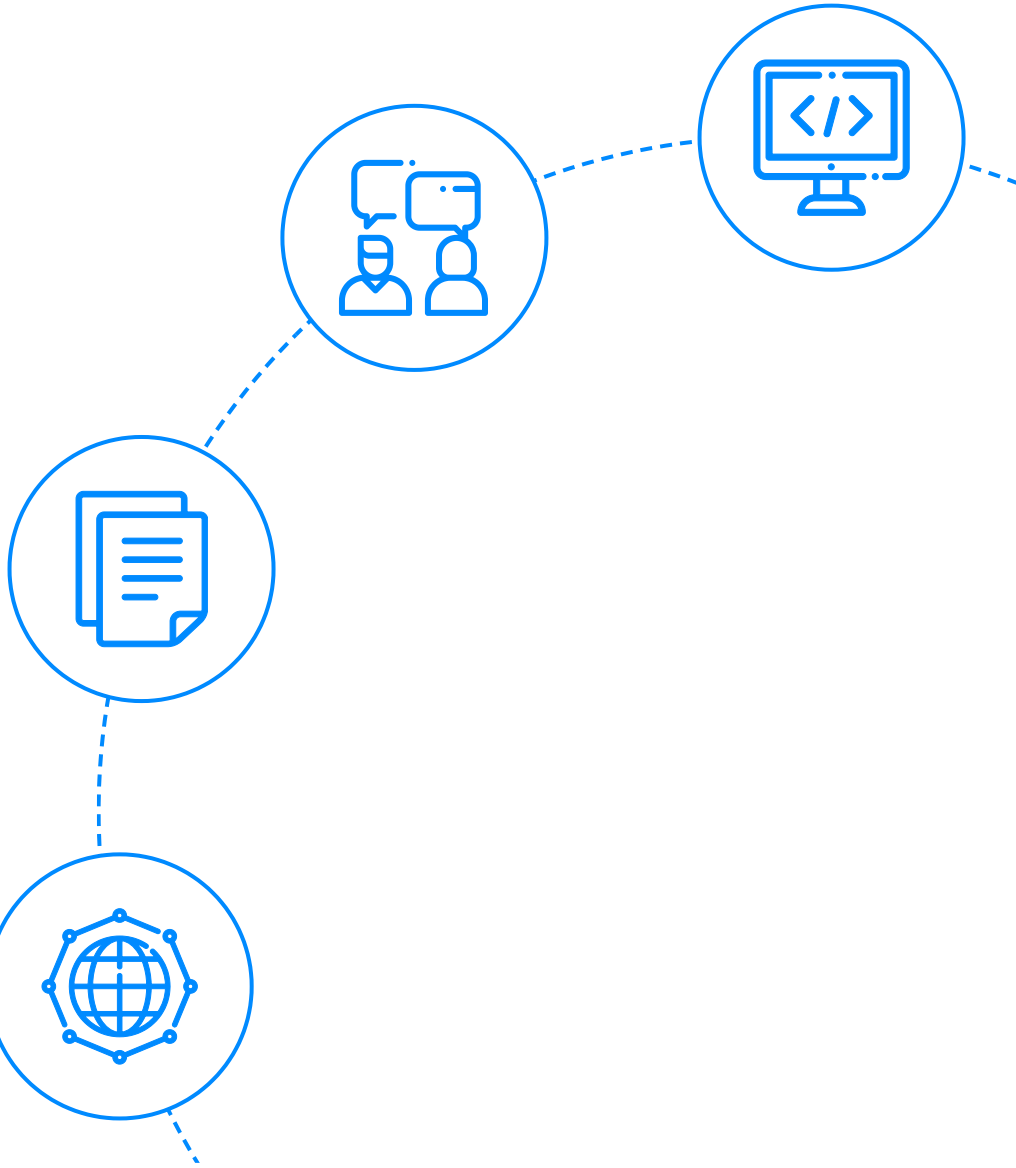
InterviewBit

Java String Interview Questions



To view the live version of the page, click here.

Contents

Java String Interview Questions for Freshers

Java String Interview Questions for Experienced

Java String Interview Questions for Experienced

(.....Continued)

19. Explain the string subSequence method.
20. What do you mean by StringJoiner?
21. How can a Java string be converted into a byte array?
22. In Java, how do you convert a string to an integer and vice versa?
23. How can we convert string to StringBuilder?
24. How do you check whether a String is empty in Java?
25. How many objects will be created for the following codes:

String Programming Questions

26. How to print all permutations of string in Java?
27. Write a program to calculate the total number of characters in the String?
28. How to reverse a string in Java?
29. How to convert an Array to String in Java?
30. Is it possible to count the number of times a given character appears in a String?
31. In what way should two strings be compared to determine whether they are anagrams?
32. How can we remove a specific character from a String?
33. Write a program to check whether the given input string is a palindrome.
34. What will be the output of the below program?
35. What is the output of the below program?

