30 Java String Coding Problems with Solutions

---

1. Reverse a String

```java
public class ReverseString {
    public static String reverse(String str) {
        return new StringBuilder(str).reverse().toString();
    }

    public static void main(String[] args) {
        System.out.println(reverse("hello")); // Output: olleh
    }
}
```

---

2. Check if a String is Palindrome

```java
public class PalindromeCheck {
    public static boolean isPalindrome(String str) {
        return str.equals(new StringBuilder(str).reverse().toString());
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome("madam")); // true
    }
}
```

---

3. Count Occurrences of a Character

```java
public class CharCount {
    public static int countChar(String str, char ch) {
        return (int) str.chars().filter(c -> c == ch).count();
    }

    public static void main(String[] args) {
        System.out.println(countChar("hello", 'l')); // 2
    }
}
```

---

4. Find Duplicate Characters in a String

```java
import java.util.*;

public class DuplicateCharacters {
    public static void findDuplicates(String str) {
        Map<Character, Integer> map = new HashMap<>();
        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        map.forEach((k, v) -> {
            if (v > 1) System.out.println(k + ": " + v);
        });
    }

    public static void main(String[] args) {
        findDuplicates("programming");
    }
}
```

---

5. Remove Duplicates from a String

```java
public class RemoveDuplicates {
    public static String removeDuplicates(String str) {
        return str.chars().distinct()
                .collect(StringBuilder::new, StringBuilder
                    ::appendCodePoint, StringBuilder::append)
                .toString();
    }

    public static void main(String[] args) {
        System.out.println(removeDuplicates("helloooaa")); // helo
    }
}
```

---

6. Check if Two Strings are Anagrams

```java
import java.util.Arrays;

public class AnagramCheck {
    public static boolean isAnagram(String s1, String s2) {
        char[] a1 = s1.toCharArray();
        char[] a2 = s2.toCharArray();
        Arrays.sort(a1);
        Arrays.sort(a2);
        return Arrays.equals(a1, a2);
    }

    public static void main(String[] args) {
        System.out.println(isAnagram("listen", "silent")); // true
    }
}
```

---

7. Find First Non-Repeated Character

```java
import java.util.LinkedHashMap;
import java.util.Map;

public class FirstNonRepeated {
    public static char firstNonRepeated(String str) {
        Map<Character, Integer> map = new LinkedHashMap<>();
        for (char ch : str.toCharArray()) {
            map.put(ch, map.getOrDefault(ch, 0) + 1);
        }
        for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            if (entry.getValue() == 1) return entry.getKey();
        }
        return '_';
    }

    public static void main(String[] args) {
        System.out.println(firstNonRepeated("swiss")); // w
    }
}
```

---

8. Check if a String Contains Only Digits

```java
public class OnlyDigits {
    public static boolean isNumeric(String str) {
        return str.matches("\\d+");
    }

    public static void main(String[] args) {
        System.out.println(isNumeric("12345")); // true
        System.out.println(isNumeric("12a45")); // false
    }
}
```

---

9. Count Vowels and Consonants

```java
public class CountVowelsConsonants {
    public static void countVowelsConsonants(String str) {
        int vowels = 0, consonants = 0;
        str = str.toLowerCase();
        for (char ch : str.toCharArray()) {
            if ("aeiou".indexOf(ch) != -1) vowels++;
            else if (Character.isLetter(ch)) consonants++;
        }
        System.out.println("Vowels: " + vowels + ", Consonants: "
            + consonants);
    }

    public static void main(String[] args) {
        countVowelsConsonants("Hello");
    }
}
```

---

10. Check if a String is a Rotation of Another String

```java
public class StringRotation {
    public static boolean isRotation(String s1, String s2) {
        return (s1.length() == s2.length()) && (s1 + s1).contains(s2);
    }

    public static void main(String[] args) {
        System.out.println(isRotation("abcd", "cdab")); // true
    }
}
```

---

11. Find the longest word in a sentence

```java
import java.util.*;
public class StringProblems {

    // 11. Find the longest word in a sentence
    public static String longestWord(String sentence) {
        return Arrays.stream(sentence.split(" "))
                     .max(Comparator.comparingInt(String::length))
                     .orElse("");
    }
    public static void main(String[] args) {
        System.out.println("Longest Word: " + longestWord("Java is
            a powerful programming language"));
    }
}
```

---

12. Convert a string to Title Case

```java
import java.util.*;
public class StringProblems {

    public static String toTitleCase(String str) {
        String[] words = str.split(" ");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            sb.append(Character.toUpperCase(word.charAt(0)))
              .append(word.substring(1).toLowerCase())
              .append(" ");
        }
        return sb.toString().trim();
    }
    public static void main(String[] args) {
        System.out.println("Title Case: " + toTitleCase("hello
            world from java"));
    }
}
```

