1) What is mutable string in Java explain with an example.

Ans:

Mutable strings refer to strings that can be modified or changed after they are created.
This means we can modify individual characters, append or remove characters, or change the length of the string without creating a new string object.

Java provides the StringBuilder and StringBuffer classes for creating mutable strings
Example:

Certainly! Let's explore the concept of creating strings using `StringBuilder` and `StringBuffer` with examples:

1. Creating a string using `StringBuilder`:

```
StringBuilder stringBuilder = new StringBuilder();
stringBuilder.append("Hello");
stringBuilder.append(", ");
stringBuilder.append("World!");

System.out.println(stringBuilder);
```

In the example above, we use a `StringBuilder` object named `stringBuilder` to construct a string. We start with an empty `StringBuilder` and then use the `append()` method to add individual components of the string.

The output will be:

```
Hello, World!
```

2. Creating a string using `StringBuffer`:

```
StringBuffer stringBuffer = new StringBuffer();
stringBuffer.append("Hello");
stringBuffer.append(", ");
stringBuffer.append("World!");
```

System.out.println(stringBuffer);

In this example, we use a `StringBuffer` object named `stringBuffer` to construct a string. The process is similar to the `StringBuilder` example. We start with an empty `StringBuffer`, use the `append()` method to add components.
The output will be the:

Hello, World!

2) WAP to reverse string.

Ans:

```
public class Solution {
    public static void main(String[] args) {
        String originalString = "Hello";
        String reversedString = reverseString(originalString);
        System.out.println("Original String: " + originalString);
        System.out.println("Reversed String: " + reversedString);
    }

    public static String reverseString(String input) {
        char[] chars = input.toCharArray();
        int left = 0;
        int right = chars.length - 1;
        while (left < right) {
            char temp = chars[left];
            chars[left] = chars[right];
            chars[right] = temp;
            left++;
            right--;
        }
        return new String(chars);
    }
}
```

3) WAP to reverse a sentence while preserving the position.

Ans:

```
public class Demo {
    public static void reverse(char[] chars, int left, int right) {
        while (left < right) {
            char temp = chars[left];
            chars[left] = chars[right];
            chars[right] = temp;
            left++;
            right--;
        }
    }

    public static void main(String[] args) {

        String s = "Hello you met aty0";
        int i = 0;
        int len = s.length();
        char[] chars = s.toCharArray();
        while (i < len) {
            int index = s.indexOf(" ", i);

            int left = i;
            int right = index - 1;
            i = index + 1;

            if (index == -1) {
                i = s.lastIndexOf(" ");
                left = ++i;
                right = len - 1;
                reverse(chars, left, right);
                break;
            }

            reverse(chars, left, right);
        }
        String str = new String(chars);
        System.out.println("Reversed by word" + str);
    }
}
```

4) WAP to sort a string Alphabetically.

Ans:

```
public class Demo {
    public static void sort(char[] chars) {
        for (int i = 0; i < chars.length; i++) {

            for (int j = i + 1; j < chars.length; j++) {
                if ((int) chars[j] < (int) chars[i]) {
                    int temp = chars[i];
                    chars[i] = chars[j];
                    chars[j] = (char) temp;
                }
            }
        }

    }

    public static void main(String[] args) {

        String s = "edcbaaccbeebddeebb";
        char[] chars = s.toCharArray();
        sort(chars);
        String str = new String(chars);
        System.out.println("sorted string " + str);
    }

}
```