**Strings Question and Answers**

**Q1.  What is String?**  
String is a class in java which is present in java.lang package. The String class represents character strings. Strings are constant, their values can not be changed after they are created.  
 **Q2  Is String immutable in java?**  
  
Yes, String class is immutable in java. Immutable means once the object is created, its value can not be changed.  
 **Q3 Is String a keyword in java ?**

No, String  is not a keyword in java.  
  
**Q4 How many objects are created using *new* keyword?**

String str = new String("JavaHungry");

Two objects are created by the above statement. One object in the heap memory and one object in the String constant pool.  
 **Q5 Write a program to reverse a String in java ?**

**sol 1)**

**import** **java.io.\***;

**import** **java.util.\***;

**public** **class** **reverseString** {

**public** **static** **void** **main**(String[] args) {

String input="AliveisAwesome";

StringBuilder input1 = **new** StringBuilder();

input1.append(input);

input1=input1.reverse();

**for** (**int** i=**0**;i<input1.length();i++)

System.out.print(input1.charAt(i));

}}

**sol 2)**

**import** **java.io.\***;

**import** **java.util.\***;

**public** **class** **reverseString** {

**public** **static** **void** **main**(String[] args) {

String input = "Be in present";

**char**[] temparray= input.toCharArray();

**int** left,right=**0**;

right=temparray.length-**1**;

**for** (left=**0**; left < right ; left++ ,right--)

{

// Swap values of left and right

**char** temp = temparray[left];

temparray[left] = temparray[right];

temparray[right]=temp;

}

**for** (**char** c : temparray)

System.out.print(c);

System.out.println();

}}

**sol3)**

**import** **java.util.\***;

**import** **java.lang.\***;

**import** **java.io.\***;

**public** **class** **reverseString**{

**public** **static** **void** **main**(String[] args) {

String input = "AliveisAwesome";

//create Method and pass input string as parameter

String reversed = reverseString(input);

System.out.println("The reversed string is: " + reversed);

}

//Method take string parameter and check string is empty or not

**public** **static** String **reverseString**(String input)

{

**if** (input.isEmpty()){

**return** input;

}

//Calling Function Recursively

**return** **reverseString**(input.substring(**1**)) + input.charAt(**0**);

}

}

**sol4)**

**import** **java.io.\***;

**import** **java.util.\***;

**public** **class** **reverseString** {

**public** **static** **void** **main**(String[] args) {

String input = "Be in present";

**byte** [] strAsByteArray = input.getBytes();

**byte** [] result = **new** **byte** [strAsByteArray.length];

**for**(**int** i = **0**; i<strAsByteArray.length; i++){

result[i] = strAsByteArray[strAsByteArray.length-i-**1**];

}

System.out.println( **new** String(result));

}}

**sol5)**

**import** **java.io.\***;

**import** **java.util.\***;

**public** **class** **reverseString** {

**public** **static** **void** **main**(String[] args) {

String input="";

System.out.println("Enter the input string");

**try**

{

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.in));

input = br.readLine();

**char**[] try1= input.toCharArray();

**for** (**int** i=try1.length-**1**;i>=**0**;i--)

System.out.print(try1[i]);

}

**catch** (IOException e) {

e.printStackTrace();

}

}}

**sol6)**

**import** **java.io.\***;

**import** **java.util.\***;

**public** **class** **reverseString** {

**public** **static** **void** **main**(String[] args) {

String input = "Be in present";

**char**[] hello=input.toCharArray();

List<Character> trial1= **new** LinkedList<>();

**for**(**char** *c:* hello)

trial1.add(c);

Collections.reverse(trial1);

ListIterator li = trial1.listIterator();

**while**(li.hasNext())

{System.out.print(li.next());}

}}

**Q6 How to convert String to char Array?**  
You can convert String to char Array using *toCharArray()* method.  
 **Q7 How many different ways you can create a String object?**  
You can create a String object using two ways. First is using *new* operator and second is using string *literal*. Objects created using *new* operator are stored in the heap memory while string *literals* are stored in the string constant pool.  
  