Let's get Started

Over the past 25 years, Java has been a popular programming language among all developers because of its user-friendly and flexible nature that can be used for platforms and web applications development. When it comes to Java interview questions, interviewers sometimes pay close attention to the Java string. The String class in Java is among the fundamentals of **Java programming** and therefore, knowledge of String is a prerequisite for every Java programmer. Whether it's a Java desktop application, enterprise application, web application, or mobile application, every Java application makes use of the String class. It is therefore one of the hottest and most important topics in Java interviews.

In this article, we have compiled a comprehensive list of insightful Java String Interview Questions for both Freshers and Experienced that focus on a range of topics including thread-safety, immutability, string methods in Java, StringBuilder and StringBuffer, memory consumption, comparing String instances in Java, using String as the key in HashMap, equals() vs == check for String, etc. These questions will be a great help to know about the String concept in detail and be prepared to tackle String-related questions during a **Java technical interview**.

Before we begin, let's have a quick look at what Java String is all about.

What is String in Java?

Strings, one of the most common objects used in Java programming, are essentially sequences of characters. As an example, the string "Scaler" contains the following characters: "S", "c", "a", "l", "e", and "r". You can either create a string by using String Literal or by using the NEW keyword. Additionally, String supports a variety of methods to operate on Strings, such as the equals method to compare two Strings, the replace method to replace String characters, the substring method to get a substring, the toUpperCase method to convert String to upper case, the split method to split a long String into multiple Strings, and so on.

Now let's look at the most common asked String Interview questions:

Java String Interview Questions for Freshers

1. How to declare a string in Java?

String declaration in Java can be done in two ways:

- **By string literal**: Double quotes are used to create Java String literals.
  - Example: `String str= "Scaler";`
- **By new keyword**: Keyword "new" is used to create a Java string.
  - Example: `String str=new String ("Scaler");`

2. Is String a primitive or derived type in Java?

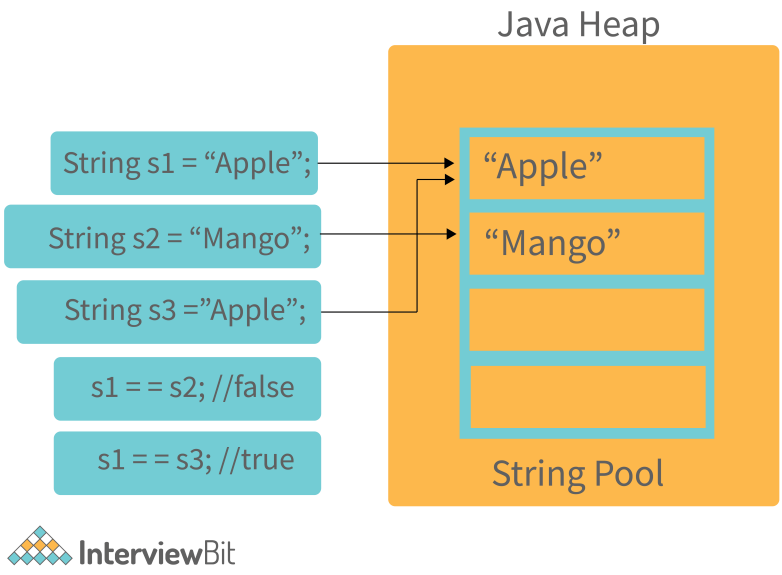
Strings are derived data types. Strings are Java objects that represent sequences of characters. String objects are created using the java.lang.String class. There are many functions that need to be called upon when processing a string, such as substring(), indexof(), equals(), toUppercase(), etc, which primitives types do not have.

3. State the difference between String in C and String in Java.

- **String in C**: In C, strings are just arrays of characters, and they are terminated with a /0, which is why we commonly refer to them as "null-terminated". Strings in C, like "abc$%", actually consist of 6 characters 'a' 'b' 'c' '$' '%' and '/0', but these can be easily manipulated.
- **String in Java**: Java treats Strings as objects, not arrays. String objects are created using the java.lang.String class. String objects in Java are immutable; you cannot modify their contents. This means whenever we manipulate a String object, the new String is created rather than the original string being modified.

4. Explain String pool in Java.

String Pool, also known as SCP (String Constant Pool), is a special storage space in Java heap memory that is used to store unique string objects. Whenever a string object is created, it first checks whether the String object with the same string value is already present in the String pool or not, and if it is available, then the reference to the string object from the string pool is returned. Otherwise, the new string object is added to the string pool, and the respective reference will be returned.



As shown in the above image, two Strings s1 and s2 are created with the values "Apple" and "Mango". Therefore, when the third String s3 containing the value "Apple" is created, instead of creating a new object, the existing object reference will be returned. Here, s1==s2 is false both strings s1 and s2 refer to different string values from the string pool i.e. apple and mango. We can see that s1==s3 is true because both strings s1 and s3 refer to a single string value from a string pool i.e., apple.

