**INDEX**

*Syntax are here, Codes are here in Eclipse.*

|  |  |  |
| --- | --- | --- |
| Sr/No | Title | Page |
| 1 | Java Collections | 1 |
| 2 | Java In-Built Functions | **5** |
| 3 | Conversion | **7** |
| 4 | Bit Magic | **8** |
| 5 | My understood ALGO, DS and OTHER Coding tricks. | **9** |
| 6 |  |  |
|  |  |  |

**Java Collections**

*There are LOT MANY inbuild functions for each collections. Check internet later if need extra stuff. Here I have mentioned functions needed for now.*

STACK: (Functions are same on both sides, use based on your need)

|  |  |
| --- | --- |
| **Stack using Stack class** | **Stack using ArrayDeque** |
| Stack<> \_\_\_=new Stack<>(); | Deque< > \_\_\_\_\_=new ArrayDeque<>(); |
| .push(); // push new element in stack | .push(); // push new element in stack |
| .Pop(); // pop element from stack | .Pop(); // pop element from stack |
| .peek(); // peek top element | .peek(); // peek top element |

Queue: (Functions are same on both sides, use based on your need)

|  |  |
| --- | --- |
| **Queue using LinkedList** | **Queue using ArrayDeque** |
| Queue<String> foodLine=new LinkedList<>(); | Deque<String> guns=new ArrayDeque<>(); |
| .offer(); // insert element in queue | .offer(); // insert element in queue |
| .peek(); // check element at front | .peek(); // check element at front |
| .poll(); // remove element at front | .poll(); // remove element at front |
|  |  |

ArrayList / LinkedList : (Functions are same on both LL and AL)

|  |  |
| --- | --- |
|  | **List< > \_\_\_\_=new ArrayList<>();**  **List< > \_\_\_\_=new LinkedList<>();** |
|  | ***List<> \_\_\_\_=new ArrayList<>();*** *or* ***LinkedList<>();*** *or* ***Vector<>();*** *or* ***Stack<>();* // we can also make List using any of these. But I prefer not using stack and vector. As we can just use functions of LL and AL on List if we made List using LL or AL.** |
|  | 1. When we define the List as follows: (USE THIS MOST OF TIME)  **List ­­­­­­\_\_\_\_ = new ArrayList();** It means we can only call the methods and reference members from the List interface.  2. If we define the ArrayList as follows:  **ArrayList ­­\_\_\_\_ = new ArrayList();** It means we can invoke available methods in ArrayList and use its members in addition to the List.  *It is better to use the List Interface if you want to take advantage of the polymorphism. In the future, if we are required to implement the interface, we will not need to change the program.* |
|  | .add(value); // add()- to add element. New element is added at the end. |
|  | .add(index, value); // We can also add at specific position, using index value. (We insert there, and other elements shift) |
|  | .addAll(); // We can concat two ArrayList’s |
|  | .set(index, value); // This updates the value at specified position. Ie. Overrides. |
|  | .size(); // Size gives no. of elements & not size/length of array. |
|  | .get(); // we can get a element at a specific index. |
|  | .remove(index); // We can remove/delete a element using its index. |
|  | .remove(type.valueOf(value)); // We can also remove element based on its 'type' and 'value' (instead of using index like above.) here type is Generic datatype. |
|  | .clear(); // It completely empty out's the Array list. |
|  | .contains("Garuda"); // It checks if given value is present of not. returns true/false. |
|  | .isEmpty(); |
| 12. | To return a List: **return(numbers);**  OTHER POSSIBLE OPERATIONS, (look internet, I have mentioned what is possible). |
| Ctrl  +  Click, to visit these websites | * [ArrayList To String Conversion](https://www.softwaretestinghelp.com/java-arraylist-conversions/#ArrayList_To_String_Conversion) * [String To ArrayList Conversion](https://www.softwaretestinghelp.com/java-arraylist-conversions/#String_To_ArrayList_Conversion) * [Convert list To ArrayList In Java](https://www.softwaretestinghelp.com/java-arraylist-conversions/#Convert_list_To_ArrayList_In_Java) * [Convert ArrayList To Set In Java](https://www.softwaretestinghelp.com/java-arraylist-conversions/#Convert_ArrayList_To_Set_In_Java) * [Convert Set To ArrayList In Java](https://www.softwaretestinghelp.com/java-arraylist-conversions/#Convert_Set_To_ArrayList_In_Java) |