String str = "javahungry";  // String literal  
String str = new String("javahungry"); // using new operator  
 **Q8 Are String thread-safe in java?**  
As we know String objects are immutable. It means they are thread-safe also.  
 **Q9 Which class is final among String, StringBuilder and StringBuffer?**  
All are final classes.  
 **Q10 Is String primitive type or object (derived) type in java?**  
String is object(derived) type in java.  
 **Q11 Can we use String in switch statement?**  
Yes, you can use String in switch statement in java 7. Prior to java 7 , you had to use if-else statements to achieve the task.

**Q13 What is the difference between String and StringBuffer in java?**  
This is one of the most asked question in the java developer interview.  
String is immutable in java. Once created its value can not be changed. StringBuffer is mutable.

**Q14 What is the difference between StringBuilder and StringBuffer in java?**  
Below are the main differences between StringBuilder and StringBuffer in java.  
1. StringBuilder is not thread-safe. StringBuffer is thread-safe.  
2. StringBuilder is not synchronized and StringBuffer is synchronized.  
3. StringBuilder is faster while StringBuffer is slower as it is thread-safe.  
 **Q15  Explain the difference between below statements:**

String str = **new** String("abc");

String str = "abc";

In the first statement, String str = new String("abc");  
JVM will create one object in the heap memory. Another object in the String constant pool, if the object is not present. Otherwise,if present in the String constant pool ,it will return the reference to it.  
In the second statement, String str = "abc";  
JVM checks for the string "abc" in the String constant pool. If the string is not present in the constant pool then it will create a new String object and stores it in pool.  
If the string "abc" is found in string constant pool , then it simply creates a reference to it but does not create a new object.   
 **Q16 How many objects will be created for the following code:**

String str1 = "abc";

String str2 = **new** String("abc");

Two objects are created. Object created using *new* operator is stored in the heap memory (str2).  
Object created using String literal str1 is stored in the string constant pool.  
 **Q17 How many objects will be created for the following code:**

String str1 = "abc";

String str2 = "abc";

Only one object is created. String str1 will create a new object in String constant pool, while String str2 will create a reference to the String str1.  
 **Q18 How many objects will be created for the following code:**

String str1 = **new** String("abc");

String str2 = **new** String("abc");

Three objects are created. For the first statement(str1) two objects are created one in String constant pool and one in heap memory.  
But for the second statement(str2), compulsory 1 new object is created in heap memory but no new object is created in string constant pool as it is already present.  
  
Hence , a total of 2+1 = 3 objects are created.

**Q19 What is String intern() method?**

When the intern method is invoked, if the String constant pool already contains a string equal to the String object as determined by the equals(Object) method, then the string from the pool is returned.  
Otherwise the String object is added to the pool and a reference to the String object is returned.  
  
The task of intern() method is to put String (which is passed to the intern method) into string constant pool. **Q20 What are mutable and immutable objects in java?**  
Mutable objects value can be changed. StringBuilder and StringBuffer are the examples of the mutable objects.  
  
Immutable objects value can not be changed once created. String is an immutable class in java.  
 **Q21 Why is String immutable in java ?**  
There are various reasons to make String immutable in java.  
**1. Security :**String is made immutable to help increase the Security. Sensitive data like username,password can be stored as the Strings can't be modified once created.  
  
**2. Class loading :**String objects are used for Class loading. It is possible that wrong class has been loaded in the JVM, if the String is mutable i.e modifiable.  
  
**3. Thread Safe :**Immutable Strings are thread-safe. Synchronization is not required when we use them in the multithreading environment.  
  
**Q22 How will you create an immutable class in java?**  
You can create immutable class in java by implementing below points:  
  
1. Make the class final so it can not be extended(inherited)  
2. Make all fields private so one can not access them from outside the class.  
3. Do not provide setter methods for the variables.  
4. Declare all mutable fields as final so that it's value can be assigned only once.  
 **Q23 How will you create mutable String objects in java?**  
As we have discussed, by using StringBuffer and StringBuilder objects.  
 **Q24 What is the difference between Java String and C,C++ Strings ?**  
In C and Java both programming language treat String object as char Array.  
Java String is an object while C String is a NULL terminated character array. Java String object allows calling different methods like toUpperCase(), length(), substring().  
 **Q25 Why String is mostly used as a key in HashMap class?**  
String is mostly  used as a key in HashMap class because it implements equals() and hashCode() methods which is required for an Object to be used as key in HashMap.  
 **Q26 Is it possible to call String class methods using String literals?**  
Yes, It is possible to call String class methods using String literals. For example  
  