5. Is String immutable or final in Java? If so, then what are the benefits of Strings being Immutable?

Yes, Strings are immutable in Java. Immutable objects mean they can't be changed or altered once they've been created. However, we can only modify the reference to the string object. The String is immutable in Java because of many reasons like security, caching, synchronization and concurrency, and class loading.

6. What does the string intern() method do in Java?

If you apply the intern() method to a few strings, you will ensure that all strings having the same content share the same memory. As soon as a String object is invoked with intern(), it first checks if the string value of the String object is already present in the string pool and if it is available, then the reference to that string from the string constant pool is returned. If not, a new string object is added to the string pool, and a reference to it is returned.

Example:

```
String str1 = new String("Scaler by InterviewBit").intern();  //Line1
String str2 = new String("Scaler by InterviewBitt").intern(); //Line2
System.out.println(str1 == str); //prints true
```

As you can see, the intern() method is invoked on the String objects. When Line1 is executed, memory is allocated within the SCP. In line 2, no new string objects are created in the SCP because str1 and str2 have the same content. As a result, the reference to the object created in line1 is returned. This means that str1 and str2 both point to the same memory. Therefore, the print statement prints true.

7. State the difference between String and StringBuffer.

String objects in Java are immutable and final, so we can't change their value after they are created. Since strings are commonly used in applications, we need to perform several operations on them such as substring(), equals(), indexof(), toUppercase(), etc. Each time we manipulate a string, a new String object is created, and all previous objects will be garbage, placing a strain on the garbage collector. This is why The Java team developed StringBuffer. A StringBuffer is a mutable object, meaning it can be changed, but the string is an immutable object, so it cannot be changed once it has been created.

- **String**

Syntax:

```
String str1="InterviewBit";
String str2=new String("Scaler");
Scanner str3=new Scanner(System.in);
String str4=str3.nextLine();
```

Example: Concatenation Example of String. A string class takes longer to perform a concatenation operation than a string buffer class.

```
public class Scanner
{
    public static void main(String []args)
    {
       StringBuilder stbu=new StringBuilder();
       //Initial object size
       System.out.println(stbu.capacity());
       String str="Scaler";
       System.out.println(str);
       String str1 = new String("InterviewBit");
       System.out.println(str1);
       str1 += " Articles";         //string update
       System.out.println(str1);
    }
}
```

Output:

```
16
Scaler
InterviewBit
InterviewBit Articles
```

- **StringBuffer**

Syntax:

```
StringBuffer var = new StringBuffer(str);
```

Example: Concatenation Example of StringBuffer. String buffer class perform concatenation operations more quickly than string classes.

```
public class StringBuffer
{
    public static void main(String []args)
    {
       StringBuilder stbu=new StringBuilder();
       //Initial object size
       System.out.println(stbu.capacity());
       StringBuffer stbr= new StringBuffer("InterviewBit");
       System.out.println(stbr);
       stbr.append(" Articles");       //string update
       System.out.println(stbr);
       stbr=new StringBuffer("Scaler");
       System.out.println(stbr);
    }
}
```

Output:

```
16
InterviewBit
InterviewBit Articles
Scaler
```

8. State the difference between StringBuffer and StringBuilder in Java.

StringBuffer and StringBuilder are two Java classes for manipulating strings. These are mutable objects, i.e., they can be modified, and provide various methods such as insert(), substring(), delete(), and append(), for String manipulation.

- **StringBuffer**: The StringBuffer class was created by the Java Team when they realized the need for an editable string object. Nevertheless, StringBuffer has all methods synchronized, meaning they are thread-safe. Therefore, StringBuffer allows only one thread to access a method at once, so it is not possible to call StringBuffer methods from two threads simultaneously, which means it takes more time to access. The StringBuffer class has synchronized methods, making it thread-safe, slower, and less efficient than StringBuilder. The StringBuffer class was introduced in Java 1.0.
  - Syntax:

```
StringBuffer var = new StringBuffer(str);
```

- **StringBuilder**: It was at that point that the Java Team realized that making all methods of StringBuffer synchronized wasn't the best idea, which led them to introduce StringBuilder. The StringBuilder class has no synchronized methods. Unlike StringBuffer, StringBuilder does not offer synchronized methods, which makes it less thread-safe, faster, and more efficient. StringBuilder was introduced in Java 1.5 in response to StringBuffer's shortcomings.
  - Syntax:

```
StringBuilder var = new StringBuilder(str);
```

9. In Java, how can two strings be compared?

In Java, there are several ways for comparing two strings. The following are a few of them:

- **String Equals Method**: In this method, the strings are compared based on the values within them. If the values of the two strings are the same, it returns true; otherwise, it returns false. This method is case-sensitive.

