Java Freshers

Candidate-ID	0012400091
Candidate Name	chinnam Vivek
Email	vivekchinnam01@gmail.com
Mobile	8688434227
Source	Naukari

Screening Test

Completed on : 23-Dec-2024 09:25 PM

#	Question	Response	Marks
1	Have you fully completed your education and prepared to work	Yes	1
2	Have you taken any long term training coure on Java Technologies	Yes	1
3	Have you recently completed your education and training, and there no is major Gap (more then one year after the education)	Yes	1
4	Are you, willing to work on-premises (there is no work from home)	Yes	1
5	Can you write simple Java Programs on your own	Yes	1
6	Are you, available immediately to join us, if we hire you	Yes	1
7	Did you complete any kind of Developer Certification in Java	Yes	1
8	Do you know how to write unit testcases using Junit	Yes	1
9	if we hire you, will you be prepared to stay and work with us at least for two years	Yes	1
10	Will you be flexible any role in Software Engineering	Yes	1
11	Explain about yourself why you are the right candiate for this Job	video response	0

Overall Score : 10/11 (Promoted)

Coding Test

Completed on : 24-Dec-2024 08:36 PM

1. Write a program or method to find the smallest and largest numbers in an array of integers. Implement two methods: findSmallest(int[] numbers): Returns the smallest number in the array. findLargest(int[] numbers): Returns the largest number in the array. Sample Input : If the input array is null or empty, the method should throw an IllegalArgumentException with the message "Array cannot be null or empty". Use Java's Stream API to solve the problem. Expected Output : Smallest number: 1 Largest number: 20
Marks Scored : 2

```
import java.util.Arrays;

public class NumberUtils {

   public static int findSmallest(int[] numbers) {
      if (numbers == null || numbers.length == 0) {
         throw new IllegalArgumentException("Array cannot be null or empty");
      }
      return Arrays.stream(numbers)
                  .min()
                  .getAsInt();
   }

   public static int findLargest(int[] numbers) {
      if (numbers == null || numbers.length == 0) {
         throw new IllegalArgumentException("Array cannot be null or empty");
      }
      return Arrays.stream(numbers)
                  .max()
                  .getAsInt();

   }
   public static void main(String[] args) {
      int[] numbers = {1,2,3,4,5,20};
      try {
         int smallest = findSmallest(numbers);
         int largest = findLargest(numbers);

         System.out.println("smallest number:" + smallest);
         System.out.println("largest number:" + largest);

      } catch (IllegalArgumentException e) {
         System.out.println(e.getMessage());
      }
   }
}
```

2. Write a Java program that counts the occurrences of words in a given input string that are of a specified minimum length or longer. The program should return a map of words sorted in descending order of their frequency. If two words have the same frequency, they should be sorted alphabetically in ascending order. Sample Input : Ignore case sensitivity (e.g., "Test" and "test" should be treated as the same word). Words are sequences of

alphanumeric characters. Punctuation and other symbols should not be considered part of words. Words shorter than the specified minimum length should be excluded from the count. Expected Output : Words with minimum length 8: exceptional: 2 wonderful: 1
Marks Scored : 4

```
import java.util.*;
import java.util.stream.Collectors;

public class WordCounter {

  public static Map getWordsWithMinLength(String input, int minLength) {
    if (input == null || input.isEmpty()) {
      return Collections.emptyMap();
      //throw new IllegalArgumentException("Input string cannot be null or empty");
    }

    String[] words = input.split("\\W+");
    Map wordCount = new HashMap<>();

    for (String word : words) {
      if (word.length() >= minLength) {
        wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
      }
    }

    return wordCount.entrySet().stream().sorted((entry1, entry2) -> {
            int frequencyComparison = Integer.compare(entry2.getValue(), entry1.getValue());
            if (frequencyComparison == 0) {
              return entry1.getKey().compareTo(entry2.getKey());
            }
            return frequencyComparison;
          })
          .collect(Collectors.toMap(
            Map.Entry::getKey,
            Map.Entry::getValue,
            (e1, e2) -> e1,
            LinkedHashMap::new
          ));
  }
  public static void main(String[] args) {
    String input = "The exceptional and exceptional people were wonderful.";
    int minLength = 8;
    try {

      Map wordCounts = getWordsWithMinLength(input, minLength);
      System.out.println("words with minimum length " + minLength + ":");
      wordCounts.forEach((word, count) -> System.out.println(word + ":" + count));
    } catch (IllegalArgumentException e) {
      System.out.println(e.getMessage());
    }
```

```
  }
}
```

3. Question: Write Java Program to use and apply the concept of Design Pattern Singleton with simplest program. Expected Output: Instance 1: Singleton@15db9742 Instance 2: Singleton@15db9742 Both instances are the same (Singleton verified). Message from Singleton: Hello from Singleton!
Marks Scored : 0

```
public class Singleton {

  // Private static instance of the class
  private static Singleton instance;

  // Private constructor to prevent instantiation
  private Singleton() {}

  // Public method to provide access to the instance
  public static Singleton getInstance() {
     if (instance == null) {
        instance = new Singleton();
     }
     return instance;
  }

  // Example method to demonstrate functionality
  public String getMessage() {
     // Implement your code here
     //return "Hello from Singleton!";
     return "Message from Singleton: Hello from Singleton!";
     //System.out.println("Message from Singleton: Hello from Singleton!");
  }

  // Main method to test the Singleton functionality
  public static void main(String[] args) {
     // Implement your code here
     Singleton instance1 = Singleton.getInstance();
     Singleton instance2 = Singleton.getInstance();

     System.out.println("Instance 1: \n" + instance1);
     System.out.println("Instance 2: " + instance2);

     if (instance1 == instance2) {
       System.out.println("Both instances are the same (Singleton verified). ");

     }
     //System.out.println("Both instances are the same (Singleton verified). "+ instance1.getMessage());

     // Implement your code here
     //System.out.println("Are both instances the same? yes" );
```

```
        String msg = instance1.getMessage();
        System.out.println(msg);

    }
}
```

Overall Score : 6/12 (Promoted)