To get input,

Scanner sc = new Scanner(System.in);

int intData = sc.nextInt(); //to get integer intData

sc.nextLine(); //to get new line input, newline is the delimiter

sc.next(); //to get string input, space “ “ is the delimiter

break with label, it must be near the loop

== compares two references are equal or not, whereas equals() checks if the values are equal

Classes can be instantiated by new operator, clone or Class.forName/newInstance()

array :- arr.length

Collections:= entity.size()

List contains ALVS(ArrayList, LinkedList, Vector, Stack)

ArrayList list = new ArrayList();

Iterator iter = list.iterator();

while(iter.hasNext()) {

System.out.println(iter.next())

}

or

System.out.println();

for(String element : queue) {

System.out.println(element);

}

Similar to next() and hasNext(), previous() and hasPrevious() are also present

Stack operations: push, peek, pop

Queue; PriorityQueue, ArrayDeque

Set: HLT(HashSet, LinkedHashSet, TreeSet)

HashSet: it uses hash table for storage and the elements stored must be unique.

LinkedHashSet: It is similar to HashSet, but it maintains insertion order and permits null values.

SortedSet: alternate of set and the items are sorted.

TreeSet : It extends SortedSet and access and retrieval times are faster.

ArrayList:

- It is dynamic array but with no limit

- It maintains insertion order

- Can contain duplicate data.

- While removing an item, it is bit slower than LinkedList because a lot of shifting need to occur

- Does not support concurrent access.

- Generics ArrayList<String> list = new ArrayList<String>();

- list.add("abc");

- list.set(index, value); => list.set(1, "def");

- list.get(1);

- To sort ArrayList, Collections.sort(list);

- list1.addAll(list2) to add/merge two LinkedHashSet

- To reverse a list, Collections.reverse(list);

- To remove duplicates in list, Set<String> s = new LinkedHashSet<String>(list);

LinkedList:

- Doubly linked list.

- Can contain duplicate data.

- maintains insertion order

- Does not support concurrent access.

- Fast because read and write takes place in O(1)

- Supports add(), addFirst(), addLast()

Shallow copy vs deep copy c++:

- In shallow copy, the values of object are copied and dynamically created memory locations are shared between two objects whereas

in deep copy, dynamic memory locations are created newly and copied using copy constructor

Shallow copy vs deep copy java:

- Similar to above(c++) but the reference is shared in shallow copy

Array vs List:

When more access operations happens, it is better to use array(coz it is O(1)). If add/delete operations are happening more, it is

better to use list(O(1))

List vs Set:

List stores duplicate data whereas set stores unique data

List stores insertion order, set is random

PriorityQueue:

Ascending order PriorityQueue and descending order PriorityQueue

For custom objects, the class must implement Comparable<ClassName> and override compareTo method

HashMap:

for(Map.Entry<Integer,Integer> entrySet : freq.entrySet()) {

System.out.println("Frequency of " + entrySet.getKey() + " is " + entrySet.getValue());

}

HashMap vs TreeMap:

Hashmap can contain one null key whereas TreeMap cannot contain null key

HashMap maintains no order whereas TreeMap by default stores in Ascending order

str.substring(i, j); returns string in str between i and j-1

To convert String to char[], str.toCharArray()

To convert String to Character[], iterate and set, characterArray = str.charAt(i)

To convert char[] to String, String.valueOf(chArr);

Character.toLowerCase()

Character.toUpperCase()

str1.concat(str2)

str1.indexOf(str2) != -1 means str2 is present in str1

Permutation of string:

Number of levels is same as no. of characters in the string

try {

statements;

throw new Exception(“Invalid number”);

}catch(Exception ex) {

System.out.println(ex);

ex.printStackTrace();

}

static blocks gets executed even before main function in a class

static arrays should be initialized inside a static block

formula to find day of the week for any date:

String []DAYS;

            DAYS = new String[]{"SUNDAY","MONDAY","TUESDAY","WEDNESDAY","THURSDAY","FRIDAY","SATURDAY"};

        day += ((month < 3) ? year-- : (year - 2));

        int dayOfWeek = ((((23 \* month) / 9) + day + 4 + (year / 4)) - (year / 100) + (year / 400)) % 7;

        return DAYS[dayOfWeek];