Strings in Java

—----------

Assignment Questions

—----------

Q1. What is a String in java?

Ans:- In Java, a `String` is a sequence of characters. It's an immutable object, meaning once it's created, its content cannot be changed. Instead, any modification results in a new `String` object. The `String` class provides various methods to perform operations on strings such as substring, concatenation, trimming, replacing, and more. Java strings are stored as arrays of Unicode characters internally.

Example:
```java
String greeting = "Hello, World!";
```

Here, `greeting` is a string that contains the text "Hello, World!".

Q2. Types of string in java are?

Ans:- In Java, strings can be categorized based on how they are created and stored. The two main types are:

1. **String Literals**: These are sequences of characters in double quotes and are stored in a special area of memory called the string pool. This allows Java to optimize

resources; if a string literal with the same content is created again, no new memory is allocated. Instead, both references point to the same memory location.

```java
String str1 = "Hello";
```

2. **String Objects**: Created using the `new` keyword, these strings are stored in the heap memory. Each time you use the `new` keyword, a new memory location is created, even if the string content is identical to another string.

```java
String str2 = new String("Hello");
```

Of the two, string literals are more memory-efficient for commonly-repeated strings. However, both types use the `String` class and have access to the same methods.

Q3. in how many ways can you create string objects in java?

Ans:- In Java, there are primarily two ways to create string objects:

1. **Using String Literals**:
   When a string is created using double quotes, it's treated as a string literal. The Java String Pool (a pool in the heap memory)

stores these literals, ensuring that if a string with the same content is created again, no new memory allocation is done. Instead, the same reference is used.

```java
String str1 = "Hello";
```

2. **Using the `new` Keyword**:
   This approach creates a new string object in the heap memory, irrespective of the content. So even if the content is the same, a new memory location is always allocated.

```java
String str2 = new String("Hello");
```

While these are the primary ways, it's worth noting that strings can also be created/derived using methods such as `substring()`, which generate new string objects based on existing ones. Furthermore, classes like `StringBuilder` or `StringBuffer` can be used to construct strings and then convert them to string objects using the `toString()` method.

Q4. What is a string constant pool?

Ans:- In Java, the **String Constant Pool** refers to a special area in the heap memory where the JVM stores string literals. When a string literal is defined using double quotes, the JVM checks the String Constant Pool to see if an identical string already exists. If it does, the reference to the existing string is used; if not, a new string is created in the pool. This mechanism

optimizes memory usage and promotes efficient string handling.

For example:
```java
String str1 = "Hello";
String str2 = "Hello";
```

Both `str1` and `str2` will reference the same memory location in the String Constant Pool since they have the same content. This reusability makes operations with string literals more memory-efficient.

Q5. What do you mean by mutable and immutable objects?

Ans:- **Mutable objects** are those whose state or content can be changed after they are created. Modifying a mutable object will change its content without creating a new object in memory. Examples in Java include `StringBuilder`, `StringBuffer`, and most collection objects like `ArrayList`.

Immutable objects, on the other hand, cannot be modified once they are created. Any operation that seems to modify the content actually results in a new object being created, while the original object remains unchanged. In Java, the `String` class is a prominent example of an immutable object.

Advantages of immutability:
1. **Thread Safety**: Immutable objects are inherently thread-safe as their state cannot be changed.
2. **Security**: They can't be tampered with or accidentally changed.

3. **Clarity**: Reading code becomes easier since you know certain objects won't change state.

However, the creation of new objects for every modification in immutable classes can be less memory-efficient compared to modifying existing mutable objects.

Q6. Where exactly is the string constant pool located in the memory?

Ans:- The **String Constant Pool** is a dedicated location within the Java Heap memory. The Heap is where the runtime data area in the JVM allocates memory for all class instances and arrays. The String Constant Pool is a specific part of this heap memory where the JVM stores string literals, optimizing memory usage by reusing references to identical strings.

It's important to understand that while the String Constant Pool was traditionally located in the PermGen (Permanent Generation) space of the heap in older versions of Java, since Java 8, the PermGen space has been replaced with the Metaspace, and the String Constant Pool has been moved to reside within the main part of the heap memory. This change aids garbage collection and offers more flexibility in the management of the String Constant Pool.

Q7. What is Mutable string in java explain with an example?

Ans:- In Java, strings themselves (`String` objects) are immutable, which means their content cannot be changed after they are created. However, if you want to use a mutable version

of a string, Java provides `StringBuilder` and `StringBuffer` classes.

The primary difference between `StringBuilder` and `StringBuffer` is that `StringBuilder` is not thread-safe (faster for single-threaded operations), while `StringBuffer` is thread-safe (synchronized).

Here's a brief overview and example using `StringBuilder`, the mutable string in Java:

StringBuilder:
1. **Mutable**: You can modify the content without creating a new object.
2. **Not Thread-Safe**: Suitable for single-threaded scenarios.

Example:

```java
public class MutableStringDemo {
    public static void main(String[] args) {
        StringBuilder builder = new StringBuilder("Hello");
        builder.append(", World!");  // Modify the original StringBuilder object

        System.out.println(builder.toString());  // This will print "Hello, World!"
    }
}
```

Q8. WAP to reverse a string
Input: “ PWSKILLS”.