Syntax:

```
str1.equals(str2);
```

For example:

```
Input 1= Scaler
Input 2= InterviewBit
Output= false

Input 1= Scaler
Input 2= Scaler
Output= true

Input 1= Scaler
Input 2= scaler
Output= false
```

- **String Equals Ignore Case**: By using this method, the two strings are compared without taking into account the case (upper or lower). It returns true if the two values are the same and not null.

  Syntax:

```
str1.equalsIgnoreCase(str2);
```

For Example:

```
Input 1= Scaler
Input 2= InterviewBit
Output= false

Input 1= Scaler
Input 2= Scaler
Output= true

Input 1= Scaler
Input 2= scaler
Output= true
```

- **Object Equals Method**: The method returns true if its arguments are equal, otherwise, it returns false. Accordingly, if both arguments are null, the result is true, and if just one argument is null, the result is false.

Syntax:

```
Object.equals(str1, str2)
```

For example:

```
Input 1= Scaler
Input 2= InterviewBit
Output= false

Input 1= Scaler
Input 2= Scaler
Output= true

Input 1= Scaler
Input 2= null
Output= false

Input 1= null
Input 2= null
Output= True
```

- **String Compare To Method**: This method compares input strings with each other. Upon comparison, the following value is returned:

1. If (str1>str2), a positive value is returned.
2. If (str1==str2), 0 is returned.
3. If (str1<str2), a negative value is returned.

Syntax:

```
str1.compareTo(str2)
```

Example:

```
Input 1= InterviewBit
Input 2= Scaler
Output= -10

Input 1= Scaler
Input 2= Scaler
Output= 0

Input 1= Scaler
Input 2= InterviewBit
Output= 10
```

10. What is the difference between str1 == str2 and str1.equals(str2)?

Java offers both the equals() method and the "==" operator for comparing objects. However, here are some differences between the two:

- Essentially, equals() is a method, while == is an operator.
- The == operator can be used for comparing references (addresses) and the .equals() method can be used to compare content. To put it simply, == checks if the objects point to the same memory location, whereas .equals() compares the values of the objects.

Example:

```
public class StringComparison
{
   public static void main(String[] args)
   {
       String str1=new String("Scaler");
       String str2=new String("Scaler");
       System.out.println(str1 == str2);
       System.out.println(str1.equals(str2));
   }
}
```

Output:

```
false
true
```

In this example, two different String objects are being created, str1 and str2.

- If str1 and str2 are compared using the == operator, then the result will be false, because both have different addresses in the memory. Both must have the same address in the memory for the result to be true.
- If you use the equals method, the result is true since it's only comparing the values given to str1 and str2, even though they are different objects.

11. Is it possible to compare Strings using the == operator? If so, what is the risk involved?

Yes, you can compare strings using the == operator. One can use == operators for reference comparison (address comparison). The majority of the time, developers compare strings with the == operator, instead of using the equals() method, resulting in an error.

Example:

```
public class StringComparison
{
    public static void main(String args[])
   {
       String str1="Scaler";
       String str2="Scaler";
       String str3=new String("Scaler");
       System.out.println(str1==str2);
       //true because both points to same memory allocation

      System.out.println(str1==str3);
      //false because str3 refers to instance created in heap

      System.out.println(str1.equals(str3));
      //true because both share same content
      //even if both are different string objects
}
}
```

Output:

```
true
false
true
```

12. What is the use of the substring() method in Java?

The substring method is used to return substring from a specified string. This method takes two parameters i.e., beginIndex (the starting index) and endIndex (the ending index). In the case of substring(), method startIndex is inclusive and endIndex is exclusive.

Syntax:

```
substring(int beginIndex, int endIndex)
```

Or

```
substring(int beginIndex)
```

Here,

- **beginIndex**: Index that marks the starting of subsequence and it is inclusive.
- **endIndex**: Index that marks the ending of subsequence and it is exclusive.

Example:

```
import java.lang.Math;
public class InterviewBit
{
   // driver code
   public static void main(String args[])
   {
        String str = "Scaler by InterviewBit";

       //prints substring from 7th index
        System.out.print("Returns: ");
        System.out.println(str.substring(7));
        // prints substring from 0-6, exclusive 6th index
        System.out.print("Returns: ");
        System.out.println(str.substring(0, 6));
       // prints the substring from 10-22, exclusive 22th index
        System.out.print("Returns: ");
        System.out.println(str.substring(10, 22));
   }
   }
```

Output:

```
Returns: by InterviewBit
Returns: Scaler
Returns: InterviewBit
```

13. Can we use a string in the switch case in java?

Yes, Java allows you to use strings in switch case conditions. Below is a Java program that shows the use of string in switch case.