13. Find all permutations of a string

```java
import java.util.*;
public class StringProblems {

        public static void permute(String str, String result) {
        if (str.isEmpty()) {
            System.out.println(result);
            return;
        }
        for (int i = 0; i < str.length(); i++) {
            permute(str.substring(0, i) + str.substring(i + 1),
                result + str.charAt(i));
        }
    }

    public static void main(String[] args) {
        System.out.println("Permutations of 'abc':");
        permute("abc", "");
   }
}
```

14. Convert string to integer without `Integer.parseInt()`

```java
import java.util.*;
public class StringProblems {

    public static int stringToInt(String str) {
      int num = 0;
      for (char c : str.toCharArray()) {
          num = num * 10 + (c - '0');
      }
      return num;
   }
   public static void main(String[] args) {
       System.out.println("String to Integer: " + stringToInt
           ("1234"));
  }
}

```

15. Find the frequency of words in a sentence

```java
import java.util.*;
public class StringProblems {

        public static void wordFrequency(String sentence) {
         Map<String, Integer> map = new HashMap<>();
         for (String word : sentence.split(" ")) {
             map.put(word, map.getOrDefault(word, 0) + 1);
         }
         map.forEach((k, v) -> System.out.println(k + ": " + v));
     }

     public static void main(String[] args) {
        System.out.println("Word Frequency in 'java java python c++
            c++ java':");
         wordFrequency("java java python c++ c++ java");
   }
}
```

16. Convert `String` to `char[]` without `.toCharArray()`

```java
import java.util.*;
public class StringProblems {

        // 16. Convert String to char[] without .toCharArray()
    public static char[] stringToCharArray(String str) {
        char[] chars = new char[str.length()];
        for (int i = 0; i < str.length(); i++) {
            chars[i] = str.charAt(i);
        }
        return chars;
    }

    public static void main(String[] args) {
       System.out.println("Convert String to char[]: " + Arrays
           .toString(stringToCharArray("hello")));
   }
}
```

17. Remove all non-alphabetic characters from a string

```java
import java.util.*;
public class StringProblems {

    // 17. Remove all non-alphabetic characters from a string
    public static String removeNonAlphabets(String str) {
        return str.replaceAll("[^a-zA-Z]", "");
    }

    public static void main(String[] args) {
        System.out.println("Remove Non-Alphabets: " +
            removeNonAlphabets("h3ll0 w@rld!"));
    }
}

```

18. Convert lowercase to uppercase without `toUpperCase()`

```java
import java.util.*;
public class StringProblems {

    // 18. Convert lowercase to uppercase without toUpperCase()
    public static String toUpperCaseManual(String str) {
        StringBuilder sb = new StringBuilder();
        for (char c : str.toCharArray()) {
            sb.append((c >= 'a' && c <= 'z') ? (char) (c - 32) : c
                );
        }
        return sb.toString();
    }

    public static void main(String[] args) {
        System.out.println("To UpperCase: " +
            toUpperCaseManual("hello"));
    }
}
```

19. Reverse words in a sentence

```java
import java.util.*;
public class StringProblems {

    // 19. Reverse words in a sentence
    public static String reverseWords(String sentence) {
        String[] words = sentence.split(" ");
        Collections.reverse(Arrays.asList(words));
        return String.join(" ", words);
    }

    public static void main(String[] args) {
            System.out.println("Reverse Words: " + reverseWords
                ("hello world from java"));
  }
}
```

20. Check if two strings are one edit distance apart

```java
import java.util.*;
public class StringProblems {

    // 20. Check if two strings are one edit distance apart
    public static boolean isOneEditDistance(String s1, String s2)
        {
        if (Math.abs(s1.length() - s2.length()) > 1) return false;
        int count = 0, i = 0, j = 0;
        while (i < s1.length() && j < s2.length()) {
            if (s1.charAt(i) != s2.charAt(j)) {
                if (++count > 1) return false;
                if (s1.length() > s2.length()) i++;
                else if (s1.length() < s2.length()) j++;
                else { i++; j++; }
            } else { i++; j++; }
        }
        return true;
    }

    public static void main(String[] args) {
           System.out.println("One Edit Distance: " +
               isOneEditDistance("cat", "cut"));
    }
}
```

---

21. Implement `strStr()` function (Find substring index)

```java
import java.util.*;
public class StringProblems {

    // 21. Implement strStr() function (Find substring index)
    public static int strStr(String haystack, String needle) {
        return haystack.indexOf(needle);
    }

    public static void main(String[] args) {
            System.out.println("strStr: " + strStr("hello", "ll"
                ));
    }
}
```

22. Compress a string (aaabb -> a3b2)

```java
import java.util.*;
public class StringProblems {

    // 22. Compress a string (aaabb -> a3b2)
    public static String compressString(String str) {
        StringBuilder sb = new StringBuilder();
        int count = 1;
        for (int i = 1; i <= str.length(); i++) {
            if(i==str.length() || str.charAt(i) != str.charAt(i-1
                )) {
                sb.append(str.charAt(i - 1)).append(count);
                count = 1;
            } else {
                count++;
            }
        }
        return sb.toString();
    }


    public static void main(String[] args) {
            System.out.println("Compressed: " + compressString
                ("aaabb"));
  }
}
```

23. Find longest common prefix in an array of strings

```java
import java.util.*;
public class StringProblems {

    // 23. Find longest common prefix in an array of strings
    public static String longestCommonPrefix(String[] strs) {
        if (strs.length == 0) return "";
        String prefix = strs[0];
        for (String str : strs) {
            while (!str.startsWith(prefix)) {
                prefix = prefix.substring(0, prefix.length() - 1);
                if (prefix.isEmpty()) return "";
            }
        }
        return prefix;
    }

    public static void main(String[] args) {
        System.out.println("Longest Prefix: " +
            longestCommonPrefix(new String[]{"flower", "flow",
            "flight"}));
    }
}
```

---

24. Implement `isSubstring()` without `contains()`

```java
import java.util.*;
public class StringProblems {

   // 24. Implement isSubstring() without contains()
    public static boolean isSubstring(String str, String sub) {
        return str.indexOf(sub) != -1;
    }
    public static void main(String[] args) {
        System.out.println("isSubstring: " + isSubstring("hello
            world", "world"));
    }
  }

```

---

25. Print all substrings of a string

```java
import java.util.*;
public class StringProblems {

    // 25. Print all substrings of a string
    public static void printSubstrings(String str) {
        for (int i = 0; i < str.length(); i++) {
            for (int j = i + 1; j <= str.length(); j++) {
                System.out.println(str.substring(i, j));
            }
        }
    }
    public static void main(String[] args) {
        printSubstrings("abc");
    }
}
```

---

26. Convert a number to words (123 -> "One Hundred Twenty-Three")

```java
import java.util.*;
public class StringProblems {

    // 26. Convert a number to words (123 -> "One Hundred Twenty-Three")
    public static String numberToWords(int num) {
        String[] belowTen = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine"};
        String[] belowTwenty = {"Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};
        String[] belowHundred = {"", "", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};
        if (num < 10) return belowTen[num];
        if (num < 20) return belowTwenty[num - 10];
        if (num < 100) return belowHundred[num / 10] + " " + belowTen[num % 10];
        return belowTen[num / 100] + " Hundred " + numberToWords(num % 100);
    }
    public static void main(String[] args) {
        System.out.println("Number to Words: " + numberToWords(100));
    }
}
```

---

27. Validate an email address format

```java
import java.util.*;
public class StringProblems {

    // 27. Validate an email address format
    public static boolean isValidEmail(String email) {
        return email.matches("^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$");
    }
    public static void main(String[] args) {
        System.out.println("Valid Email: " + isValidEmail
            ("test@example.com"));
    }
}
```

---

28. Convert CamelCase to snake_case

```java
import java.util.*;
public class StringProblems {

    // 28. Convert CamelCase to snake_case
    public static String camelToSnake(String str) {
        return str.replaceAll("([a-z])([A-Z])", "$1_$2"
            ).toLowerCase();
    }
    public static void main(String[] args) {
        System.out.println("Camel to Snake: " + camelToSnake
            ("camelCaseString"));
    }
}
```

---

29. Find the lexicographically smallest string

```java
import java.util.*;
public class StringProblems {

    // 29. Find the lexicographically smallest string
    public static String smallestLexicographic(String[] arr) {
        Arrays.sort(arr);
        return arr[0];
    }
    public static void main(String[] args) {
        System.out.println("Smallest Lexicographic: " +
            smallestLexicographic(new String[]{"banana", "apple",
            "cherry"}));
    }
}
```

---

30. Remove consecutive duplicate words from a sentence

```java
import java.util.*;
public class StringProblems {

    // 30. Remove consecutive duplicate words from a sentence
    public static String removeConsecutiveDuplicates(String
        sentence) {
        String[] words = sentence.split(" ");
        StringBuilder sb = new StringBuilder();
        String prev = "";
        for (String word : words) {
            if (!word.equals(prev)) {
                sb.append(word).append(" ");
                prev = word;
            }
        }
        return sb.toString().trim();
    }
    public static void main(String[] args) {
        System.out.println("Removed Duplicates: " +
            removeConsecutiveDuplicates("hello hello world world
            java java"));
    }
}
```

Happy learning! 🌸