"javahungry".indexOf(u)  
"javahungry".charAt(0)  
"javahungry".compareTo("javahungry")  
 **Q27 How to Split String in java?**  
You can use split() method of java.lang.String class or StringTokenizer to  split a comma separated String. String split() method is easier to use and better because it expects a regular expression and returns an array of String which you can manipulate in the program code.

**Q28 Write a java program to find the first non repeated character in the String?**

**import** **java.util.HashMap**;

**import** **java.util.Scanner**;

**public** **class** **FirstNonRepeated** {

**public** **static** **void** **main**(String[] args)

{

// TODO Auto-generated method stub

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

Scanner in = **new** Scanner (System.in);

String s=in.nextLine();

**char** c=firstNonRepeatedCharacter(s);

System.out.println("The first non repeated character is : " + c);

}

**public** **static** Character **firstNonRepeatedCharacter**(String str)

{

HashMap<Character,Integer> characterhashtable=

**new** HashMap<Character ,Integer>();

**int** i,length ;

Character c ;

length= str.length(); // Scan string and build hash table

**for** (i=**0**;i < length;i++)

{

c=str.charAt(i);

**if**(characterhashtable.containsKey(c))

{

// increment count corresponding to c

characterhashtable.put( c , characterhashtable.get(c) +**1** );

}

**else**

{

characterhashtable.put( c , **1** ) ;

}

}

// Search characterhashtable in in order of string str

**for** (i =**0** ; i < length ; i++ )

{

c= str.charAt(i);

**if**( characterhashtable.get(c) == **1** )

**return** c;

}

**return** **null** ;

}

}

**Q29 How do you compare two Strings in Java?**  
Use equals() method to compare two Strings.Avoid using "==" operator. The main reason to use equals() method is that it always compare String values i.e content. == operator compares the reference in the memory.

String str1 = "abc";

String str2 = **new** String("abc");

System.out.println(str1 == str2); //false

System.out.println(str1.equals(str2)); //true

**Q30 Explain the difference between str1.equals("abc") and "abc".equals(str1), where str1 is any String object?**  
If str1 value is "abc" then both statements will give the result true. Main difference between the two statement arises when we pass str1 value as NULL. If the str1 is null then first statement will throw null pointer exception while second statement will return false.

**Q31 Find out if String has all Unique Characters?**

**sol 1)**

**import** **java.util.ArrayList**;

**import** **java.util.Collections**;

**import** **java.util.HashSet**;

**public** **class** **uniquechar** {

**public** **static** **void** **main** (String args[])

{

**boolean** result=**false**;

String inputstring="Alve i@wsom";

System.out.println(inputstring);

HashSet < Character> uniquecharset= **new** HashSet();

**for**(**int** i=**0**;i < inputstring.length();i++)

{

result=uniquecharset.add(inputstring.charAt(i));

**if** (result == **false**)

**break**;

}

System.out.println(result); }

}

**sol2)**

**import** **java.util.ArrayList**;

**import** **java.util.Collections**;

**import** **java.util.HashSet**;

**public** **class** **UniqueChar**2 {

**public** **static** **void** **main** (String args[])

{

**boolean** result=**false**;

String inputstring="Alive is awesome";

System.out.println("String method 2 answer "+ method2(inputstring));

}

**public** **static** **boolean** **method2**(String input)

{

**for**(**int** i=**0**; i < input.length();i++)

{

**char** charcterofinputstring=input.charAt(i);

**int** count=**0**;

**for**(**int** j=**i**; j < input.length();j++)

{

**if** (charcterofinputstring==input.charAt(j))

count++;

}

**if**(count > **1**)

**return** **false**;

}

**return** **true**;

}

}

**sol3)**

**import** **java.util.ArrayList**;

**import** **java.util.Collections**;

**import** **java.util.HashSet**;

**public** **class** **UniqueChar3** {

**public** **static** **void** **main** (String args[])

{

**boolean** result=**false**;

String inputstring="Alive is awesome";