Example:

```
public class StringinSwitchCase
{
   public static void main(String[] args)
   {
       String fruit = "Apple";
       switch(fruit)
       {
           case "Mango":
               System.out.println("Sweet");
               break;
           case "Apple":
               System.out.println("Delicious");
               break;
           case "Orange":
               System.out.println("Luscious");
               break;
           default:
               System.out.println("Not a fruit");
       }
   }
}
```

Output:

```
Delicious
```

Java String Interview Questions for Experienced

14. What are the different string methods in Java?

There are various string operations in Java that allow us to work with strings. These methods or operations can be used for string handling in Java as well as string manipulation in Java. Some of such methods are as follows:

- **split()**: Split/divide the string at the specified regex.
- **compareTo()**: Compares two strings on the basis of the Unicode value of each string character.
- **compareToIgnoreCase()**: Similar to compareTo, but it also ignores case differences.
- **length()**: Returns the length of the specified string.
- **substring()**: Returns the substring from the specified string.
- **equalsIgnoreCase()**: Compares two strings ignoring case differences.
- **contains()**: Checks if a string contains a substring.
- **trim()**: Returns the substring after removing any leading and trailing whitespace from the specified string.
- **charAt()**: Returns the character at specified index.
- **toLowerCase()**: Converts string characters to lower case.
- **toUpperCase()**: Converts string characters to upper case.
- **concat()**: Concatenates two strings.

15. Is String thread-safe in Java?

Strings are immutable objects, which means they can't be changed or altered once they've been created. As a result, whenever we manipulate a String object, it creates a new String rather than modifying the original string object. In Java, every immutable object is thread-safe, which means String is also thread-safe. As a result, multiple threads can access a string. For instance, if a thread modifies the value of a string, instead of modifying the existing one, a new String is created, and therefore, the original string object that was shared among the threads remains unchanged.

16. Why is a string used as a HashMap key in Java?

Basically, the HashMap object can store key-value pairs. When creating a HashMap object and storing a key-value pair in that object, you will notice that while storing, the hash code of the key will be calculated, and its calculated value will be placed as the resultant hash code of the key. Now, when the key is passed to fetch its value, then the hash code of the key is calculated again, and if it's equal to the value of the hash code initially calculated, the initial value placed as the resultant hash code of the key is retrieved or fetched.

Let's say we utilized a variable as a key to store data and then changed the value of that variable. In this case, since we have altered the key, the hash code calculated of the current key will not match the hash code at which its value was originally stored. This makes retrieval impossible. String values are immutable, so once they've been created, they can't be changed. As a result, it is recommended to use Strings as HashMap keys.

17. What is the best way to split a string in Java?

Split() is a Java method for breaking a string based on a Java string delimiter (specified regex). For example, a space or a comma(,) will usually be used as the Java string split attribute to break or split the string.

Syntax:

```
string.split(String regex, int limit)
```

Here,

- **regex**: String is divided at this specified regex.
- **limit (optional parameter)**: Controls or limits the number of resulting substrings. Split() returns all potential substrings if the limit parameter is not specified or is 0.

Example:

```
public class SplitString
{
 public static void main(String[] args)
 {
   String str = "Scaler by InterviewBit";
   // split string from space
   String[] result = str.split(" ");
   for (int i=0; i < result.length; i++)
   {
     System.out.println(result[i]);
   }
 }
}
```

Output:

```
Scaler
by
InterviewBit
```

18. Why char array is preferred over a String in storing passwords?

There are various reasons why a char array rather than a string should be used to store passwords. The following are a few of them:

- **Strings are immutable**: The content of Strings cannot be modified/overwritten because any modification will result in the creation of a new String. As a result, we should always save sensitive data like passwords, Social Security numbers, and so on in a char[] array rather than a String.
- **Security**: Because String is immutable, storing the password as plain text keeps it in memory until it is cleaned up by the garbage collector. As string uses SCP (String Constant Pool) for re-usability of a string, it's possible that it'll remain in memory for a long time, and anyone with access to the SCP or memory dump can simply identify or retrieve the password in plain text. That's another reason why we should use an encrypted password instead of plain text.
- **Logfile safety**: With an array, the data can be erased or wiped up, overwritten and the password will not be present anywhere in the system. Whereas, when using plain String, the chances of mistakenly printing the password to monitors, logs, or other insecure locations are substantially higher.

19. Explain the string subSequence method.

The Java String subSequence() method is a built-in function that returns a charSequence (a subsequence) from a string.

20. What do you mean by StringJoiner?

StringJoiner is a Java class that allows you to construct or create a sequence of strings (characters) that are separated by delimiters like a hyphen(-), comma(,), etc. Optionally, you can also pass suffix and prefix to the char sequence.

