|  |  |
| --- | --- |
| Week3 | Java Fundamentals-III – arrays, strings, access modifiers, packages,  regular expressions, Java new features |

An **array** is a collection of Homogenious data types (similar data type) which has contiguous memory location.

Arrays are declared with **[]** after the data type

**index** beginning with 0

Size of array is found by **length** property.

Java arrays can store only a fixed set of elements

|  |  |
| --- | --- |
| Array Declarartion | Array Initialization |
| dataType[] arr; (or)  dataType []arr; (or)  dataType arr[]; | 1. arrayRefVar=new dataType[size]; |

Styles of array declarations in java

1. int a[]=new int[5];
2. int []b=new int[6];
3. int c[];

c=new int[10];

1. int d[]=new int[]{10,20,30};

**NOTE**:- java.lang.**ArrayIndexOutOfBoundsException** will be raised when tried to access ayyar elements out of index.

Array elements can be printed by using

|  |  |
| --- | --- |
| A normal for loop | for each loop |
| 1. int a[]={33,3,4,5}; 2. for(int i=0;i<a.length;i++) 3. System.out.println(a[i]); 4. } | 1. int arr[]={33,3,4,5}; 2. for(int i:arr) 3. System.out.println(i); 4. } |

**NOTE**:- Name of the array is equivalent to the address of zeroth elementof the array.

So to pass an array to a method , we can pass the name of the array.

**INTERVIEW QUESTIONS**

**Can you pass the negative number in Array size?**  
No, you can not pass the negative number as Array size. If you pass a negative number in Array size then you will not get the compiler error. Instead, you will get the NegativeArraySizeException at run time.  
  
**Can you declare an Array without Array size?**  
No, you can not declare Array without Array size. You will get compile time error.  
  
**Where does Array stored in JVM memory ?**  
Array is an object in java. So, Array is stored in heap memory in JVM.

**Which is legal int[] arr or int arr[] ?**  
Both are legal statements. It is preferred to write int[] arr instead of int arr[].

**WAP to find largest value in the array.**

Suppose given array is 173,29,391,41 then the largest value is 391

**WAP to find intersection of two arrays in java?**

Suppose given two arrays array1= {1,4,7, 9, 2} arrray2 = {1,7,3,4,5}

the answer should be {1,4,7}

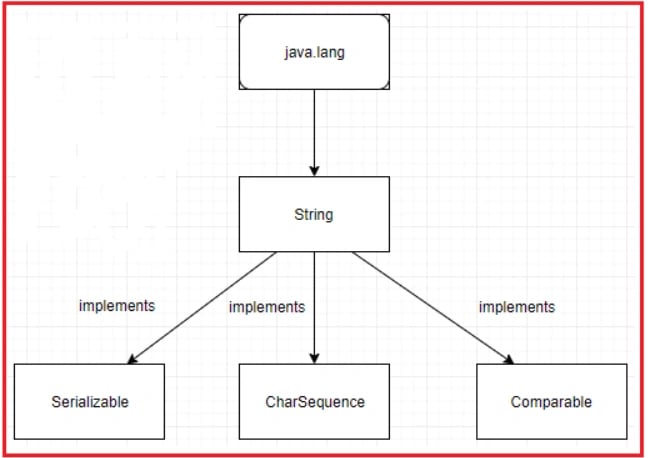
**WAP to merge two arrays in java?**

If array1 is {1,2,3} and array2 is {4,5,6} then merged array will be {1,2,3,4,5,6}

**Strings,** a sequence or series of characters, enclosed in a pair of double quotes (“”).

In java, String is an immutable, once created, a [String cannot be changed or modified](https://javahungry.blogspot.com/2015/07/why-string-is-final-or-immutable-in-java-with-example.html) and all such operations on a String lead to creation of a new String object.

The java.lang.String class implements *Serializable*,  *Comparable* and *CharSequence* [interfaces](https://www.javatpoint.com/interface-in-java)



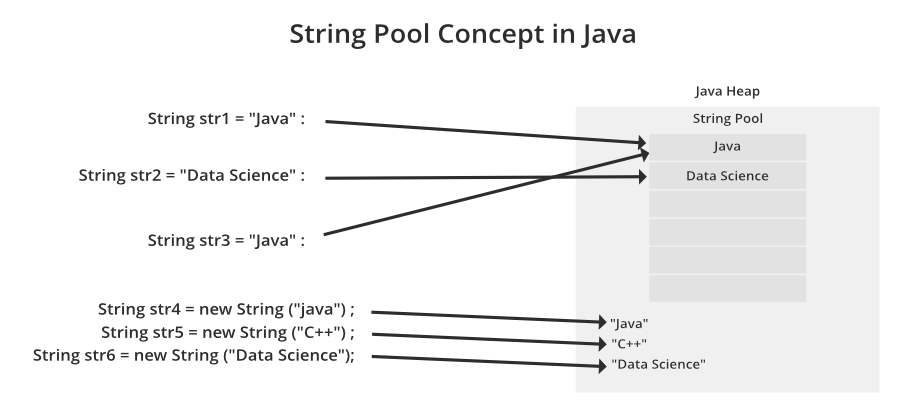
# Difference Between String , StringBuilder And StringBuffer Classes

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***String*** | ***StringBuffer*** | ***StringBuilder*** |
| **Storage** | Constant String Pool | Heap | Heap |
| **Modifiable** | No (immutable) | Yes( mutable ) | Yes( mutable ) |
| **Thread Safe** | Yes | Yes | No |
| **Performance** | Slow | Slow but faster than String | Fast |
| **Example** | **String demo = "hello" ;**  **demo="Bye" ;**  // "hello" string still exists in string constant pool and its value is not [overrided](https://javahungry.blogspot.com/2018/11/method-overriding-in-java-with-examples.html) but we lost reference to the  "hello"string | **StringBuffer demo1 = new StringBuffer("Hello");**  **demo1 = new StringBuffer("Bye");**  // Above statement is right as it modifies the value which is allowed in the StringBuffer | **StringBuilder demo2= new StringBuilder("Hello");**  **demo2=new StringBuilder("Bye");**  / / Above statement is right as it modifies the value which is allowed in the StringBuilder |

### string created without using new keyword will be stored in **Java String Pool**

### string created using new keyword will be stored in **Java Heap**

Every immutable object in Java is **thread-safe**

****

**GUESS THE O/P**

Sample Code Snippet #1:

String str1 = new String("java");

String str2 = new String("java");

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

String str1 = "java";

String str2 = "java";

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

**Java String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), substring() etc.

public int **compareTo**(String str)---for comparing two strings

returns 0 if equal

<0 if str1 is less than str2

>0 if str2 is less than str1

**WAP to remove special char from the given string**

**Most commonly used methods in String class**

****

**WAP to remove duplicate chars from the given string**

* Using jav 8 + lamda expressions
* indexOf()
* character array
* set interface method

**WAP to remove special char from the given string**

replaceAll() & regular expression, replaceAll([^\\s ] ,””)

**WAP to sort the string without using sort() method**

**WAP to reverse a string without using reverse() method.**

**WAP to find the first non repeated character in the String?**  
**Input:** programming

**Output:**First non-repeating character is: p

**Java Regular expressions**

The **Java Regex** or Regular Expression is an API to define a pattern for searching or manipulating strings.

Java Regex API provides 1 interface and 3 classes in **java.util.regex** package

1. MatchResult interface
2. Matcher class
3. Pattern class
4. PatternSyntaxException class

import java.util.regex.\*;

public class RegexExample1{

public static void main(String args[]){

//1st way

Pattern p = Pattern.compile(".s");//. represents single character

Matcher m = p.matcher("as");

boolean b = m.matches();

//2nd way

boolean b2=Pattern.compile(".s").matcher("as").matches();

//3rd way

boolean b3 = Pattern.matches(".s", "as");

System.out.println(b+" "+b2+" "+b3);

}

}

## **Regex Character classes**

| **Character class** | **Description** |
| --- | --- |
| **[xyz]** | x,y or z |
| **[^xyz]** | Any characters other than x,y or z |
| **[a-zA-Z]** | characters from range a to z or A to Z. |
| **[a-f[m-t]]** | Union of a to f and m to t. |
| **[a-z && p-y]** | All the range of elements intersection between two ranges |
| **[a-z && [^bc]]** | a to z union with except b and c |
| **[a-z && [^m-p]]** | a to z union with except range m to p |

## **Regex Metacharacters**

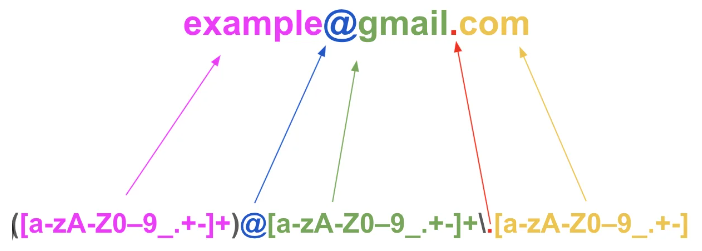
| **Regex** | **Description** |
| --- | --- |
| **X?** | X appears once or not |
| **X+** | X appears once or more than once |
| **X\*** | X appears zero or not once |
| **X{n}** | X appears n times |
| **X{n,}** | X appears n times or more than n |
| **X{n,m}** | X appears greater than equal to n times and less than m times. |

## **Java Regex Finder Example**

| **Regex** | **Description** |
| --- | --- |
| **.** | Any character |
| **\d** | Any digits, [0-9] |
| **\D** | Any non-digit, [^0-9] |
| **\s** | Whitespace character, [\t\n\x0B\f\r] |
| **\S** | Non-whitespace character, [^\s] |
| **\w** | Word character, [a-zA-Z\_0-9] |
| **\W** | Non-word character, [^\w] |
| **\b** | Word boundary |
| **\B** | Non -Word boundary |

**What is difference between matches() and find() in Java Regex?**matches() returns true only if the whole string matches the specified pattern while find() returns trues even if a substring matches the pattern.

**WAP to validate email id using regular expressions**

****

**Java Lamda Expressions**

An interface which has only one abstract method is called functional interface.

Lambda expression provides implementation of functional interface.

## Java Lambda Expression Syntax:-- **(argument-list) -> {body}**

Java lambda expression is consisted of three components.

**1) Argument-list:** It can be empty or non-empty as well.

**2) Arrow-token:** It is used to link arguments-list and body of expression.

**3) Body:** It contains expressions and statements for lambda expression.

**Java Packages**

A **java package** is a group of similar types of classes, interfaces and sub-packages.

1) Java package is used to categorize the classes and interfaces so that they can be easily maintained.

2) Java package provides access protection.

3) Java package removes naming collision.

The **package keyword** is used to create a package in java.

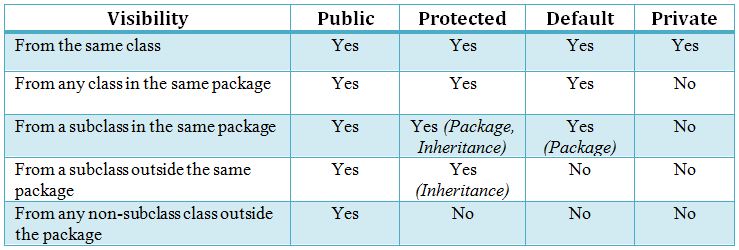
To compile javapackages following command to be used

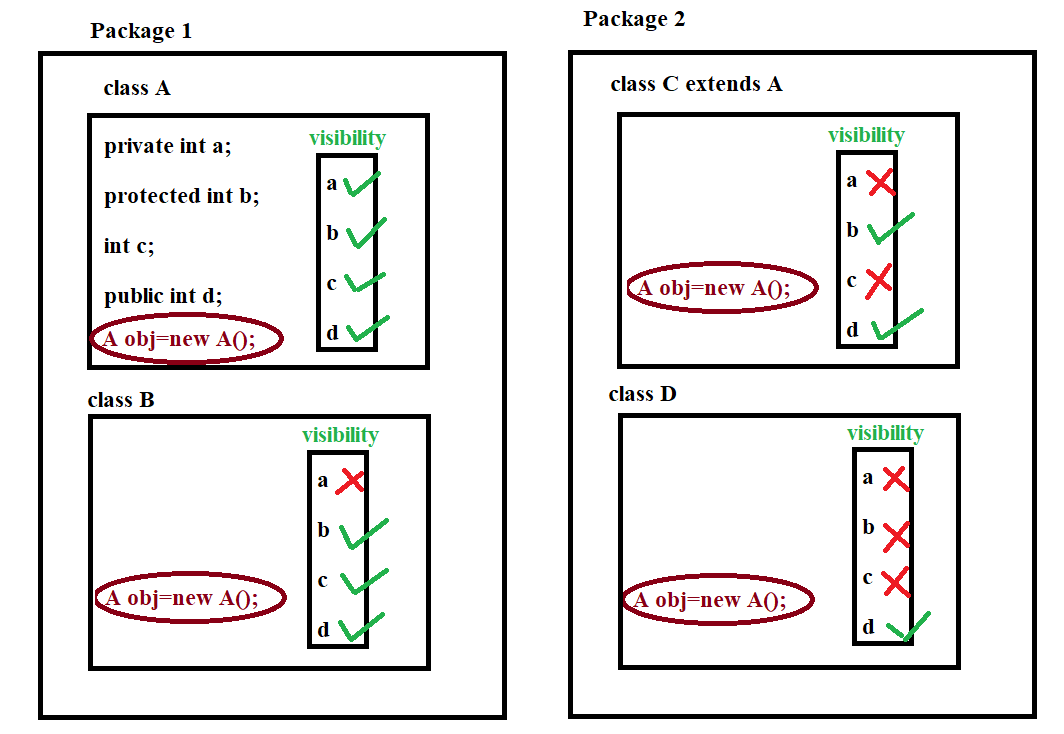
**javac -d directory javafilename**

**Note:** If you import a package, subpackages will not be imported.

**Java Access Modifiers**

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class.



****