DSA PROJECT

A TEXTEDITOR WITH VARIOUS FUNCTIONALITIES

AKHIL VAVADIA : 1401095
RATNESH SHAH : 1401110
MAHARSH PATEL: 1401109

AIM:

- BUILD A TEXT-EDITOR INCLUDING FOLLOWING FUNCTIONALITIES:
- SPELL CHECKER
- PARENTHESIS MATCHING
- SYNTAX HIGHLIGHTING
- FIND & REPLACE
- AUTO COMPLETION

DATASTRUCTURES TO BE TILL NOW:

- ARRAYLIST
- TERNARY SEARCH TREE
- STACK
- LINKED LIST

SPELL CHECKER

- THE DATA STRUCTURE USED FOR IMPLEMENTING SPELL CHECKER IS TERNARY SEARCH TREE
- Methods implemented are:
- 1) Insert Word
- 2) Search Word
- 3) Delete Word
- 4) Check Empty
- 5) Make Empty

To Open the Dictionary text File we have used Click Here.

INSERT WORD :

- Code:

```
public void insert(String word) {
    root = insert(root, word.toCharArray(), 0);
}

// function to insert for a word
public TSTNode insert(TSTNode r, char[] word, int ptr)
{
    if (r == null)
        r = new TSTNode(word[ptr]);

    if (word[ptr] < r.data)
        r.left = insert(r.left, word, ptr);
    else if (word[ptr] > r.data)
        r.right = insert(r.right, word, ptr);
    else
    {
        if (ptr + 1 < word.length)
            r.middle = insert(r.middle, word, ptr + 1);
        else
            r.isEnd = true;
    }
    return r;
}
```

SEARCH WORD:

- Code:

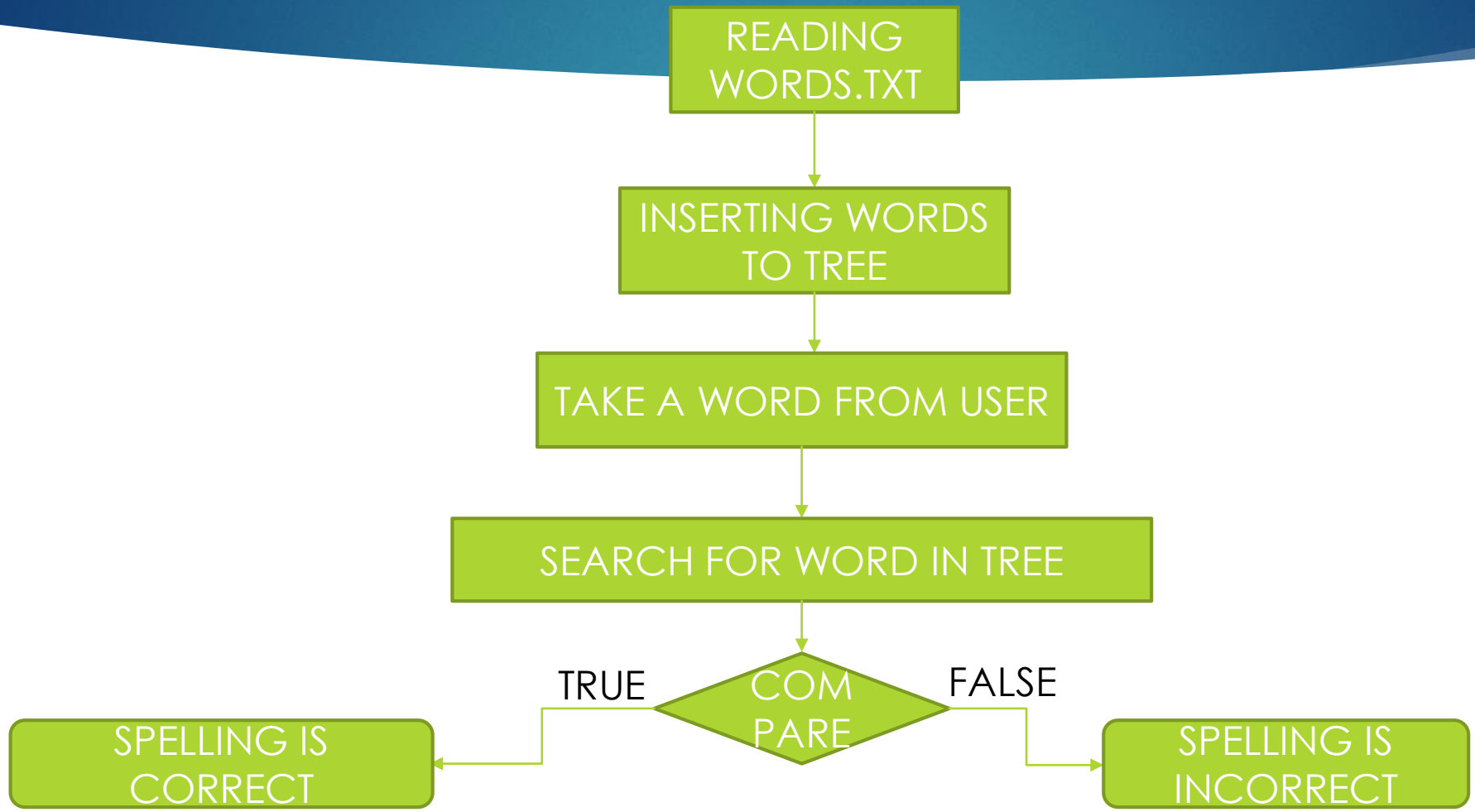
```
// function to search for a word
public boolean search(String word)
{
    return search(root, word.toCharArray(), 0);
}

// function to search for a word
private boolean search(TSTNode r, char[] word, int ptr)
{
    if (r == null)
        return false;

    if (word[ptr] < r.data)
        return search(r.left, word, ptr);

    else if (word[ptr] > r.data)
        return search(r.right, word, ptr);

    else
    {
        if (r.isEnd && ptr == word.length - 1)
            return true;

        else if (ptr == word.length - 1)
            return false;

        else
            return search(r.middle, word, ptr + 1);
    }
}
```

DELETE WORD

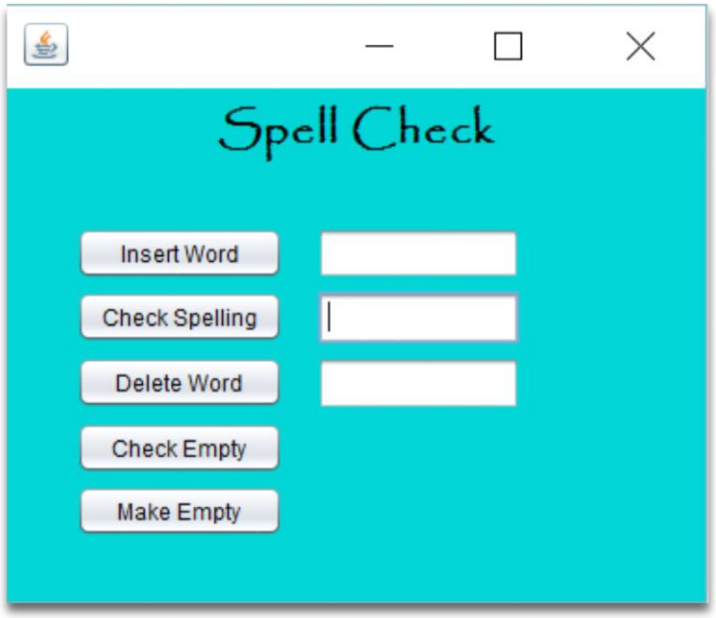
- Code:

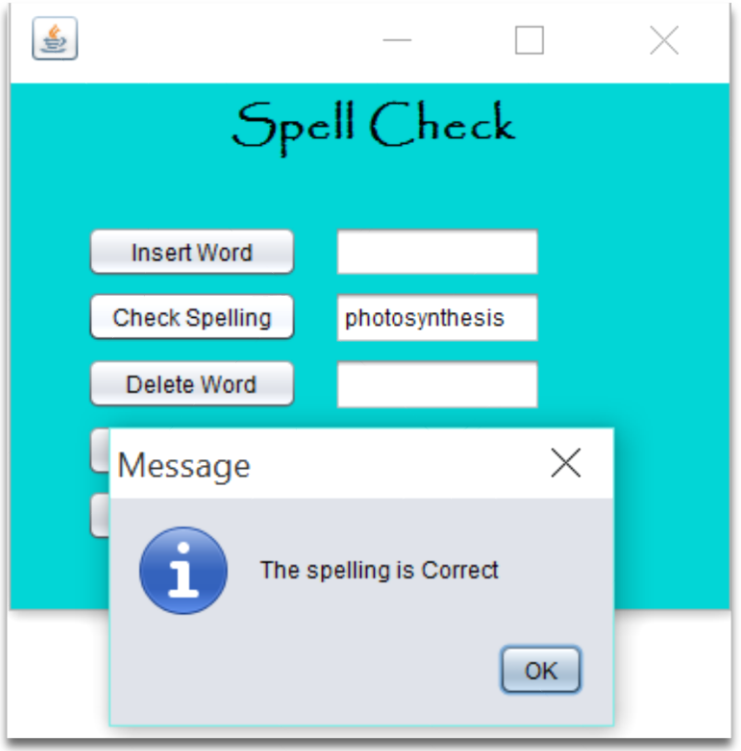
```
// function to delete a word
  public void delete(String word)
  {
      delete(root, word.toCharArray(), 0);
  }
  // function to delete a word
  private void delete(TSTNode r, char[] word, int ptr)
  {
      if (r == null)
          return;

      if (word[ptr] < r.data)
          delete(r.left, word, ptr);
      else if (word[ptr] > r.data)
          delete(r.right, word, ptr);
      else
      {
          // to delete a word just make isEnd false
          if (r.isEnd && ptr == word.length - 1)
              r.isEnd = false;

          else if (ptr + 1 < word.length)
              delete(r.middle, word, ptr + 1);
      }
  }
```

FLOW DIAGRAM

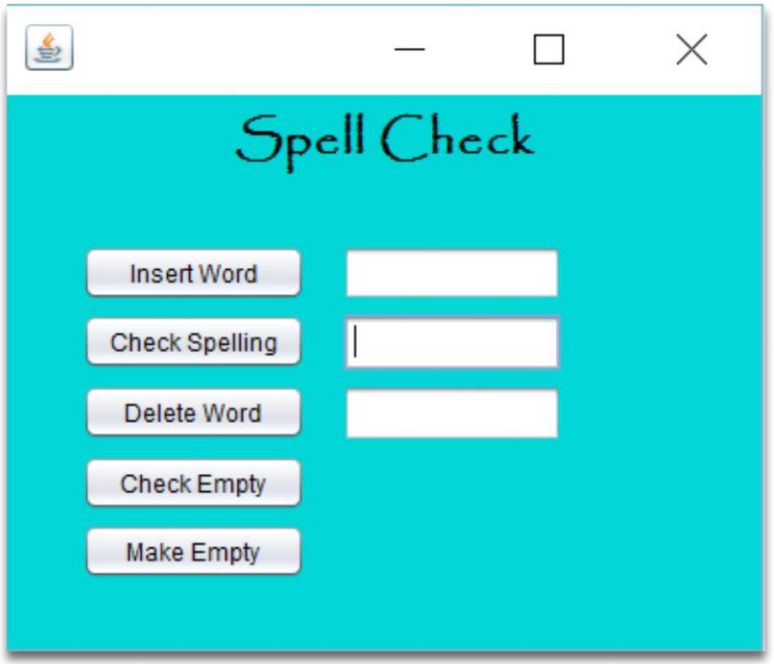


OUTPUT:

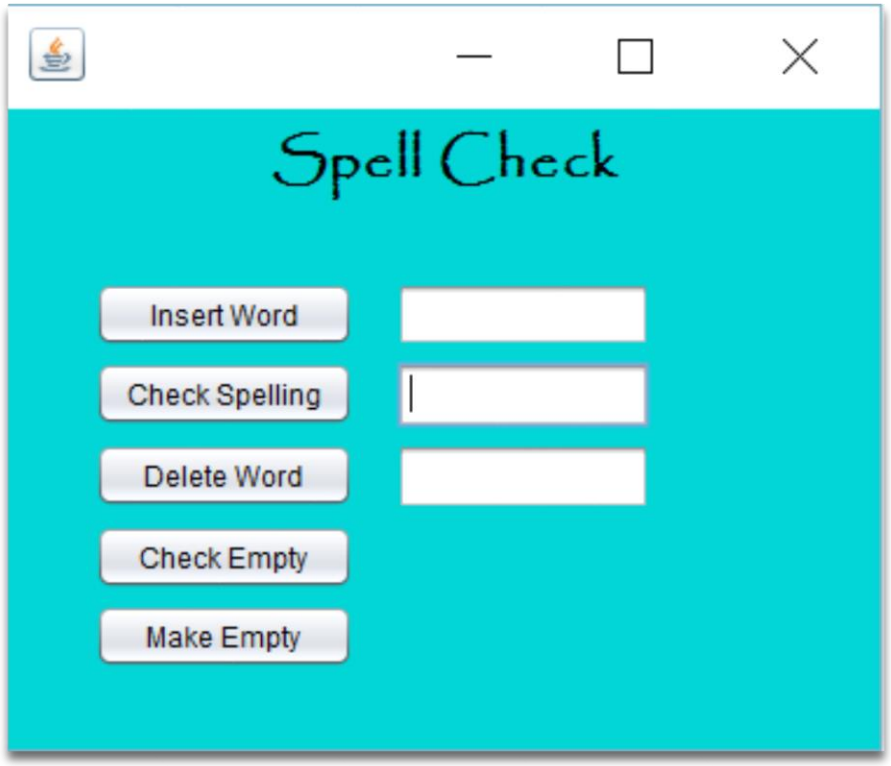

Spell Check
Insert Word
Check Spelling
Delete Word
Check Empty
Make Empty

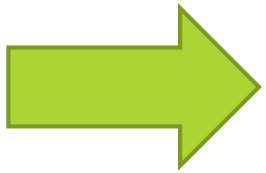

Spell Check
Insert Word
Check Spelling
photosynthesis
Delete Word
Message
The spelling is Correct
OK

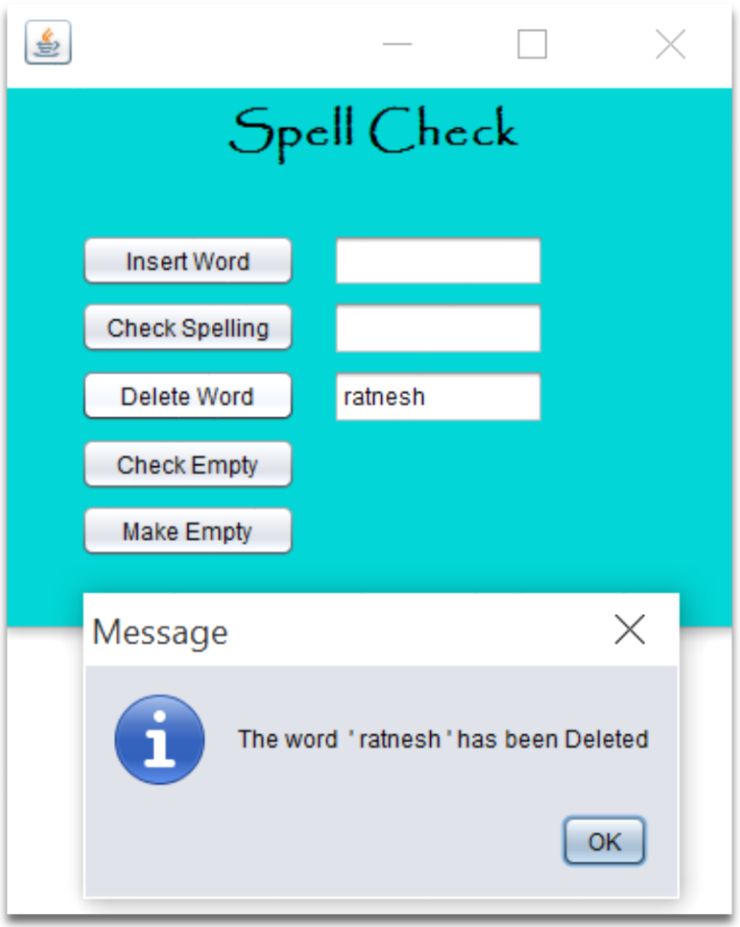
OUTPUT FOR INSERT WORD



OUTPUT FOR DELETE WORD:







PARENTHESIS MATCHING

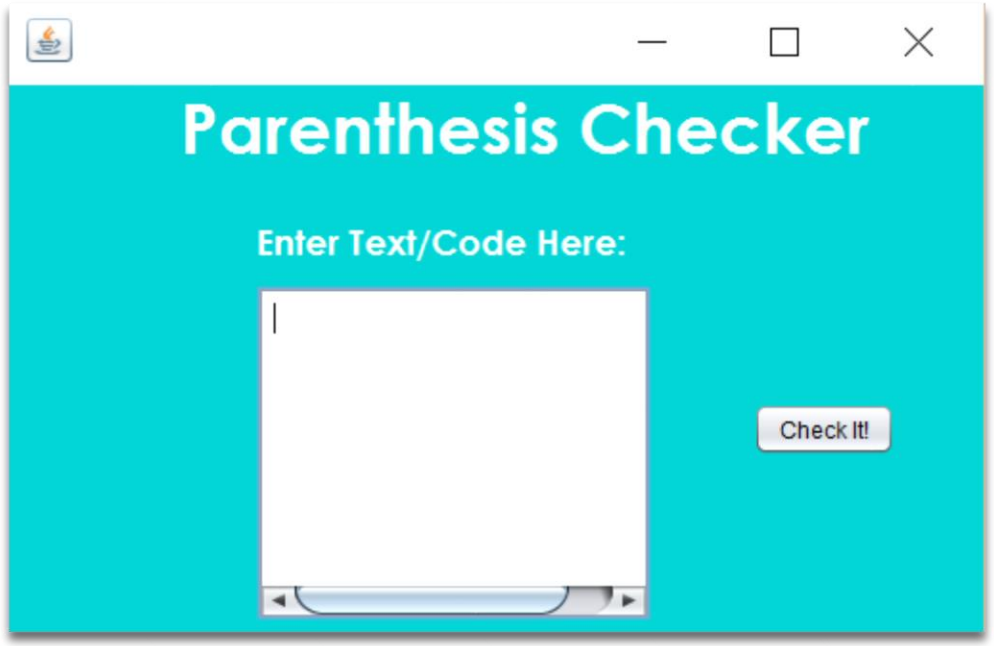
- IMPLEMENTED STACK USING ARRAY
- METHODS:
- 1) PUSH
- 2) POP
- 3) PEEK
- 4) ISEMPTY

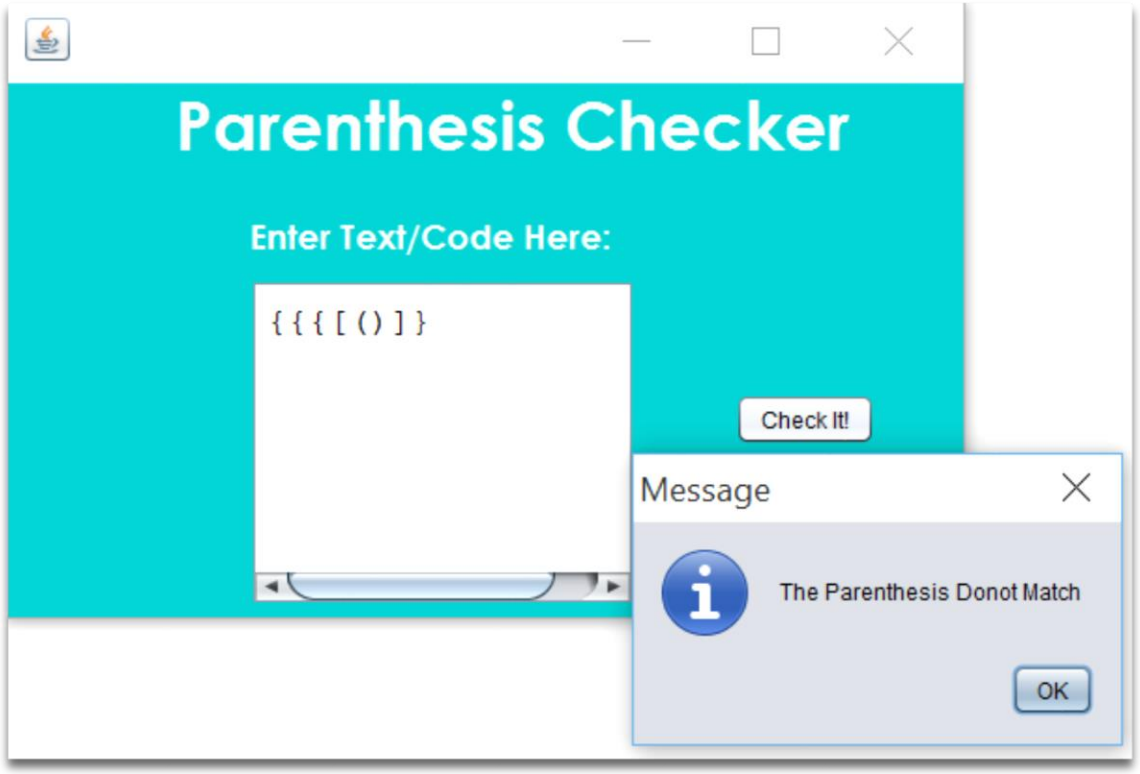
CHECKING OF PARENTHESIS:

```
public static boolean Check(String s) {
      Stack obj=new Stack(s.length());
       for (int i = 0; i < s.length(); i++) {

          if(s.charAt(i) == Left_Bracket)
          { //If '(' is encountered in the string than it is pushed into the stack
             obj.push(Left_Bracket);
          }

          else if(s.charAt(i) == L_BRACE)
          { //If '{' is encountered in the string than it is pushed into the stack
             obj.push(L_BRACE);
          }

          else if(s.charAt(i) == L_BRACKET)
          {//If '[' is encountered in the string than it is pushed into the stack
             obj.push(L_BRACKET);
          }

          else if (s.charAt(i) == Right_Bracket)
          {//If ')' is not obtained after the opening '(' then it returns false
             if (obj.isEmpty())
                return false;
             if (obj.pop() != Left_Bracket)
                return false;
          }

          else if (s.charAt(i) == R_BRACE) {
                if (obj.isEmpty())
                   return false;
                if (obj.pop() != L_BRACE)
                   return false;
             }

             else if (s.charAt(i) == R_BRACKET) {
                if (obj.isEmpty())
                   return false;
                if (obj.pop() != L_BRACKET)
                   return false;
             }

       }
       return obj.isEmpty();
    }
```

OUTPUT:

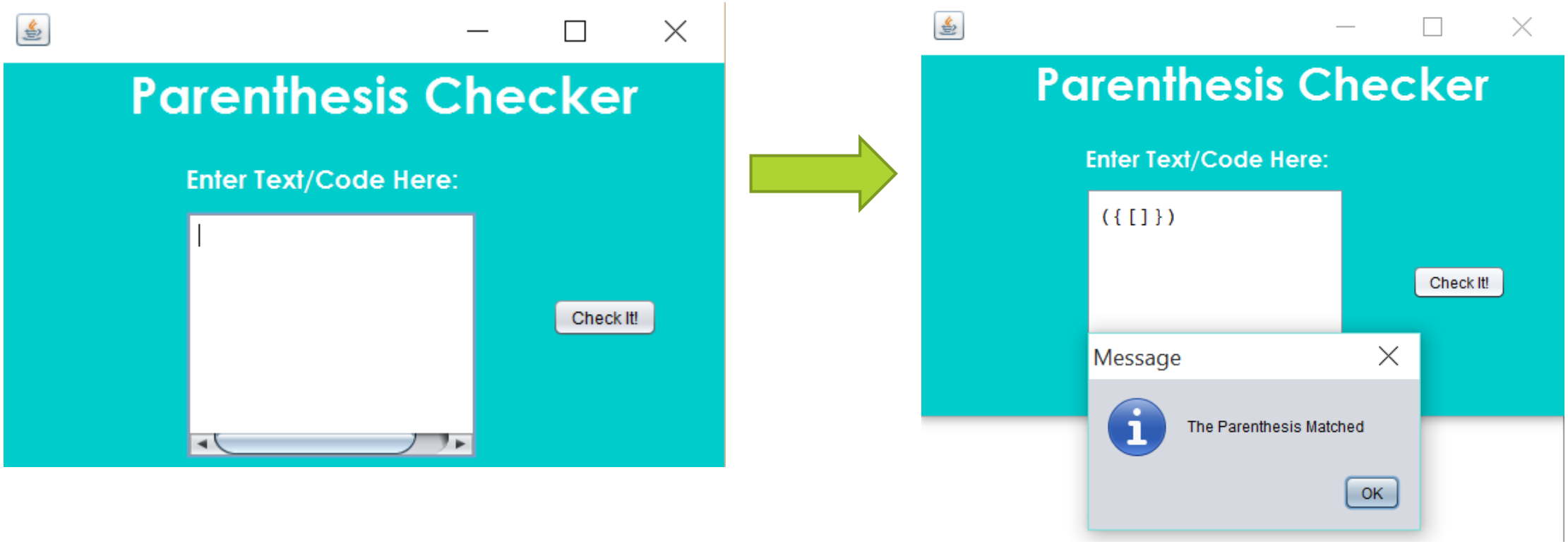
- PARENTHESIS DONOT MATCH CASE:



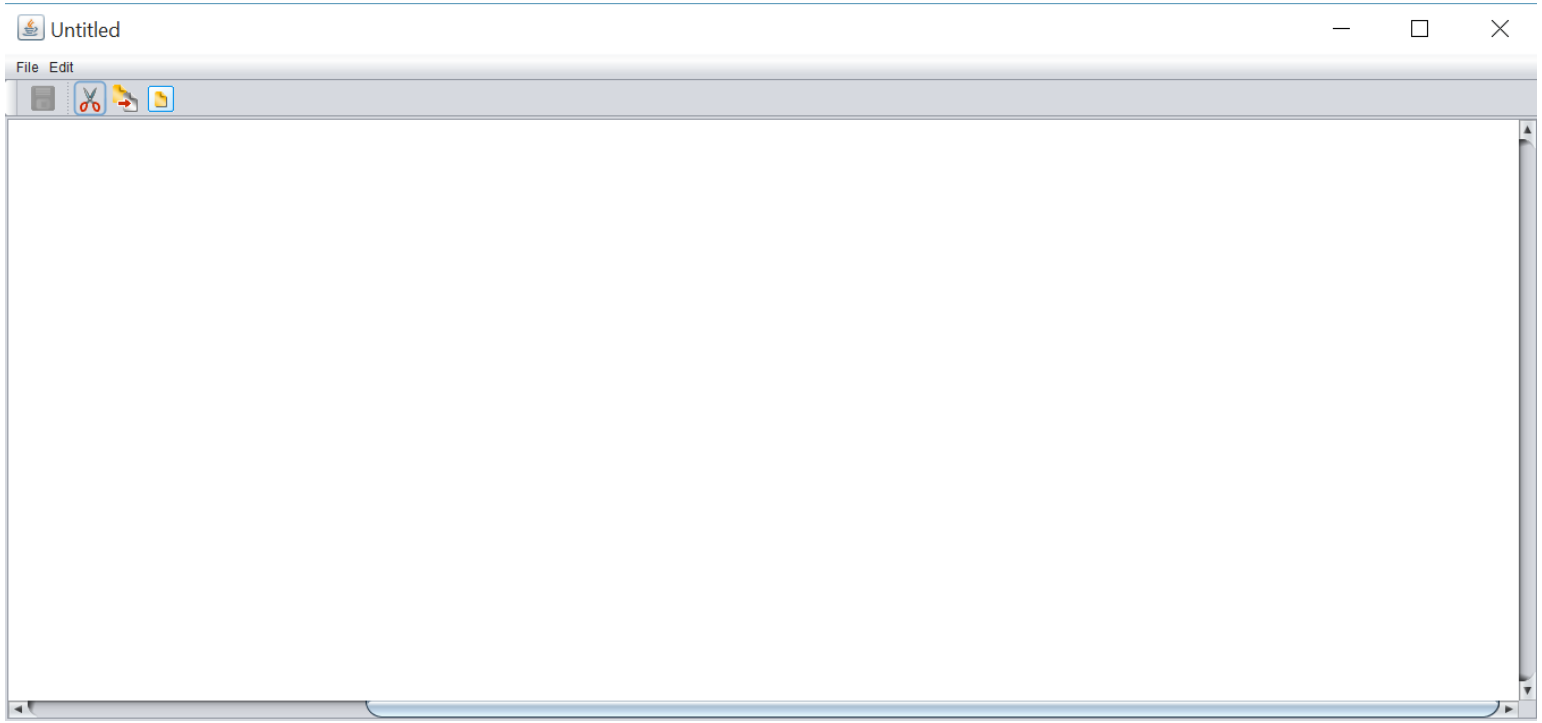


OUTPUT:

- PARENTHESIS MATCHING CASE:



TEXT EDITOR:



Click here