Qualified: Assessment Result

shruthi Kairamkonda

General Programming (basic) Results

Assessment Summary

100%

Team

Global

58 m, 34 s Active Time

25th Team Percentile

25th Global Percentile

- **Invited** by **Dev Eval** on Friday, October 2, 2020 10:38 AM
  Sent once, and opened 2 times
- **Invited** by **Dev Eval** on Tuesday, October 6, 2020 11:39 AM and then **canceled**
- shruthi Kairamkonda **opened** this assessment on Friday, October 2, 2020 3:22 PM
- shruthi Kairamkonda **started** this assessment on Tuesday, October 6, 2020 1:06 AM
- shruthi Kairamkonda **submitted** this assessment after **1 day and 18 minutes** on Wednesday, October 7, 2020 1:25 AM

This candidate spent **58 minutes** active in the browser working on the assessment which places them in the **25th Team Percentile** amongst candidates

Review Summary

Danny Brinlee –

Solutions Summary

Challenge	Score		Active Time
#1: Get the Middle Character	100%	Team	9 m, 20 s
#2: Multiples of 3 and 5	100%	Team	9 m, 50 s
#3: Markdown Headers	100%	Team	39 m, 24 s

Qualified: Assessment Code Solution

#1: Get the Middle Character

Scoring

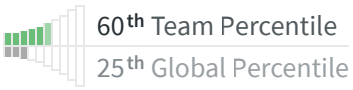
100%

4 / 4 Tests (1 Attempt)

Timing

9 m, 20 s Active Time

3,122 ms Run Time



Max Code Similarity

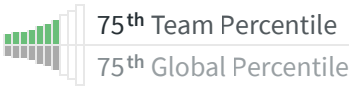
80%

5 Matches

Code Complexity

60

Lower is better



Challenge Reviews

Danny Brinlee

Instructions

You are going to be given a word. Your job is to return the middle character of the word. If the word's length is odd, return the middle character. If the word's length is even, return the middle 2 characters.

`Challenge.getMiddle(str)`

Find the middle character(s) of a word.

Parameters

`str`: ***`String`*** - word to pull middle characters from

Return Value

`String` - letter(s) in the middle of the word

Constraints

`0 < str < 1000`

Examples

str	Return Value
`"test"`	`"es"`

str	Return Value
`"testing"`	`"t"`
`"middle"`	`"dd"`

Solution Code

Java

```java
class Challenge {
  public static String getMiddle( String str ) {
    final int length = str.length();
    if((length % 2) == 0)
    {
       System.out.println("Given String length is even");
       final int index = length/2 - 1;
       return str.substring(index, index+2);
    }

    System.out.println("Given String length is odd");
    final int index = length/2;
    return str.substring(index, index+1);

      }
}
```

Candidate's Tests

```java
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.Arrays;
public class GetMiddleTests {
  @Test
  public void shouldFindTheMiddleCharacters() {
    assertEquals("A", Challenge.getMiddle("A"));
    assertEquals("es", Challenge.getMiddle("test"));
    assertEquals("t", Challenge.getMiddle("testing"));
    assertEquals("dd", Challenge.getMiddle("middle"));
  }
}
```

Qualified: Assessment Code Solution

#2: Multiples of 3 and 5

Scoring

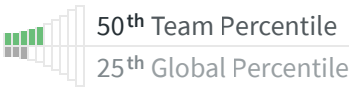
100%

4 / 4 Tests (3 Attempts)

Timing

9 m, 50 s Active Time

2,860 ms Run Time



Max Code Similarity

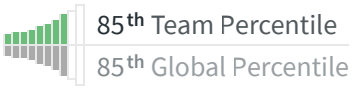
64%

1 Match

Code Complexity

64

Lower is better



Challenge Reviews

Danny Brinlee

Instructions

Task

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Complete the function `solution` so that it returns the sum of all the multiples of 3 or 5 **below** the number passed in.

