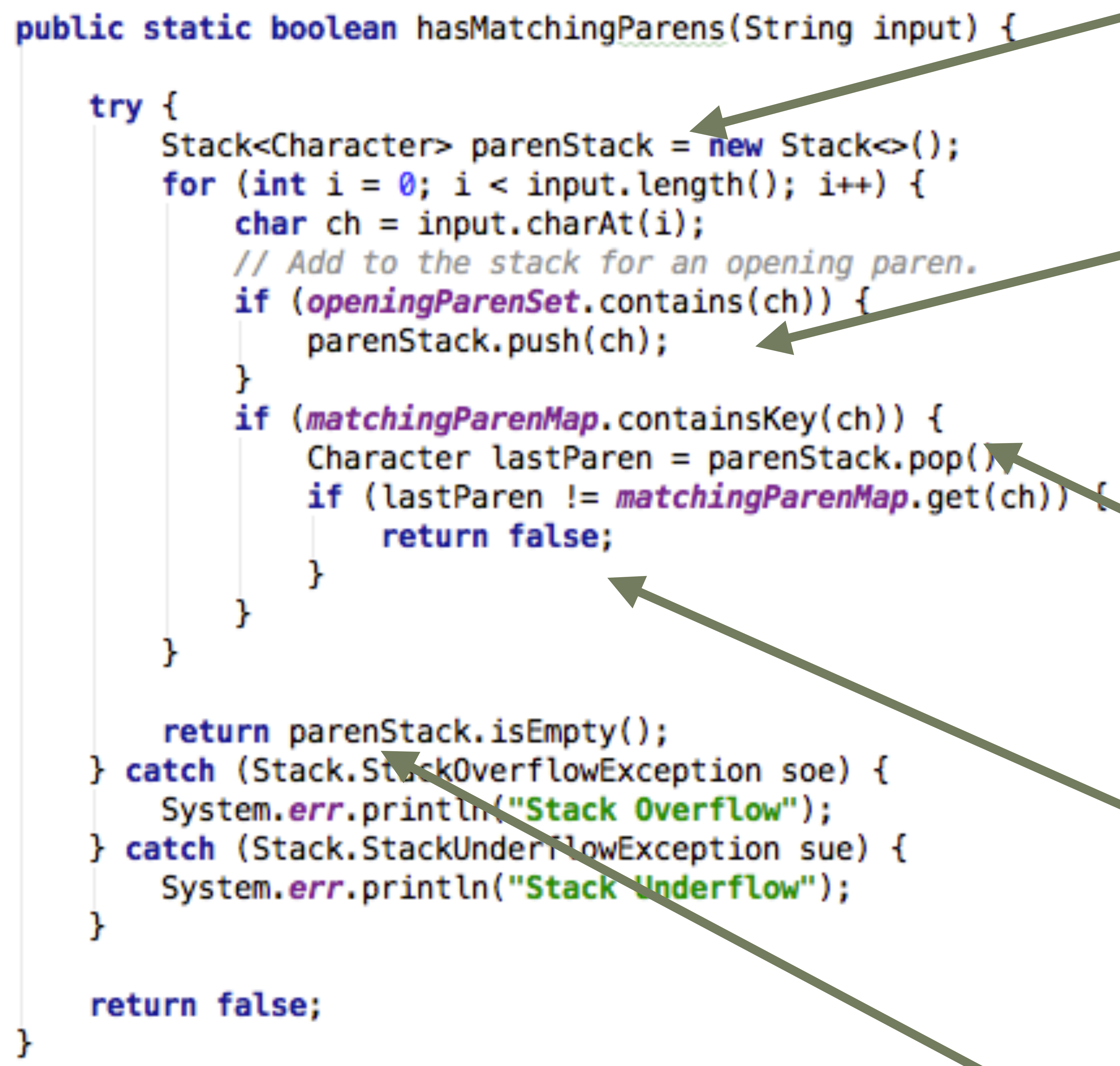


```java
private static final Map<Character, Character> matchingParenMap = new HashMap<>();
private static final Set<Character> openingParenSet = new HashSet<>();

static {
    matchingParenMap.put(')', '(');
    matchingParenMap.put(']', '[');
    matchingParenMap.put('}', '{');
    openingParenSet.addAll(matchingParenMap.values());
}
```

SET OF OPENING BRACKETS

CHECK IF THE PARENTHESIS MATCH



```java
public static boolean hasMatchingParens(String input) {

    try {
        Stack<Character> parenStack = new Stack<>();
        for (int i = 0; i < input.length(); i++) {
            char ch = input.charAt(i);
            // Add to the stack for an opening paren.
            if (openingParenSet.contains(ch)) {
                parenStack.push(ch);
            }
            if (matchingParenMap.containsKey(ch)) {
                Character lastParen = parenStack.pop();
                if (lastParen != matchingParenMap.get(ch)) {
                    return false;
                }
            }
        }

        return parenStack.isEmpty();
    } catch (Stack.StackOverflowException soe) {
        System.err.println("Stack Overflow");
    } catch (Stack.StackUnderflowException sue) {
        System.err.println("Stack Underflow");
    }

    return false;
}
```

SET UP A STACK TO HOLD ALL OPENING BRACKETS

PUSH THE BRACKETS FOUND ON TO THE STACK WHENEVER WE SEE AN OPENING BRACKET

IF IT'S A CLOSING BRACKET, POP THE TOP ELEMENT OF THE STACK TO SEE IF THE STACK HOLDS THE MATCHING OPENING BRACKET

IF THERE IS A MISMATCH RETURN FALSE

IF WE RUN THROUGH THE ENTIRE STRING AND THE STACK IS EMPTY AT THE END, THE BRACKETS ALL MATCH!