Example:

```
// importing StringJoiner class
import java.util.StringJoiner;
public class ExampleofStringJoiner
{
   public static void main(String[] args)
   {
       StringJoiner joinStrings = new StringJoiner(",", "[", "]");
       // passing comma(,) and square-brackets as delimiter

       // Adding values to StringJoiner
       joinStrings.add("Scaler");
       joinStrings.add("By");
       joinStrings.add("InterviewBit");
       System.out.println(joinStrings);
   }
}
```

Output:

```
[Scaler,By,InterviewBit]
```

21. How can a Java string be converted into a byte array?

The getBytes() method allows you to convert a string to a byte array by encoding or converting the specified string into a sequence of bytes using the default charset of the platform. Below is a Java program to convert a Java String to a byte array.

Example:

```
import java.util.Arrays;
public class StringToByteArray
{
   public static void main(String[] args)
   {
      String str = "Scaler";
 byte[] byteArray = str.getBytes();
  // print the byte[] elements
     System.out.println("String to byte array: " + Arrays.toString(byteArray));
   }
}
```

Output:

```
String to byte array: [83, 99, 97, 108, 101, 114]
```

22. In Java, how do you convert a string to an integer and vice versa?

There is an Integer class in the Java lang package that provides different methods for converting strings to integers and vice versa. The parseInt() method allows you to convert a String into an integer and the toString() method allows you to convert an Integer into a String. Below is a Java program to convert a string to an integer and vice versa.

Example:

```
public class StringtoInteger {
  public static void main(String args[])
  {
     String str1 = "1296";
     int  i= Integer.parseInt(str1);
     System.out.println(i);
     String str2 = Integer.toString(i);
     System.out.println(str2);
  }
}
```

Output:

```
1296
1296
```

23. How can we convert string to StringBuilder?

The append() method can be used to convert String to StringBuilder, and the toString() method can be used to convert StringBuilder to String. Below is a Java program to convert a string array to one StringBuilder object using the append method.

Example:

```
public class StringToStringBuilder {
  public static void main(String args[]) {
     String strs[] = {"Scaler", "by", "InterviewBit!"};
     StringBuilder sb = new StringBuilder();
     sb.append(strs[0]);
     sb.append(" "+strs[1]);
     sb.append(" "+strs[2]);
     System.out.println(sb.toString());
  }
}
```

Output:

```
Scaler by InterviewBit!
```

24. How do you check whether a String is empty in Java?

The Java String class contains a particular method for determining whether or not a string is empty. The isEmpty() method determines whether or not a string has zero length. In the case where the length of the string is zero, it returns true, or else it returns false.

Example:

```
public class StringEmpty
{
   // Function to determine if String is empty
   public static boolean isStringEmpty(String str)
   {
       //Use the isEmpty() method
       //to determine if the string is empty.
       if (str.isEmpty())
           return true;
       else
          return false;
   }
   public static void main(String args[])
   {
       String str1="InterviewBit";     //non-empty string
       String str2="";                 //empty string
       System.out.println("Str1 \"" + str1 + "\" is empty? " + isStringEmpty(str1));
       System.out.println("Str2 \"" + str2 + "\" is empty? " + isStringEmpty(str2));
  }
}
```

Output:

```
Str1 "InterviewBit" is empty? false
Str2 "" is empty? true
```

25. How many objects will be created for the following codes:

A.

```
String str1 = "abc";                          //Line1
String str2 = new String("abc");              //Line2
```

B.

```
String str1 = "abc";                          //Line1
String str2 = "abc";                          //Line2
```

C.

```
String str1 = new String("abc");                    //Line1
String str2 = new String("abc");                    //Line2
```

- **For A:** In this case, two objects will be created. We know that whenever a Java string is created using a new keyword, then two objects will be created i.e. one in the Heap Area and another one in the String constant pool. When the line1 is executed, the new string object str1 gets created and stored in the string constant pool. However, when line2 is executed, only one object is created using a new operator that gets stored in the heap memory (str2). This is because String constant pool already has a String object with the same string value (abc), and therefore, the reference of the string str1 from the string constant pool is returned.
- **For B:** In this case, one object will be created. Here, for line1 (str1), one new object will get created in String constant pool, whereas for line 2, string str2 will create a reference to the String str1 because the string constant pool already has a String object str1 with the same string value (abc).
- **For C:** In this case, three objects will be created. In the case of line1 (str1), two objects are created, one in the string constant pool and one in the heap memory. As for line 2 (str2), one new object is created and stored in heap memory, but not in the string constant pool because a String constant pool object str1 already has the string object str1 with the same string value (abc).

