a not followed by b a(?!b)

a not followed by any non space character a(?![\s])

take all the element before raj in a paragraph (.\*raj)

print all the string having raj in paragraph (.\*raj.\*)

$ + \* . | \d \D \w \W \s \S (?i) (?s) (?m) (.\*).(txt|csv|xlsx)

(<\/body>) means closing body tag.

[^abc] [abc][vz]

byte range -128 to 127 short 16 bit int 32 bit float 32 bit

long double. Both are of 64 bits

spring-----dispatcher servlet restful----servlet container

a==b return ture on;

difference betweem equals and ==

equals-.used for object comparision matches string content

== - operqator used for primitive type comparision matches references

Hashcode- used for object enequility.If you object have same hashcode means they must have same hashvalue.s

Hashmap works on principal pf hashing.It uases hashfunction which is hashcode which return hashvalue>with the help of hashvalue bucket location can be find.Bucket contains multipal key value pair as Map.Entry class.It stores value in term of linked list.Bucket has hashvalue key value and bucket index.

Nested class:and inner class are same

Why non static inner class doesnot have static reference

Because an instance of an inner class is implicitly associated with an instance of its outer class, it cannot define any static methods itself

Theyn why fo defione inside inner class directly assign it to outer clas Anyhow instance of inner class is implicitly associated with outer class.

Annonymus inner class: is the class where instanciation and calling of a class happens in single line.

You can create instancewith the help of anonymous inner class

/\*public class NameComparator implements Comparator<Employee> {

@Override

public int compare(Employee emp1, Employee emp2) {

return emp1.getName().compareTo(emp2.getName()); } }\*/ Collections.sort(list,Employee.JobTitleAndageComaprator);

Map.Entry object which contains both key and value object.

The interface is used to define API. It tells about the contract your classes will follow.It is pure abstract class.It also supports abstraction because a client can use interface method to leverage multiple implementations e.g. byusing List interface you can take advantage of random access of ArrayList as well as flexible insertion and deletion of LinkedList. The interface doesn't allow you to write code to keep things abstract but from Java 8 you can declare static and default

methods inside interface

which are concrete.

Stirng and String Buffer(immutable)

==============================================================================

equals- compare actual content of Sting

== compare memory address of the object

hashcode return hashvalue of the object. It is hashfunction which return hash value.If you call get mehtod, hashfunction willbe internally called and the hashvalue wil find the bucker location where key values are stored as value . Bucket contains multipal key value pair .Pair stored as lineked list.

Key and value are stored in value object.

contract-

1>If two object are equal then hashcode must be same.

2>if Hashcode of object are equal,then hashcode may or maynot be same.

=============================================================================

Throwable is super class of Exception as well as Error.

==============================================================================

difference between arrayList and LinkedList

In case of element access: Arralist takes O(1)

LinkedLiat takes O(n)

Delection : arraylist takes accessTime+O(n)

linkedList takes accessTime+(1)

The ArrayList must move all the elements from array[index] to array[index-1]

starting by the item to delete index. The LinkedList should navigate until that

item and then erase that node by decoupling it from the list.

Insertion : arrayList takes O(n)

LinkedList takes O(1)

================================================================================

Tree Set-implement serializble and internally it uses map to add element

HashSet is internally implemented using an HashMap. Since a Map needs key and value, a default value is used for all keys. Similar to HashMap, HashSet doesn't allow duplicate keys and only one null key, I mean you can only store one null object in HashSet.

TreeSet dosenot allow null value.

==================================================================================

Key here is to check whether candidate uses ArrayList's remove() or Iterator's remove().

===========================================================================

61) Is it possible for two unequal objects to have the same hashcode?

Yes, two unequal objects can have same hashcode that's why collision happen in a hashmap..

To avoid collision rehashing algorithm is used.It will create different hashtable and make entry in newly created table.

Performance of hashtable depend on number of bucket and load facor which is equal to 0.75.

Load factor equals to m/n

M=no of entries n= no of indexes in table

If lf equals to one,menas entries are more ot lf equals to0.1 which result lot os iteration to get value.

the equal hashcode contract only says that two equal objects must have the same hashcode it doesn't say anything about the unequal object.

64) What is the difference between Comparator and Comparable in Java? (answer)

The Comparable interface is used to define the natural order of object ---in lang package user compareTo()

while Comparator is used to define custom order. ---in util package---compare(Object o1,Object o2)

Comparable can be always one, but we can have multiple comparators to define customized order for objects.

Each character of both the strings is converted into a Unicode value for comparison. If both the strings are equal then this method returns 0 else it returns positive or negative value.

====================================================================================

63) Can we use random numbers in the hashcode() method? (answer)

No, because hashcode of an object should be always same. See the answer to learning more about things to remember while overriding hashCode() method in Java.