Iterator :

|  |  |
| --- | --- |
| **Iterator<String> itr= \_\_\_\_\_\_ .iterator();** // used for iteration in any Collections (mostly AL and LL) | |
| 1. | .hasNext(); // checks and returns if there is a value next to current pointer. |
| 2. | .next(); // next moves the pointer to next element. |

HashMap : // Code in Eclipse about hashing operations.

|  |  |
| --- | --- |
|  | HashMap<String, Integer> \_\_\_\_\_\_=new HashMap(); |
|  | .put(key, value); // to insert value, for a key, no duplicate keys allowed. But duplicate values allowed |
|  | .get(); // to get a values based on key. null if this key dont exist. So use technique i discussed in CN book. |
|  | .remove(); // remove a value based in key. |
|  | .containsKey(); // To check if contains a key. This returns true/false |
|  | .containsValue(); // To check if contains a value. This returns true/false |
|  | .values(); // gives values. |
|  | .keySet(); // gives keys |
|  | .entrySet(); // Collection of above two. |
|  | for(String a:\_\_\_\_.keySet()) {  System.out.println(a);  }  for(int b:\_\_\_\_.values()) {  System.out.println(b);  } |
|  | Set< > \_\_\_=*hash name* .keySet(); // We can store all keys in a set. |
|  | Syso(hashmap name); // we can directly print hashmap. |
|  | For(Map.Entry obj: mapName.entrySet()){  .getKey() and .getValue()  } |

Learn TreeMap and TreeSet later. It is part of collection we will use once we move on to the advance section of the DSA.

**Java In-Built Functions**

STRING :

|  |  |
| --- | --- |
| 1 | .charAt(*index*); // to get character at specific location in string. Using this & loop we can traverse too. |
| 2 | .length(); // gives length of string. (NOTE: in array we use length Ie. We don’t write round brackets in array) |
| 3 | .codePointAt(index); // This gives asci value of char at specified index. (asci is of **int type**) |
| 4 | Str1.compareTo (Str2); // zero returned if equal. A value less than 0 is returned if the string1 is less than string2 (less characters). and a value greater than 0 if the string1 is greater than string2 (more characters). |
| 5 | Str1.compareToIgnoreCase(Str2); // same as above, just ignores the upper and lower case. |
| 6 | .contains(); // Check if string contains given sequence of characters. true/false  .startsWith(); // Check if string start with a given sequence. |
| 7 | .concat(); // Appends at end of string. |
| 8 | .equals(); // Compare 2 stings. True/false. No 0 +ve or -ve like compareTo(). USE THIS INSTEAD OF compareTO(); |
| 9 | .endsWith(); // Check if string ends with specified string. True/False |
| 10 | \_\_\_\_\_.toUpperCase(); OR \_\_\_\_\_\_.toLowerCase(); // To convert string to upper/lower case. |
|  | Character.isUpperCase(); OR Character.isLowerCase(); // to check for case of character. |
| 11 | type.toString(); // we can convert any datatype to sting. type is wrapper class. |
| 12 | char \_\_\_[]=\_\_\_\_.toCharArray(); // To convert string to char array. |
| 13 | .substring(Start, End); // Gives substring of original string. Start is inclusive, End is Exclusive. ( when giving give End index+1. )  .substring(Start); // If we mention only start it goes to end by default. |
| 14 | .isEmpty(); // Checks of string is empty. True/ False |
| 15 | String A="abcdefAbcD";  String X=A.replaceAll("[AEIOUaeiou]", ""); // [] Brackets means each element is considered.    System.out.println(X); // OP- bcdfbcD |
| 16 | String Buffer  StringBuffer sb=new StringBuffer();  // StringBuffer sb=new StringBuffer("String here");  sb.append("Sanket"); // .appned("\_\_\_") To append string  sb.insert(1,"Ghost"); // To insert string at index of another string  sb.replace(1,6,"OK"); // To replace string from beginIndex and endIndex-1  sb.delete(1,3); // To delete string from beginIndex and endIndex-1 (so do endindex+1)  sb.reverse(); // To reverse  System.out.println(sb);  System.out.println(sb.length()); // To get length |
| 16 | .indexOf("\_\_\_")); // The indexOf() method returns the position of the first occurrence of specified string.  lastIndexOf("planet") // returns the position of the last occurrence of specified string. |

ARRAY:

|  |  |
| --- | --- |
|  | .binarySearch(array, key); // To perform a binary search. |
|  | .length // to get array length. |
|  | Arrays.toString(\_\_\_\_); // To convert array to string. |
|  | Arrays.sort(\_\_\_); // To sort a array in ascending order. It can sort both int and char array. Both ascending.  Arrays.sort(\_\_\_, start index, end index); // To sort in specified index. Just like substring, End index us exclusive. when giving give End index+1. |
|  | Arrays.mismatch(no,no1)); // Finds and returns the index of the first unmatched element between the two specified arrays. |

OTHER USEFUL, Conversion FUNCTIONS:

STRING TO OTHER Data-types,

datype \_\_\_= wrapper.ParseInt/Float/Double/Long( *string name* ); // STRING TO OTHER Data-types,

char \_\_\_[]=\_\_\_\_.toCharArray(); // To convert string to char array.

OTHER data-types to STRING,

String \_\_\_\_=type.toString(\_\_\_); // here type can be any wrapper.

**String** \_\_\_\_ = Arrays.toString(); // to convert array to string.

**Java Bit magic**

To perform manipulation of individual of a number.

1. **OR |**

if any one of the bits is 1, it gives 1, else it gives 0.

0101

|

0111

\_\_\_\_\_

0111

1. **AND &**

If both are 1 it gives 1 or it gives 0

0101

&

0111

\_\_\_\_\_\_

0101

1. **XOR ^**

if corresponding bits are different, it gives 1, else it shows 0.

0101

^

0111

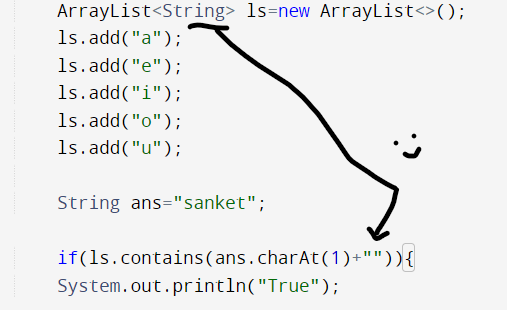
\_\_\_\_\_

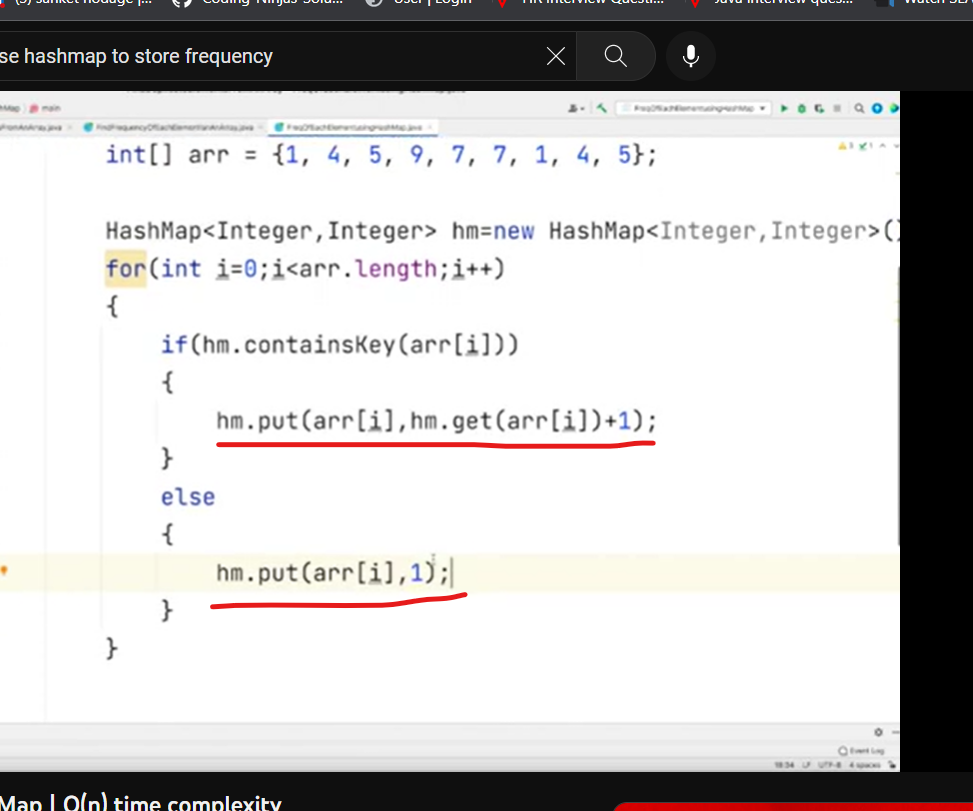
0010

1. **Compliment ~**

it makes every 0 to 1, and every 1 to 0.

**My understood ALGO, DS and OTHER Coding tricks.**



Checking string using arraylist function.

Using hashmap top count frequency.