Ans:- Certainly! Here's a simple Java program that reverses the string "PWSKILLS":

```java
public class ReverseString {
    public static void main(String[] args) {
        String input = "PWSKILLS";
        String reversed = new StringBuilder(input).reverse().toString();
        System.out.println("Reversed String: " + reversed);  // This will print "SLLIKSWP"
    }
}
```

In this program, we use the `StringBuilder` class which has a built-in `reverse()` method. After reversing the string, we convert the `StringBuilder` back to a `String` using `toString()`.

Q9. WAP to reverse a sentence while preserving the position
Input: “Think twice”.

Ans:- To reverse a sentence while preserving the position of words, you can split the sentence into words, reverse each word individually, and then join them back together.

Here's a program that demonstrates this for the sentence "Think twice":

```java
public class ReverseSentence {
    public static void main(String[] args) {
        String input = "Think twice";

        // Splitting the sentence into words
        String[] words = input.split("\\s+");

        for (int i = 0; i < words.length; i++) {
            words[i] = new
StringBuilder(words[i]).reverse().toString();
        }

        // Joining the reversed words
        String result = String.join(" ", words);
        System.out.println(result);  // This will print "knihT ecwit"
    }
}
```

This program first splits the sentence into individual words. Each word is then reversed using `StringBuilder`. Finally, the reversed words are joined back together to form the output sentence.

Q10.WAP to sort a String Alphabetically?

Ans:- Certainly! Here's a Java program to sort a string alphabetically:

```java
public class SortStringAlphabetically {
    public static void main(String[] args) {
        String input = "PWSKILLS";
        char[] chars = input.toCharArray();

        // Sort the character array
        java.util.Arrays.sort(chars);

        // Convert the sorted character array back to a string
        String sorted = new String(chars);
        System.out.println(sorted);  // This will print "IKLLPSSW"
    }
}
```

In this program, we convert the string to a character array and then use Java's built-in `Arrays.sort()` method to sort the characters. After sorting, we convert the character array back to a string for the final output.

Q11. Write a simple String program to take input from the user.

Ans:- import java.util.Scanner;

```
public class StringInputProgram {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);  // Create
a Scanner object for user input

        System.out.println("Please enter a string:");

        String userInput = scanner.nextLine();  // Read user
input

        System.out.println("You entered: " + userInput);  //
Output user input

        scanner.close();
    }
}
```

Q12. How do you concatenate two Strings in java? Give an example?

Ans:- in java,concatenate two string using “ + ”
Operator, Here is an example.

```
public class ConcatenateStrings {
    public static void main(String[] args) {
        String str1 = "Hello";
```

```
        String str2 = "World";

        String concatenatedString = str1 + " " + str2; //
Concatenating two strings with a space in between

        System.out.println(concatenatedString);  // This will print
"Hello World"
    }
}
```

Q13. How do you find the length of a String in a java explain with an example?

Ans:- In Java, i can find the length of a string using the `length()` method of the `String` class.

Here's an example to demonstrate this:

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

        int length = str.length();  // Using the length() method to get
the length of the string

        System.out.println("Length of the string '" + str + "' is: " +
length);
        // This will print "Length of the string ‘ Vikash’ is: 6"
    }
}
```

```

In the example above, the `length()` method is called on the string `str` to get its length, which is then printed to the console.

## Q14. How do you compare two Strings in java? Give an example?

Ans:- In Java, strings can be compared in multiple ways:

1. **Using `equals()` Method**: This method checks the content of the strings for equality.

   ```java
   String str1 = "Hello";
   String str2 = "Hello";
   boolean isEqual = str1.equals(str2);  // true because content is the same
   ```

2. **Using `equalsIgnoreCase()` Method**: Compares two strings for equality, ignoring case differences.

   ```java
   String str1 = "Hello";
   String str2 = "hello";
   boolean isEqual = str1.equalsIgnoreCase(str2);  // true, case is ignored
   ```

Q15. Write a program to find the length of the string “refrigerator”.
   ```

Ans:- Here's a short program to find the length of the string "refrigerator":

```java
public class StringLength {
    public static void main(String[] args) {
        String word = "refrigerator";
        int length = word.length();

        System.out.println("The length of the string 'refrigerator' is: " + length);
    }
}
```

When i run this program, it will print: "The length of the string 'refrigerator' is: 12".

Q16. Write a program to check if the letter ‘e’ is present in the word ‘umbrella’.

Ans :- Certainly! Here's a succinct program to check if the letter 'e' is present in the word "umbrella":

```java
public class CheckLetter {
    public static void main(String[] args) {
        String word = "umbrella";

        if(word.contains("e")) {
```

```
            System.out.println("The letter 'e' is present in the word 'umbrella'.");
        } else {
            System.out.println("The letter 'e' is not present in the word 'umbrella'.");
        }
    }
}
```

When I run this program, it will print: "The letter 'e' is present in the word 'umbrella'."

Q17. Write a program to delete all consonants from the string “Hello, have a good day”.

Ans:- Here's a concise program to delete all consonants from the given string:

```java
public class RemoveConsonants {
    public static void main(String[] args) {
        String sentence = "Hello, have a good day";

        // Using a regular expression to replace all consonants with an empty string
        String result = sentence.replaceAll("[b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z]", "");

        System.out.println(result);
    }
}
```

When I run this program, it will print: "eoo, ae a ooa a".