System.out.println("String method 3 answer "+ method3(inputstring));

}

**public** **static** **boolean** **method3**(String input)

{

ArrayList ar= **new** ArrayList();

**for** (**int** i=**0**; i < input.length() ; i++ )

{

**int** j = input.indexOf(input.charAt(i));

ar.add(j);

}

Collections.sort(ar);

**for** (**int** i=**0**;i < (ar.size()-**1**);i++)

{

**if** (ar.get(i) == ar.get(i+**1**))

**return** **false**;

}

**return** **true**;

}

}

**sol4)**

**import** **java.util.ArrayList**;

**import** **java.util.Collections**;

**import** **java.util.HashSet**;

**public** **class** **UniqueChar4** {

**public** **static** **void** **main** (String args[])

{

**boolean** result=**false**;

String inputstring="Alive is awesome";

System.out.println("String method 4 answer "+ method4(inputstring));

}

**public** **static** **boolean** **method4**(String input)

{

**boolean** result=**false**;

**for** (**char** *ch:* input.toCharArray())

{

**if**(input.indexOf(ch)== input.lastIndexOf(ch))

result= **true**;

**else**

{

result=**false**;

**break**;

}

}

**return** result;

}

}

**sol5)**

public static boolean checkForUnique(String str){

boolean containsUnique = false;

for(char c : str.toCharArray()){

if(str.indexOf(c) == str.lastIndexOf(c)){

containsUnique = true;

} else {

containsUnique = false;

}

}

return containsUnique;

}

**Q32 How to Count number of words in the String?**

**public** **class** **StringDemo**

{

**static** **int** i,c=**0**,res;

**static** **int** **wordcount**(String s)

{

**char** ch[]= **new** **char**[s.length()]; //in string especially we have to mention the () after length

**for**(i=**0**;i<s.length();i++)

{

ch[i]= s.charAt(i);

**if**( ((i>**0**)&&(ch[i]!=' ')&&(ch[i-**1**]==' ')) || ((ch[**0**]!=' ')&&(i==**0**)) )

c++;

}

**return** c;

}

**public** **static** **void** **main** (String args[])

{

res=StringDemo.wordcount(" manchester united is also known as red devil ");

//string is always passed in double quotes

System.out.println("The number of words in the String are : "+res);

}

}

**Q33 How to remove all the white-spaces in the String?**

**public** **class** **SqueezeString**

{

**static** **int** i;

**static** **void** **squeeze**(String s)

{

**for**(i=**0**;i<s.length();i++)

{

**char** ch=s.charAt(i);

**if**(ch != ' ')

System.out.print(ch);

}

}

**public** **static** **void** **main** (String args[])

{

System.out.println("Original String is : ");

System.out.println(" manchester united is also known as red devil ");

SqueezeString.squeeze(" manchester united is also known as red devil ");

}

}

**Q34 Print all permutations of the String ?**

**import** **java.util.Scanner**;

**public** **class** **Permutation** {

**public** **static** **void** **main** (String args[])

{

System.out.println("Please enter the string whose permutations we need to show ");

Scanner in = **new** Scanner(System.in);

String original=in.nextLine();

System.out.println("");

System.out.println("");

System.out.println("");

System.out.println("Results are :");

System.out.println("");

System.out.println("");

permute(original);

}

**public** **static** **void** **permute**( String input)

{

**int** inputLength = input.length();

**boolean**[ ] used = **new** **boolean**[ inputLength ];

StringBuffer outputString = **new** StringBuffer();

**char**[ ] in = input.toCharArray( );

doPermute ( in, outputString, used, inputLength, **0** );

}

**public** **static** **void** **doPermute** ( **char**[ ] in, StringBuffer outputString,

**boolean**[ ] used, **int** inputLength, **int** level)

{

**if**( level == inputLength) {

System.out.println ( outputString.toString());

**return**;

}

**for**( **int** i = **0**; i < inputLength; ++i )

{

**if**( used[i] ) **continue**;

outputString.append( in[i] );

used[i] = **true**;

doPermute( in, outputString, used, inputLength, level + **1** );

used[i] = **false**;

outputString.setLength( outputString.length() - **1** );

}

}

}

**Q35 How to calculate total number of characters in the String?**

**public** **class** **CountCharacters**

{

**static** **int** i,c=**0**,res;

**static** **int** **charcount**(String s)

{

**for**(i=**0**,c=**0**;i<s.length();i++)

{

**char** ch=s.charAt(i);

**if**(ch!=' ')

c++;

}

**return** c;

}

**public** **static** **void** **main** (String args[])

{

res=CountCharacters.charcount(" manchester united is also known as red devil ");

//string is always passed in double quotes

System.out.println("The number of characters in the String are : "+res);

}

}

**Q36 How to calculate total number of vowels in String?**

**public** **class** **VowelsCount**

{

**static** **int** i,c,res;

**static** **int** **vowelcount**(String s)

{

**for**(i=**0**,c=**0**;i<s.length();i++)

{

**char** ch=s.charAt(i);

**if**((ch=='a')||(ch=='e')||(ch=='i')||(ch=='o')||(ch=='u'))

c++;

}

**return** c ;

}

**public** **static** **void** **main** (String args[])

{

System.out.println("Original String is : ");

System.out.println(" manchester united is also known as red devil ");

res=VowelsCount.vowelcount(" manchester united is also known as red devil ");

System.out.println("The number of vowels in the string is :" + res);

}

}

**Q37 String concatenation in java?**

**sol1)**

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// First String as an Object**

**String str = "Hello";**

**// Concatenating and storing above String object and constant string**

**String result = str + "World";**

**// Print the result**

**System.out.println(result);**

}}

sol2)

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// Two String objects**

**String str1 = "Hello";**

**String str2 = "World";**

**/\* Using concat() method, concat first string with second string and**

**storing into result string \*/**

**String result = str1.concat(str2);**

**// Print the result**

**System.out.println(result);**

}}

**sol3)**

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// Two String objects**

**String str1 = "Hello";**

**String str2 = "World";**

**/\* Using concat() method, concat first string with second string and**

**storing into result string \*/**

**String result = str1.concat(str2);**

**// Print the result**

**System.out.println(result);**

}}

**sol4)**

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// Two String objects**

**String str1 = "Hello";**

**String str2 = "World";**

**// StringBuffer object**

**StringBuffer sb = new StringBuffer();**

**// Appending str1 and str2**

**sb.append(str1).append(str2);**

**// Print the result**

**System.out.println(sb);**

}}

**sol5)**

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// Two String objects**

**String str1 = "Hello";**

**String str2 = "World";**

**// StringBuilder object**

**StringBuilder sb = new StringBuilder();**

**// Appending str1 and str2**

**sb.append(str1).append(str2);**

**// Print the result**

**System.out.println(sb);**

}}

**sol6)**

**import** **java.lang.\*;**

**public** **class** **concatenateString** {

**public** **static** **void** **main**(String[] args) {

**// Two String objects**

**String str1 = "Hello";**

**String str2 = "World";**

**// Concatenation and storing str1 and str2 object**

**StringBuffer result = String.join("", str1, str2);**

**// Print the result**

**System.out.println(result);**

}}

**Q38 Find all possible combinations of String?**

**public** **class** **Combinations** {

**private** StringBuilder output = **new** StringBuilder();

**private** **final** String inputstring;

**public** **Combinations**( **final** String str ){

inputstring = str;

System.out.println("The input string is : " + inputstring);

}

**public** **static** **void** **main** (String args[])

{

Combinations combobj= **new** Combinations("wxyz");

System.out.println("");

System.out.println("");

System.out.println("All possible combinations are : ");

System.out.println("");

System.out.println("");

combobj.combine();

}

**public** **void** **combine**() { combine( **0** ); }

**private** **void** **combine**(**int** start ){

**for**( **int** i = start; i < inputstring.length(); ++i ){

output.append( inputstring.charAt(i) );

System.out.println( output );

**if** ( i < inputstring.length() )

combine( i + **1**);

output.setLength( output.length() - **1** );

}

}

}

**Q39 Write a java program to check if the input string is palindrome?**

**import** **java.util.\***;

**import** **java.lang.String**;

**public** **class** **Palindrome**

{

**static** Scanner console = **new** Scanner(System.in);

**public** **Palindrome**()

{

}

**public** **static** **void** **main**(String[] args)

{//declare the variables

String str,another="y";

//start the loop

**while**(another.equalsIgnoreCase("y"))

{

//prompt the user

System.out.print("Enter a word to see if its a Palindrome; ");

str= console.next();

System.out.println();

//

//the answer

**if**(isPalindrome(str))

System.out.println( str + " is a palindrome");

**else**

System.out.println(str + " not a palindrome");

System.out.print("test another(y/n)? ");

another= console.next();

}

}

**public** **static** **boolean** **isPalindrome**(String str)

{

**return** **isPalindrome**(str,**0**,str.length()-**1**);

}

**public** **static** **boolean** **isPalindrome**(String str,**int** low, **int** high)

{ **if**(high <= low)

**return** **true**;

**else** **if** (str.charAt(low)!= str.charAt(high))

**return** **false**;

**else**

**return** **isPalindrome**(str,low+**1**,high-**1**);

}

}

**Q40 What is String Constant Pool?  Why java provided String Constant pool as we can store String in the heap memory?**  
String constant pool is the memory space allocated in the heap memory to store the objects which are  
created using String literals. String constant pool is unique, there are no two String o objects which has the same value(content).  
  
**Why String Constant Pool ?**  
  
String constant pool increases the reusability of the existing String objects.  
It also saves memory as no two objects with same content are created.

**Q41 There are lot of String concatenation and String modification operations in my code. Which class should I use among String,StringBuffer and StringBuilder? Given I also want thread-safe code?**  
 **This is scenario based question. You should give answer StringBuffer.**  
You can use String also but with every modification and concatenation operation, a new String is formed as String is immutable. It will lead to the memory allocation issues.  
StringBuilder can not be used as it is not synchronized, i.e thread-safe.  
  
So, the clear answer is StringBuffer.

**Q42 Why char Array is preferred over String in storing passwords?**  
  
One of the main reason to prefer char Array over String is security risk of stealing passwords. Since String are reusable in the constant pool , there are high chances that they remain in the memory for the long duration. Anyone who has access to the memory dump can find the password in clear text.  
That's why password should be encrypted.  
  
**Q43 What is Character encoding? Explain the difference between UTF-16 and UTF-8?**  
  
When you want to represent Character using bytes, character encoding is used.  
  
The UTF-16 uses 2 bytes or 16 bits to represent a character while UTF-8 uses 1 byte or 8 bits to represent a character.  
  
**Q44 How does substring() method works in java?**  
  
substring shares the same character array as String. It can lead to the memory leak if the original String is quite big and not necessary to retain in the memory. It is unintentionally retained by substring as substring is smaller in size.It results in the prevention of large array being garbage collected.

**Q45 Anagram program in java?**   
  
Write a java program to check whether two given strings are anagram. If two strings contain same set of characters but in different order then the two strings are called anagram.

**import** **java.util.\***;

**import** **java.io.\***;

**public** **class** **Anagram**

{

**public** **static** **void** **main** (String[] args) **throws** java.lang.Exception

{

**boolean** result = isAnagram("now","own");

System.out.println(result);

}

**public** **static** **boolean** **isAnagram**(String first, String second)

{

// remove all whitespaces and convert strings to lowercase

first = first.replaceAll("\\s", "").toLowerCase();

second = second.replaceAll("\\s", "").toLowerCase();

/\* check whether string lengths are equal or not,

if unequal then not anagram \*/

**if** (first.length() != second.length())

**return** **false**;

// convert string to char array

**char**[] firstArray = first.toCharArray();

**char**[] secondArray = second.toCharArray();

// sort both the arrays

Arrays.sort(firstArray);

Arrays.sort(secondArray);

// checking whether both strings are equal or not

**return** Arrays.equals(firstArray,secondArray);

}

}

**Q46 How to Count occurrences of each character in a String in java?**   
  
Write a java program to count occurrences of each character in String in java. If the String is   
**"Java Hungry"**then the answer should be

**{ =1, a=2, r=1, u=1, v=1, g=1, H=1, y=1, J=1, n=1}**

**sol1)**

**public** **class** **EachCharacterCountInString**

{

**public** **static** **void** **main**(String[] args)

{

characterCount("Alive is Awesome");

characterCount("Java Hungry");

characterCount("USA has 50 states");

}

**static** **void** **characterCount**(String inputString)

{

//Creating a HashMap, key :Character value : occurrences as Integer

HashMap<Character, Integer> eachCharCountMap = **new** HashMap<Character, Integer>();

//Converting inputString to char array

**char**[] charArray = inputString.toCharArray();

//traversal of each Character of charArray

**for** (**char** c : charArray)

{

**if**(eachCharCountMap.containsKey(c))

{

//If char is present in eachCharCountMap, increment count by 1

eachCharCountMap.put(c, eachCharCountMap.get(c)+**1**);

}

**else**

{

//If char is not present in eachCharCountMap,

//Putting this char to eachCharCountMap with 1 as it's initial value

eachCharCountMap.put(c, **1**);

}

}

//Showing the eachCharCountMap

System.out.println(eachCharCountMap);

}

}

**Output:**

{ =2, A=1, a=1, s=2, e=3, v=1, w=1, i=2, l=1, m=1, o=1}

{ =1, a=2, r=1, u=1, v=1, g=1, H=1, y=1, J=1, n=1}

{ =3, 0=1, A=1, a=2, S=1, s=3, t=2, U=1, 5=1, e=1, h=1}

**sol2)**

**public** **class** **AlphabetFrequencyString** {

**static** **int** i,j,k,c=**0**,w;

**static** **char** m; //we can only define static for variables and fns not for arrays

**public** **static** **void** **main**(String[] args) {

System.out.println("Input string is : ");

System.out.println("Alive is awesome");

System.out.println("");

System.out.println("");

System.out.println("Output :");

frequencycount("Alive is awesome");

}

**static** **void** **frequencycount**(String s)

{

**char**[] z=**new** **char**[s.length()];

**for**(w=**0**;w<s.length();w++)

z[w]=s.charAt(w);

**for**(i=**0**;i<w;i++)

{

**char** ch=z[i];

**for**(j=i+**1**;j<w;j++)

{

**if**(z[j]==ch)

{

**for**(k=j;k<(w-**1**);k++)

z[k]=z[k+**1**];

w--;

j=i;

}

}

}

**int**[] t=**new** **int**[w];

**for**(i=**0**;i<w;i++)

{

**for**(j=**0**,c=**0**;j<s.length();j++)

{

**if**(z[i]==s.charAt(j))

c++;

}

t[i]=c ;

System.out.print(z[i]+"="+c+",");

}

}

}

**Q47 Convert Lowercase to Uppercase in java and Uppercase to Lowercase without using built in method ?**

**public** **class** **ChangeCase**

{

**static** **int** i;

**static** **void** **changecase**(String s)

{

**for**(i=**0**;i<s.length();i++)

{

**int** ch=s.charAt(i);

**if**(ch>**64**&&ch<**91**)

{

ch=ch+**32**;

System.out.print( (**char**) ch);

}

**else** **if**(ch>**96**&&ch<**123**)

{

ch=ch-**32**;

System.out.print( (**char**) ch);

}

**if**(ch==**32**)

System.out.print(" ");

}

}

**public** **static** **void** **main** (String args[])

{

System.out.println("Original String is : ");

System.out.println("Alive is awesome ");

ChangeCase.changecase("Alive is awesome ");

}

}

**Q48** **How to remove specific characters in the String?**

**import** **java.util.Scanner**;

**public** **class** **RemoveSpecificCharacter** {

**public** **static** **void** **main**(String[] args)

{

String originalstring="Alive is Awesome";

System.out.println("Original string is >> "+ originalstring);

System.out.println("");

System.out.println("Please enter unwanted characters as String");

System.out.println("");

System.out.println("");

Scanner in =**new** Scanner(System.in);

String removecharacterstring=in.nextLine();

String output=removeSpecificChars(originalstring, removecharacterstring);

System.out.println("");

System.out.println("");

System.out.print("Output is >> " );

System.out.println(output);

}

**public** **static** String **removeSpecificChars**(String originalstring ,String removecharacterstring)

{

**char**[] orgchararray=originalstring.toCharArray();

**char**[] removechararray=removecharacterstring.toCharArray();

**int** start,end=**0**;

//tempBoolean automatically initialized to false ,size 128 assumes ASCII

**boolean**[] tempBoolean = **new** **boolean**[**128**];

//Set flags for the character to be removed

**for**(start=**0**;start < removechararray.length;++start)

{

tempBoolean[removechararray[start]]=**true**;

}

//loop through all characters ,copying only if they are flagged to false

**for**(start=**0**;start < orgchararray.length;++start)

{

**if**(!tempBoolean[orgchararray[start]])

{

orgchararray[end++]=orgchararray[start];

}

}

**return** **new** **String**(orgchararray,**0**,end);

}

}

**Q49 How to Convert Signed Integer to String?**

**sol1)**

public **class** **intToString** {

public static void main(String args [])

{

String result= intToStr(**23**);

System.out.println("Output is : "+ result);

}

public static final int MAX\_DIGITS = **10**;

public static String intToStr( int num ){

int i = **0**;

boolean isNegative = false;

/\* Buffer big enough **for** largest int **and** - sign \*/

char[] temp = new char[ MAX\_DIGITS + **1** ];

/\* Check to see **if** the number **is** negative \*/

**if**( num < **0** ){

num = -num;

isNegative = true;

}

/\* Fill buffer **with** digit characters **in** reverse order \*/

do {

temp[i++] = (char)((num % **10**) + '0');

num /= **10**;

} **while**( num != **0** );

StringBuilder b = new StringBuilder();

**if**( isNegative )

b.append( '-' );

**while**( i > **0** ){

b.append( temp[--i] );

}

**return** b.toString();

}

}

***Method 2 :*Convert int to string using valueOf() method**

int number=**537**;

String result = String.valueOf(number);

System.out.println(result);

***Method 3 :* Convert int to string using toString() method**

int number=**537**;

String result = Integer.toString(number);

System.out.println(result);

***Method 4 :*Append  empty string ""  to the integer using + operator**

int number=**537**;

System.out.println(number + "" );

**Big Question !! Which is most efficient method to convert integer to string?and Why ?**  
  
We are discussing comparison  between Method 2 , Method 3 and Method 4 , since the Method 1 is our own algorithm , we are not comparing it with the rest.  
  
So the Order is :  
  
**Method 3 (recommended)> Method  2 > Method  4 (least preferred)**  
  
Why ?  
  
Method 4 is the least preferred because  
  
It uses empty string "" with + operator which internally performs :

StringBuilder sbobject= new StringBuilder();

sbobject.append();

sbobject.append(number);

**return** sbobject.toString();

So there is so much overhead before it reaches toString() method.  
  
  
Method 2 ,[valueOf() method internal implementation](http://grepcode.com/file/repository.grepcode.com/java/root/jdk/openjdk/7-b147/java/lang/String.java?av=f) is like this  :

public static String valueOf(Object obj) {

**return** (obj == null) ? "null" : obj.toString();

}

Above, we can see that  valueOf() internally calling toString() method , to convert integer to string.  
  
Hence  Method 3 is the most efficient where we are directly calling toString() method to achieve our goal .  
  
Please mention in the comments if you know any other way of converting integer to string.

**Q50 Find the length of the String without using length() method?**

**public** **class** **StringLength**

{

**static** **int** i,c,res;

**static** **int** **length**(String s)

{

**try**

{

**for**(i=**0**,c=**0**;**0**<=i;i++,c++)

s.charAt(i);

}

**catch**(Exception e)

//Array index out of bounds and array index out of range are different exceptions

{

System.out.print("length is ");

// we can not put return statement in catch

}

**return** c;

}

**public** **static** **void** **main** (String args[])

{

System.out.println("Original String is : ");

System.out.println("Alive is awesome ");

res=StringLength.length("Alive is awesome ");

System.out.println( res);

}

}