Specification

`Challenge.solution(number)`

Finds the sum of multiples of 3 or 5 that is less than the provided number

Parameters

`number`: ***Integer*** - Maximum number to check against

Return Value

Integer - Sum of all multiples of either 3 or 5

Examples

number	Return Value

number	Return Value
`10`	`23`
`200`	`9168`

Solution Code

Java

```java
public class Solution {
    public static int solution(int number) {
      //find multiples of 3 below the given number
      int sum = 0;
      int three = 3;
      int multiple = 1;
      while(three*multiple < number)
      {
        sum = sum + (three*multiple);
        multiple++;
      }

      multiple = 1;
      while(5*multiple < number)
      {
        if((5*multiple)%3 != 0)
        {
           sum = sum + (5*multiple);
        }
        multiple++;
      }

        return sum;
    }
}
```

Candidate's Tests

```java
import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class SolutionTest {

    @Test
    public void testBasics() {
        assertEquals(23, Solution.solution(10));
    }
}
```

Qualified: Assessment Code Solution

#3: Markdown Headers

Scoring

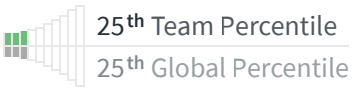
100%

3 / 3 Tests (10 Attempts)

Timing

39 m, 24 s Active Time

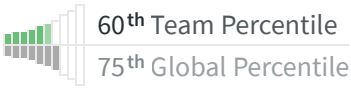
3,033 ms Run Time



Code Complexity

101

Lower is better



Challenge Reviews

Danny Brinlee

Instructions

Background

Markdown (https://en.wikipedia.org/wiki/Markdown) is a formatting syntax used by many documents (these instructions, for example!) because of its plain-text simplicity and its ability to be translated directly into HTML.

Task

Let's write a simple markdown parser function that will take in a single line of markdown and be translated into the appropriate HTML. To keep it simple, we'll support only one feature of markdown in atx syntax: headers.

Headers are designated by ***(1-6)*** hashes followed by a space, followed by text. The number of hashes determines the header level of the HTML output.

Specifications

`Challenge.markdownParser(markdown)`

Transforms given string into correct header form

Parameters

`markdown`: ***`String`*** - String to be changed into markdown format

Return Value

String - Formatted string

Examples

markdown	Return Value
`"# Header"`	`"<h1>Header</h1>"`
`"## Header"`	`"<h2>Header</h2>"`
`"###### Header"`	`"<h6>Header</h6>"`

Additional Rules

Header content should only come after the initial hashtag(s) plus a space character.

Invalid headers should just be returned as the markdown that was recieved, no translation necessary.

Spaces before and after both the header content and the hashtag(s) should be ignored in the resulting output.

Solution Code

Java

```
class Challenge {
    public static String markdownParser(String markdown) {
      markdown = markdown.strip();
      final int index = markdown.trim().indexOf(' ');
      if(index > 0)
      {
        final String mkdown = markdown.substring(0, index);
        long count = mkdown.chars().filter(ch -> ch == '#').count();
        if(mkdown.length() != count || count > 6)
        {
          System.out.println("Invalid Markdown chars found");
          return markdown;
        }

        String header = new StringBuilder().append("h").append(count).append(">").toString();

        return new StringBuilder().append("
<").append(header).append(markdown.substring(index+1).trim()).append("
</").append(header).toString();
      }

      return markdown;

```

```
22
23
24      }
25  }
```

Candidate's Tests

```
1   import org.junit.Test;
2   import static org.junit.Assert.*;
3   public class MarkdownParserTests {
4     @Test
5     public void basicValidCases() {
6       String expected = "<h1>header</h1>";
7       String actual = Challenge.markdownParser("# header");
8       assertEquals(expected, actual);
9       String expected1 = "<h2>smaller header</h2>";
10      String actual1 = Challenge.markdownParser("## smaller header");
11      assertEquals(expected1, actual1);
12    }
13    @Test
14    public void basicInvalidCases() {
15      String expected = "#Invalid";
16      String actual = Challenge.markdownParser("#Invalid");
17      assertEquals(expected, actual);
18    }
19  }
```