String Programming Questions

26. How to print all permutations of string in Java?

The term "permutation of the string" refers to all of the conceivable new strings that can be created by swapping the positions of the given string's characters. For example, the string CAT has a total of 6 permutations i.e., [CAT, CTA, ACT, ATC, TCA, TAC]. Below is a Java program to print all permutations of a given string.

```
public class InterviewBit {

   // Function to display all permutations of the string str
   static void printallPermutns(String str, String str2)
   {
       // check if string is empty or null
       if (str.length() == 0)
         {
           System.out.print(str2 + " ");
           return;
         }

       for (int i = 0; i < str.length(); i++)
         {
           // ith character of str
           char ch = str.charAt(i);
           // Rest of the string after excluding
           // the ith character
           String str3 = str.substring(0, i) + str.substring(i + 1);
           // Recursive call
           printallPermutns(str3, str2 + ch);
        }
   }
   // Driver code
   public static void main(String[] args)
   {
       String s = "cat";
       printallPermutns(s, "");
   }
}
```

Output:

```
cat cta act atc tca tac
```

27. Write a program to calculate the total number of characters in the String?

To find the total count of characters in a specified string, we can use the for loop, while loop, or do while loop. Below is a Java program to calculate the total number of characters in a string using for loop.

```
public class TotatCharacters
{
    public static void main(String[] args)
    {
       String str = "Scaler by InterviewBit";
       int count = 0;
       System.out.println("Input String: "+str);

       //Count total characters in the given string except space
       for(int i = 0; i < str.length(); i++)
          {
             if(str.charAt(i) != ' ')
             count++;
          }

       //Display total number of characters in the given string
       System.out.println("The total number of characters in the given string: " + cour
    }
}
```

Output:

```
Input String: Scaler by InterviewBit
The total number of characters in the given string: 20
```

28. How to reverse a string in Java?

There are different ways to reverse a string in Java-like using CharAt() method, StringBuilder/StringBuffer Class, Reverse Iteration, etc. The reverse() method is available in both the StringBuilder and StringBuffer classes and is commonly used to reverse a string. The reverse () method simply reverses the order of the characters. Below is a Java program to reverse a string using the StringBuilder class.

```
public class ReverseString
{
     // function to reverse a string using StringBuilder
     public static String revstr(String str)
     {
          return new StringBuilder(str).reverse().toString();
     }
     public static void main(String[] args)
     {
         String str= "Scaler by InterviewBit";
         str= revstr(str);
         System.out.println("Result after reversing a string is: "+ str);
    }
}
```

Output:

```
Result after reversing a string is: tiBweivretnI yb relacS
```

29. How to convert an Array to String in Java?

An array can be converted to a string in four different ways such as Arrays.toString() method, String.Join() method, StringBuilder.append() method, and Collectors.joining() method. Here, we will see an example of the Array.toString() method. Arrays.toString() returns a string representation of the array contents. The string represents the array's elements as a list, enclosed in square brackets ("[]"). The characters ", " (a comma) followed by a space are used to separate adjacent elements. It returns “null” if the array is null.

```
import java.util.Arrays;
public class ArrayToString
{
     public static void main(String[] args)
     {
          String[] strArray = { "Scaler", "by", "InterviewBit"};
          String str1 = ConvertArraytoString(strArray);
       System.out.println("An array converted to a string: " + str1);
     }
     // Using the Arrays.toString() method
     public static String ConvertArraytoString(String[] strArray)
     {
       return Arrays.toString(strArray);
     }
}
```

Output:

```
An array converted to a string: [Scaler, by, InterviewBit]
```

30. Is it possible to count the number of times a given character appears in a String?

The charAt() method of the String class can be used to find out the number of times a specified character appears in a string. Below is a Java program to check the occurrences of a specified character in a string.

```
public class CkeckforOccurences
{
     public static void main(String[] args)
     {
         String str= "InterviewBit";
         char ch = 'e';  //character to look for occurrences is e
         int count = 0;
         for (int i = 0; i < str.length(); i++)
           {
           if (str.charAt(i) == ch)
               {
               count++;
               }
           }
       System.out.println("The character '" + ch + "' appears " + count + " times in the
       }
}
```

Output:

```
The character 'e' appears 2 times in the given string 'InterviewBit'.
```

As you can see, the above program checks how many times the character ch occurs in the string str. Whenever we encounter the character ch in the string, we increase the count by one. Finally, we print the total character count at the end.

31. In what way should two strings be compared to determine whether they are anagrams?

If two strings contain the same characters but in a different order, they can be said to be anagrams. Consider dusty and study. In this case, dusty's characters can be formed into a study, or study's characters can be formed into dusty. Below is a java program to check if two strings are anagrams or not.

```
import java.util.Arrays;
public class CheckAngagram
{
   public static void main(String[] args)
    {
        String str1 = "Bored";
        String str2 = "Robed";

        //Convert strings to lowercase
        str1 = str1.toLowerCase();
        str2 = str2.toLowerCase();
        // Check to see if the lengths are the same
        if(str1.length() == str2.length())
        {
             // convert strings into char array
             char[] str1charArray = str1.toCharArray();
             char[] str2charArray = str2.toCharArray();
             // sort the char array
             Arrays.sort(str1charArray);
             Arrays.sort(str2charArray);
             // if the sorted char arrays are same or identical
             // then the strings are anagram
             boolean result = Arrays.equals(str1charArray, str2charArray);
             if(result)
             {
                System.out.println(str1 + " and " + str2 + " are anagrams of each other
              }
             else {
                System.out.println(str1 + " and " + str2 + " are not anagrams of each o
              }
        }
        else {
           System.out.println(str1 + " and " + str2 + " are not anagrams of each other.
        }
    }
}
```

