IMPORTANT JAVA PROGRAMS BASED ON STRING

Check if two strings are anagrams

```
import java.util.Arrays;

public class AnagramCheck {
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";

        if (isAnagram(str1, str2)) {
            System.out.println("Strings are anagrams.");
        } else {
            System.out.println("Strings are not anagrams.");
        }
    }

    public static boolean isAnagram(String str1, String str2) {
        if (str1.length() != str2.length()) {
            return false;
        }
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();

        Arrays.sort(arr1);
        Arrays.sort(arr2);

        return Arrays.equals(arr1, arr2);
    }
}
```

Explanation:

- **if (str1.length() != str2.length())**: Checks if both strings have the same length.
- **Arrays.sort(arr1)**: Sorts the character array of the first string.
- **Arrays.equals(arr1, arr2)**: Compares the sorted character arrays of both strings.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Check if a string is a palindrome

```
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";

        if (isPalindrome(str)) {
            System.out.println("String is a palindrome.");
        } else {
            System.out.println("String is not a palindrome.");
        }
    }

    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(left) != str.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

Explanation:

- **while (left < right)**: Loops through the string comparing characters from the start and end.
- **str.charAt(left) != str.charAt(right)**: If characters don't match, it's not a palindrome.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Count the number of vowels and consonants in a string

```java
public class CountVowelsConsonants {
    public static void main(String[] args) {
        String str = "automation";
        int[] count = countVowelsAndConsonants(str);

        System.out.println("Vowels: " + count[0]);
        System.out.println("Consonants: " + count[1]);
    }

    public static int[] countVowelsAndConsonants(String str) {
        int vowelCount = 0;
        int consonantCount = 0;
        String vowels = "aeiouAEIOU";

        for (char ch : str.toCharArray()) {
            if (vowels.indexOf(ch) != -1) {
                vowelCount++;
            } else if (Character.isLetter(ch)) {
                consonantCount++;
            }
        }

        return new int[]{vowelCount, consonantCount};
    }
}
```

Explanation:

- **vowels.indexOf(ch) != -1**: Checks if the character is a vowel.
- **Character.isLetter(ch)**: Ensures that only letters are counted as consonants.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Find the first non-repeating character in a string

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

public class FirstNonRepeatingChar {
    public static void main(String[] args) {
        String str = "automation";
        char result = findFirstNonRepeating(str);
        System.out.println("First non-repeating character: " + result);
    }

    public static char findFirstNonRepeating(String str) {
        Map<Character, Integer> charCountMap = new LinkedHashMap<>();

        for (char ch : str.toCharArray()) {
            charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);
        }

        for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {
            if (entry.getValue() == 1) {
                return entry.getKey();
            }
        }
        return '\0';
    }
}
```

Explanation:

- **charCountMap.getOrDefault(ch, 0) + 1**: Increments the count of each character.
- **if (entry.getValue() == 1)**: Finds the first character that appears only once.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Reverse a string

```java
public class ReverseString {
    public static void main(String[] args) {
        String str = "Selenium";
        String reversed = reverse(str);
        System.out.println("Reversed string: " + reversed);
    }

    public static String reverse(String str) {
        StringBuilder reversedStr = new StringBuilder();

        for (int i = str.length() - 1; i >= 0; i--) {
            reversedStr.append(str.charAt(i));
        }

        return reversedStr.toString();
    }
}
```

Explanation:

- **for (int i = str.length() - 1; i >= 0; i--)**: Loops through the string from the end to the beginning.
- **reversedStr.append(str.charAt(i))**: Appends each character to the reversed string.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Check if a string contains only digits

```java
public class CheckDigits {
    public static void main(String[] args) {
        String str = "12345";

        if (containsOnlyDigits(str)) {
            System.out.println("String contains only digits.");
        } else {
            System.out.println("String contains non-digit characters.");
        }
    }

    public static boolean containsOnlyDigits(String str) {
        for (char ch : str.toCharArray()) {
            if (!Character.isDigit(ch)) {
                return false;
            }
        }
        return true;
    }
}
```

Explanation:

- **Character.isDigit(ch)**: Checks if each character is a digit.
- If any character is not a digit, it returns false.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Count the occurrence of each character in a string

```java
import java.util.HashMap;
import java.util.Map;

public class CharOccurrence {
    public static void main(String[] args) {
        String str = "testing";
        countCharOccurrence(str);
    }

    public static void countCharOccurrence(String str) {
        Map<Character, Integer> charCountMap = new HashMap<>();

        for (char ch : str.toCharArray()) {
            charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1);
        }

        for (Map.Entry<Character, Integer> entry : charCountMap.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
        }
    }
}
```

Explanation:

- **charCountMap.put(ch, charCountMap.getOrDefault(ch, 0) + 1)**: Increments the count of each character.
- **for (Map.Entry<Character, Integer> entry)**: Iterates through the map to print the count of each character.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Remove duplicate characters from a string

```java
public class RemoveDuplicates {
    public static void main(String[] args) {
        String str = "automation";
        String result = removeDuplicates(str);
        System.out.println("String after removing duplicates: " + result);
    }

    public static String removeDuplicates(String str) {
        StringBuilder result = new StringBuilder();

        for (char ch : str.toCharArray()) {
            if (result.indexOf(String.valueOf(ch)) == -1) {
                result.append(ch);
            }
        }

        return result.toString();
    }
}
```

Explanation:

- **result.indexOf(String.valueOf(ch)) == -1**: Checks if the character is already present in the result.
- If not present, the character is appended to the result.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Find all substrings of a string

```java
public class Substrings {
    public static void main(String[] args) {
        String str = "abc";
        findAllSubstrings(str);
    }

    public static void findAllSubstrings(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));
            }
        }
    }
}
```

Explanation:

- **str.substring(i, j)**: Extracts all substrings starting from index i to j.
- Nested loops ensure that all possible substrings are printed.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Find the most frequent character in a string

```
import java.util.HashMap;
import java.util.Map;

public class MostFrequentChar {
    public static void main(String[] args) {
        String str = "success";
        char mostFrequent = findMostFrequentChar(str);
        System.out.println("Most frequent character: " + mostFrequent);
    }

    public static char findMostFrequentChar(String str) {
        Map<Character, Integer> charCountMap = new HashMap<>();
        int maxCount = 0;
        char mostFrequent = '\0';

        for (char ch : str.toCharArray()) {
            int count = charCountMap.getOrDefault(ch, 0) + 1;
            charCountMap.put(ch, count);

            if (count > maxCount) {
                maxCount = count;
                mostFrequent = ch;
            }
        }
        return mostFrequent;
    }
}
```

Explanation:

- **if (count > maxCount)**: Tracks the character with the highest frequency.
- Updates the most frequent character during iteration.

IMPORTANT JAVA PROGRAMS BASED ON STRING

Convert the first letter of each word in a string to uppercase

```
public class FirstLetterUppercase {
    public static void main(String[] args) {
        String sentence = "quality assurance automation testing";
        String result = convertToUpperCase(sentence);
        System.out.println("Converted sentence: " + result);
    }

    public static String convertToUpperCase(String sentence) {
        StringBuilder result = new StringBuilder();
        boolean capitalize = true;

        for (char ch : sentence.toCharArray()) {
            if (capitalize && Character.isLetter(ch)) {
                result.append(Character.toUpperCase(ch));
                capitalize = false;
            } else {
                result.append(ch);
            }

            if (ch == ' ') {
                capitalize = true;
            }
        }

        return result.toString();
    }
}
```

Explanation:

- **boolean capitalize = true**: A flag to indicate when to capitalize a letter.
- **Character.toUpperCase(ch)**: Converts the character to uppercase if it is the first letter of a word.
- The flag is reset after every space character, allowing the first letter of the next word to be capitalized.
