1.Remove unwanted spaces from sentence

```
package str;

public class Remove {

	public static void main(String[] args) {
		String str=" Ilakkiya,     Saadhana,     nesa  ";

			String[] words = str.trim().split("\\s+");

			StringBuilder remove = new StringBuilder();
			for (String word : words) {
			   remove.append(word).append(" ");
			}

			System.out.println("Original string: " + str);
			System.out.println("Remove string: " +
remove.toString().trim());
		}
	}
```

Output:
Original string: Ilakkiya,   Saadhana,   nesa
Remove string: Ilakkiya, Saadhana, nesa

2.remove duplicate characters from string

```
package str;

public class Duplicate {

   public static void main(String[] args) {
      String s = "Final keyword used for variables methods class";
      int length = s.length();
      s = s.toLowerCase();
      for (int i = 0; i < length; i++) {
```

```
            char crnt = s.charAt(i);


            if (crnt != ' ') {
               boolean duplicate = false;

               for (int j = i + 1; j < length; j++) {
                  if (crnt == s.charAt(j)) {
                     duplicate = true;

                     s = s.substring(0, j) + ' ' + s.substring(j + 1);
                  }
               }


               if (duplicate) {
                  System.out.print(crnt + " ");
               }
            }
         }
      }
}
```

Output:

f i a l e o r d s

3.remove white spaces from given string

```
package str;

public class WhiteSpace {

        public static void main(String[] args) {

                    String original= "  final  meth od cannot overri  de  .";

                    String white= original.replaceAll("\\s+", "");

                    System.out.println("Original string: " + original);
```

```
                System.out.println("Remove all white space: " +
white);
            }
        }
```

Output:
Original string: final meth od cannot overri de .
Remove all white space: finalmethodcannotoverride.

Interview Questions:

1. What is a mutable string in Java?

In a mutable string, we can change the value of the string and JVM doesn't create a new object. In a mutable string, we can change the value of the string in the same object. To create a mutable string in java, Java has two classes StringBuffer and StringBuilder where the String class is used for the immutable string.

2. Why StringBuffer objects in Java are mutable

StringBuffer is mutable, because its value can be changed after it is created.

3. What is the difference between length and capacity in Java StringBuffer?

Length:

- It is used to find the length of the string.
- By default, the length of an empty string is 0.
- It is the actual size of the string that is currently stored.
- One can count the length by simply counting the number of characters.

Capacity:

- It is used to find the capacity of StringBuffer.
- By default, an empty string buffer contains 16 character capacity.
- It is the maximum size that it can currently fit.
- The capacity can be calculated by counting the number of characters and adding 16 to it.

4. How will you add string in StringBuffer? Give an example.

StringBuffer. append(char[] astr) is the inbuilt method which appends the string representation of the char array argument to this StringBuffer sequence.

```
package str;

public class Append {

	public static void main(String[] args) {


			StringBuffer str = new StringBuffer("Hello ");

			// Append a string to the StringBuffer
			str.append("world!");

			// Print the updated StringBuffer
			System.out.println("Updated StringBuffer: " +
str.toString());
		}
	}
```

5. When will you use StringBuffer if String class is already available

String Modification: If you need to perform a lot of modifications (such as appending, inserting, or deleting characters) on a string,

using StringBuffer is more efficient than creating new strings with the String class. This is because StringBuffer is mutable, allowing you to modify the contents in-place without creating new objects.

Performance Considerations: When you need to concatenate multiple strings in a loop or perform many modifications to a string, using StringBuffer can be more efficient in terms of performance compared to repeatedly concatenating strings using the + operator or using the concat() method of the String class.