# **Arrays:**

An array is a collection of similar data elements.

Array index always begins with 0 and ends with size-1.

In java, Array itself an object.

Array reference is called as an object reference or reference variable.

Example:

Int arrlist[] = new int[5];

Or

Int[] arrlist = new int[5];

Or

Int []arrlist = new int[5];

Diagram

Description automatically generated with low confidence

|  |  |
| --- | --- |
| **Datatype** | **default value** |
| **byte** | **0** |
| **short** | **0** |
| **int** | **0** |
| **long** | **0** |
| **float** | **0.0f** |
| **double** | **0** |
| **char** | **Black Space (Ascii value 0)** |
| **boolean** | **FALSE** |

NOTE: LOCAL VARIABLES MUST BE INITIALIZED BEFORE ACCESS, OTHERWISE COMPILE TIME ERROR OCCURS.

Graphical user interface, text, application

Description automatically generated

**Array Assignment Syntax**:

Arrayreference[index] = Literal.

Examples:

arrlist[0] = 100;

arrlist[1] = 200;

**Initialization:**

Int Arrayreference[] = {Literal1,Literal2,….};

Int x[] = {100,200,300,…...};

Application, table

Description automatically generated with medium confidence

Graphical user interface, text, application, Word

Description automatically generated

//OUTPUT

14

53

59

14

53

# Strings

## String class

Strings in java are objects that are backed up by a character array. Like arrays, strings are immutable (cannot grow) as well. Whenever a change to a string is made an entirely new string is created.

Whenever a string object is created as literal, the object is created in a string constant pool.

For example:

String str = “Gowtham”;

The string can also be declared using the ‘new’ operator, i.e. dynamically allocated. These are assigned a new memory location in a heap. This string will not be added to the constant pool.

String str = new String(“Gowtham”);

If we want to store this string in a constant pool we need to intern it.

String str1 = str.intern();

It is preferred to use string literal as it allows JVM to optimize memory locations.

Refer code snippet: StringJoinerExample1

|  |
| --- |
| **package** arraysAndstrings;  **public** **class** StringJoinerExample1  {  **public** **static** **void** main(String[] args)  {  String str = String.*join*(" < ", "one","two","three");  String str1 = String.*join*(" ", "My","Name","is","Gowtham");  String str2 = String.*join*(" -> ", "wakeup","eat","play","sleep");    System.***out***.println(str);  System.***out***.println(str1);  System.***out***.println(str2);    //String length  System.***out***.println(str2.length());    //CharAt(index)  **char** i = str2.charAt(0);  System.***out***.println(i);    //substring(int beginindex)  String sub = str2.substring(10);  System.***out***.println(sub);    //substring(int beginindex, int endindex)  String sub1 = str2.substring(10,15);  System.***out***.println(sub1);    //Concat(String str)  String str4 = "Gowtham";  String str5 = "Anusha";  String str9 = str5.concat(str4);  System.***out***.println(str9);    //int index(String s)  **int** x = str9.indexOf("t");  System.***out***.println(x);    //int index(String s,int i)  **int** y = str5.indexOf("t",1);  System.***out***.println(y);    //int lastIndexOf(String s,int i)  **int** z = str5.indexOf("G");  System.***out***.println(z);    //boolean equals( Object otherObj): Compares this string to the specified object  Boolean out = "Gowtham".contentEquals("Gowtham"); //=> True  System.***out***.println(out);  out = "Gowtham".contentEquals("gowtham"); //false  System.***out***.println(out);    // boolean equalsIgnoreCase (String anotherString): Compares string to another string, ignoring case considerations.    Boolean out1 = "Gowtham".equalsIgnoreCase("Gowtham"); //=> True  System.***out***.println(out1);  out1 = "Gowtham".equalsIgnoreCase("gowtham"); //true  System.***out***.println(out1);    // int compareTo( String anotherString): Compares two string lexicographically.  String s1 = "Gowtham";  String s2 = "gowtham";  **int** out2 = s1.compareTo(s2);  System.***out***.println(out2);    /\* This returns difference s1-s2. If :  out < 0 // s1 comes before s2  out = 0 // s1 and s2 are equal.  out > 0 // s1 comes after s2.  \*/    //int compareToIgnoreCase( String anotherString): Compares two string lexicographically, ignoring case considerations.  String s3 = "Gowtham";  String s4 = "gowtham";  **int** out3 = s3.compareToIgnoreCase(str5);  System.***out***.println(out3);  /\* This returns difference s1-s2. If :  out < 0 // s1 comes before s2  out = 0 // s1 and s2 are equal.  out > 0 // s1 comes after s2.  \*/    //String toLowerCase(): Converts all the characters in the String to lower case  String str6 = "Hello World";  System.***out***.println(str6.toLowerCase());    //String toUpperCase(): Converts all the characters in the String to upper case.  String str7 = "Hello World";  System.***out***.println(str7.toUpperCase());    //String trim(): Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.  String str8 = " Gowtham Anusha ";  System.***out***.println(str8.trim());    //String replace (char oldChar, char newChar): Returns new string by replacing all occurrences of oldChar with newChar.  String s6 = "feeksforfeeks";  String s7 = s6.replace('f','g'); // returns “geeksgorgeeks”  System.***out***.println(s7);    }  } |