Output:

```
bored and robed are anagrams of each other.
```

In the above program, there are two strings i.e., str1 and str2. Here, we are comparing str1 and str2 to determine if they are anagrams. The strings are first converted to lowercase since Java is case-sensitive and B and b are two different characters in Java. Here,

- **str1.toCharArray()**: Convert or transform the string into a char array.
- **Arrays.sort()**: It sorts the char arrays.
- **Arrays.equals()**: Checks or verifies if sorted char arrays are equal.

In the case where sorted arrays are equal, the strings are anagrams.

32. How can we remove a specific character from a String?

There are several ways to remove a character from a string, such as removing the character at the beginning of the string, the end of the string, or at a specific position. It is possible to remove a specific character from a string using the replace() method. You can also use remove() with different variations like replaceFirst(), replaceAll(), etc. Below is a Java program that uses replace(), replaceFirst(), and replaceAll() methods to remove characters from a String.

```
public class RemoveCharacter
{
  public static void main(String args[])
    {
      String str = "Scaler by InterviewBit";

      //remove the specified character.
      System.out.println("String after removing 'e' = "+str.replace("e", ""));
     //remove the first occurrence of the specified character.
     System.out.println("String after removing First 'e' = "+str.replaceFirst("e", ""));

    //remove all occurrences of the specified character.
  System.out.println("String after replacing all small letters = "+str.replaceAll("([A-
  }
}
```

Output:

```
String after removing 'e' = Scalr by IntrviwBit
String after removing First 'e' = Scalr by InterviewBit
String after replacing all small letters = caler by nterviewit
```

33. Write a program to check whether the given input string is a palindrome.

When a string is the same when read right to left or left to right, it is called a palindrome. To assess whether a string is a palindrome or not, we first reverse the string and then compare the reversed string with the original one. Below is a Java program that will check if a string is a palindrome.

```
public class PalindromeChecker
{
    public static void main (String[] args)
    {
       String str1 = "rotator";
       String revstr = reverseString(str1); //revstr=reverse string
       if (str1.equals(revstr))
          {
              System.out.println("The string" + str1 + " is a Palindrome String.");
          }
        else
          {
           System.out.println("The string" + str1 + " is not a Palindrome String.");
          }
    }
    // a method for reversing a string
    public static String reverseString(String str2)
    {
        String revstr = "";
        for (int i = str2.length() - 1; i >= 0; i--)
         {
            revstr += str2.charAt(i);
         }
            return revstr;
     }
}
```

Output:

```
The string rotator is a Palindrome String.
```

As shown in the above example, we have a string "Rotator" stored in string object "str1" and another string object "revstr" to store the reverse of str1. To check whether two strings are equal or not, we have used the equals() method.

34. What will be the output of the below program?

```
String str1 = "scaler";                          //Line1
String str2 = new String("scaler");      //Line2
str2.intern();                                        //Line 3
System.out.println(str1 == str2);
```

The output of the above program is false. We know that when the intern() method is executed or invoked on a string object, then it checks whether the String pool already has a same string value (scaler) or not, and if it is available, then the reference to the that string from the string constant pool is returned. In the above example, the intern method is invoked on str2. However, since we didn't assign it back to str2, str2 remains unchanged and therefore, both str1 and str2 have different references. If we change the code in line 3 to str2 = str2.intern(), then the output will be true.

35. What is the output of the below program?

```
public class StringTest
{
public static void main(String[] args)
          {
 String str1 = new String("interviewbit");
 String str2 = new String("INTERVIEWBIT");
 System.out.println(str1 = str2);
}
}
```

This program prints "INTERVIEWBIT" since str2 String is assigned to str1 String. The comparison operator “==” should not be confused with the assignment operator “=”.

Conclusion

Here's everything you need to know about Java String interview questions and answers. To summarize, there are many specifics related to String that every Java programmer needs to be familiar with and these String questions will not just help you prepare better for Java interviews, but will also open a new door to learning more about String.

The more familiar you are with these frequently asked interview questions, the greater your chances of getting hired.

Hopefully, we were able to answer any questions or concerns you had. Wishing you luck in your future endeavours.

Recommended Interview Preparation Resources

- Java Projects With Source Code
- Java MCQ With Answers
- Java Interview Questions for 5 years Experience
- Technical Interview Questions
- Coding Interview Questions
- InterviewBit Blog
- DSA

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions