Create an Arraylist of WordPair objects.

Run | Download | Show CodeLens | Reformat | Pair?

11/10/2024, 12:41:52 PM - 2 of 2

```
1 //Code by Akshat Garg
2 import java.util.*;
3
4 public class WordPairTest
5 {
6     public static void main(String[] args)
7     {
8         ArrayList<WordPair> pairs = new ArrayList<WordPair>();
9
10        pairs.add(new WordPair("hi", "there"));
11        pairs.add(new WordPair("hi", "bye"));
12        System.out.println(pairs);
13    }
14 }
15
16 class WordPair
17 {
18    private String word1;
19    private String word2;
20
21    public WordPair(String word1, String word2)
22    {
23        this.word1 = word1;
24        this.word2 = word2;
25    }
26
27    public String getFirst()
28    {
29        return word1;
30    }
31
32    public String getSecond()
33    {
34        return word2;
35    }
36
37    public String toString()
38    {
39        return "(" + word1 + ", " + word2 + ")";
40    }
41 }
42
```

```
[(hi, there), (hi, bye)]
```

Result	Expected	Actual	Notes
Pass	[(hi, there), (hi, bye)]	[(hi, there), (hi, bye)]	Expected output from main
Pass	true	true	Checking that code contains ArrayList declaration

You got 2 out of 2 correct. 100.00%

Activity: 7.3.5.1 ActiveCode (ArrayListWordPair1)

FRQ WordPairs Challenge: Complete the constructor for `WordPairsList` below which will add pairs of words from a given array to the `ArrayList`. Then, complete the method `numMatches()` as described below this exercise.

Run | 11/10/2024, 12:51:31 PM - 5 of 5 | Download | Show CodeLens | Reformat | Pair?

```
//Code by Akshat Garg
import java.util.*;

public class WordPairsList
{
    private ArrayList<WordPair> allPairs;

    public WordPairsList(String[] words)
    {
        // WRITE YOUR CODE HERE
        // initialize allPairs to an empty ArrayList of WordPair objects
        allPairs = new ArrayList<WordPair>();
        // nested loops through the words array to add each pair to allPairs
        for (int k = 0; k < words.length-1; k++){
            for (int i = k + 1; i < words.length; i++){
                allPairs.add(new WordPair(words[k],words[i]));
            }
        }
    }

    public int numMatches()
    {
        int count = 0;
        // Write the code for the second part described below
        for (int j = 0; j < allPairs.size();j++){
            if (allPairs.get(j).getFirst() == allPairs.get(j).getSecond()){
                count++;
            }
        }
        return count;
    }

    public String toString()
    {
        return allPairs.toString();
    }

    public static void main(String[] args)
    {
        String[] words = {"Hi", "there", "Tyler", "Sam"};
        WordPairsList list = new WordPairsList(words);
        System.out.println(list);
```

```
[(Hi, there), (Hi, Tyler), (Hi, Sam), (there, Tyler), (there, Sam), (Tyler, Sam)]
The number of matched pairs is: 0
```

Result	Expected	Actual	Notes
Pass	[(Hi, there), (Hi, Tyler), (Hi, Sam), (there, Tyler), (there, Sam), (Tyler, Sam)]	[(Hi, there), (Hi, Tyler), (Hi, Sam), (there, Tyler), (there, Sam), (Tyler, Sam)]	Part 1 - Add all word pairs from main()
Pass	[(Hi, Hi), (Hi, Test), (Hi, Test), (Hi, Test), (Hi, Test), (Hi, Test), (Test, Test)]	[(Hi, Hi), (Hi, Test), (Hi, Test), (Hi, Test), (Hi, Test), (Hi, Test), (Test, Test)]	Part 1 - Add all word pairs with {"Hi", "Hi", "Test", "Test"}
Pass	The number of matched pairs is: 2	The number of matched pairs is: 2	Part 2 - numMatches() with {"Hi", "Hi", "Test", "Test"}

You got 3 out of 3 correct. 100.00%

Activity: 7.3.5.2 ActiveCode (challenge-7-3-WordPairs)

I honestly expected some error from the allPairs.get(j).getFirst() on line 26, it just seemed awkward. Instead it turned out quite smooth and I didn't even get any errors. I didn't see the pseudocode at first and made the traversal myself, and was pleased to see the bounds line up perfectly with the pseudo code and my solution. Pretty fun and interesting problem, very smooth.