## StringBuffer class

StringBuffer is a peer class of String that provides much of the functionality of strings. The string represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

**Constructors of StringBuffer class**

1. StringBuffer(): It reserves room for 16 characters without reallocation

StringBuffer s = new StringBuffer();

2. StringBuffer( int size): It accepts an integer argument that explicitly sets the size of the buffer.

StringBuffer s = new StringBuffer(20);

3. StringBuffer(String str): It accepts a string argument that sets the initial contents of the StringBuffer object and reserves room for 16 more characters without reallocation.

StringBuffer s = new StringBuffer("GeeksforGeeks");

**Methods of StringBuffer class**

Methods Action Performed

append() Used to add text at the end of the existing text.

length() The length of a StringBuffer can be found by the length( ) method

capacity() the total allocated capacity can be found by the capacity( ) method

charAt()

delete() Deletes a sequence of characters from the invoking object

deleteCharAt() Deletes the character at the index specified by loc

ensureCapacity() Ensures capacity is at least equals to the given minimum.

insert() Inserts text at the specified index position

length() Returns length of the string

reverse() Reverse the characters within a StringBuffer object

replace() Replace one set of characters with another set inside a StringBuffer object

## StringBuilder class

The StringBuilder in Java represents a mutable sequence of characters. Since the String Class in Java creates an immutable sequence of characters, the StringBuilder class provides an alternative to String Class, as it creates a mutable sequence of characters. The function of StringBuilder is very much similar to the StringBuffer class, as both of them provide an alternative to String Class by making a mutable sequence of characters. However the StringBuilder class differs from the StringBuffer class on the basis of synchronization. The StringBuilder class provides no guarantee of synchronization whereas the StringBuffer class does. Therefore this class is designed for use as a drop-in replacement for StringBuffer in places where the StringBuffer was being used by a single thread (as is generally the case). Where possible, it is recommended that this class be used in preference to StringBuffer as it will be faster under most implementations. Instances of StringBuilder are not safe for use by multiple threads. If such synchronization is required then it is recommended that StringBuffer be used.

**Constructors in Java StringBuilder:**

StringBuilder(): Constructs a string builder with no characters in it and an initial capacity of 16 characters.

StringBuilder(int capacity): Constructs a string builder with no characters in it and an initial capacity specified by the capacity argument.

StringBuilder(CharSequence seq): Constructs a string builder that contains the same characters as the specified CharSequence.

StringBuilder(String str): Constructs a string builder initialized to the contents of the specified string.

## Difference between String, StringBuffer and StringBuilder

Text, whiteboard

Description automatically generated

Refer Code Snippet: StringBasicExample1

|  |
| --- |
| **package** arraysAndstrings;  **public** **class** StringBasicExample1  {  **public** **static** **void** main(String[] args)  {  String str = "Apple";  StringBuffer sb = **new** StringBuffer("Apple");  StringBuilder sbr = **new** StringBuilder("Apple");    System.***out***.println(str+ " - "+ sb + " - "+sbr);  }  } |

## String Tokenizer

This is used to break the string into tokens.

Refer code snippet: StringTokenizerExample1

|  |
| --- |
| **package** arraysAndstrings;  **import** java.util.StringTokenizer;  **public** **class** StringTokenizerExample1  {  **public** **static** **void** main(String[] args)  {  StringTokenizer str = **new** StringTokenizer("Gowtham Challa Lives in Atlanta.");  **while** (str.hasMoreTokens())  {  System.***out***.println(str.nextToken());  }  }  } |

## String Joiner

The java.lang.string.join() method concatenates the given elements with the delimiter and returns the concatenated string.Note that if an element is null, then null is added.

Refer code snippet: StringJoinerExample1

|  |
| --- |
| **package** arraysAndstrings;  **public** **class** StringJoinerExample1  {  **public** **static** **void** main(String[] args)  {  String str = String.*join*(" < ", "one","two","three");  String str1 = String.*join*(" ", "My","Name","is","Gowtham");  String str2 = String.*join*(" -> ", "wakeup","eat","play","sleep");    System.***out***.println(str);  System.***out***.println(str1);  System.***out***.println(str2);  }  } |