======================================================

unchecked exception

nullpointer

arithatic

classnotfound

arrayoutofbondexception

Checked Exception

ioexception

sqlexception

classnotfoundexception

filenotfoundexception

======================================================

java5

geberics

autoboxing/unboxing

java 6

scripting languague support

jdbc 4.0

annotation support

java 7

swith satment with string

java nio package Files,write

try with resources

java 8

labmda expression------ parameter -> expression body

new date and time api---------LocalDate,LocalDateTime,LocalDate,LocalTime

interface with default method

java 9

money api support

=============================================================

qualifier annotation used with autowire

suppose you want to generate different object wirh same class

•In the above points we mentioned many scenarios where code has to be changed. All changes made, need to and should be tested. How can we test the Person class without including the message delivery class such as the Email? Testing, in many cases, is left as an afterthought. The way we have the Person class constructed makes it hard to test it without involving the Email class. Furthermore, how would we automate such test? How can we use JUnit (Homepage), or the like, to automate out tests?

out.println == <%= %>

request response page pagecontext config application session exception out

error handeling in jsp

<@page errorpage="error.jsp">(logic inside)

<@ page isErrorPage="true"> printing the info

====================================================

directive tells how jsp is converted into servlets

page include taglib

1>defines in entire page

2>including html or any other resource

3>used to define tah library .We use TLD to define tags

Tight coupling is when a group of classes are highly dependent on one another.

Loose coupling is achieved by means of a design that promotes single-responsibility and separation of concerns.IN spring,looking at java file u cannot know whihc method will be called becoz property is defined in java file.if you can to cople any other class you donot have to change anything in java file Just configure property of the object in

xml class.

Concrete class has all its method implemented. Abstract class all its method except some (at least one) method(s) un-implemented so that you can extend it and implement the un-implemented method.

The static variable gets memory only once in class area at the time of class loading

package com.java2novice.algos;

 ArrayList Implementation in java

|  |
| --- |
| Write a program to implement your own ArrayList class. It should  contain add(), get(), remove(), size() methods. Use dynamic array logic.  It should increase its size when it reaches threshold. |

import java.util.Arrays;

public class MyArrayList {

    private Object[] myStore;

    private int actSize = 0;

    public MyArrayList(){

        myStore = new Object[10];

    }

    public Object get(int index){

        if(index < actSize){

            return myStore[index];

        } else {

            throw new ArrayIndexOutOfBoundsException();

        }

    }

    public void add(Object obj){

        if(myStore.length-actSize <= 5){

            increaseListSize();

        }

        myStore[actSize++] = obj;

    }

    public Object remove(int index){

        if(index < actSize){

            Object obj = myStore[index];

            myStore[index] = null;

            int tmp = index;

            while(tmp < actSize){

                myStore[tmp] = myStore[tmp+1];

                myStore[tmp+1] = null;

                tmp++;

            }

            actSize--;

            return obj;

        } else {

            throw new ArrayIndexOutOfBoundsException();

        }

    }

    public int size(){

        return actSize;

    }

    private void increaseListSize(){

        myStore = Arrays.copyOf(myStore, myStore.length\*2);

        System.out.println("\nNew length: "+myStore.length);

    }

    public static void main(String a[]){

        MyArrayList mal = new MyArrayList();

        mal.add(new Integer(2));

        mal.add(new Integer(5));

        mal.add(new Integer(1));

        mal.add(new Integer(23));

        mal.add(new Integer(14));

        for(int i=0;i<mal.size();i++){

            System.out.print(mal.get(i)+" ");

        }

        mal.add(new Integer(29));

        System.out.println("Element at Index 5:"+mal.get(5));

        System.out.println("List size: "+mal.size());

        System.out.println("Removing element at index 2: "+mal.remove(2));

        for(int i=0;i<mal.size();i++){

            System.out.print(mal.get(i)+" ");

        }     } }

The java.util.Queue interface is a subtype of the java.util.Collection interface. It

represents an ordered list of objects just like a List, but its intended use is slightly

different. A queue is designed to have elements inserted at the end of the queue, and

elements removed from the beginning of the queue. Just like a queue in a

supermarket.

Here is a list of the topics covered in this text:

1. Queue Implementations

2. Adding and Accessing Elements

3. Removing Elements

4. Generic Queues

5. More Details in the JavaDoc

Queue Implementations

Being a Collection subtype all methods in the Collection interface are also available

in the Queue interface.

Since Queue is an interface you need to instantiate a concrete implementation of the

interface in order to use it. You can choose between the following Queue

implementations in the Java Collections API:

java.util.LinkedList

java.util.PriorityQueue

LinkedList is a pretty standard queue implementation.

PriorityQueue stores its elements internally according to their natural order (if they

implement Comparable), or according to a Comparator passed to the PriorityQueue.

There are also Queue implementations in the java.util.concurrent package, but I will

leave the concurrency utilities out of this tutorial.

Here are a few examples of how to create a Queue instance:

Queue queueA = new LinkedList();

Queue queueB = new PriorityQueue();

**Adding and Accessing Elements**

To add elements to a Queue you call its add() method. This method is inherited from the Collection interface. Here are a few examples:

Queue queueA = new LinkedList();

queueA.add("element 1");

queueA.add("element 2");

queueA.add("element 3");

The order in which the elements added to the Queue are stored internally, depends on the implementation. The same is true for the order in which elements are retrieved from the queue. You should consult the JavaDoc's for more information about the specific Queue implementations.

You can peek at the element at the head of the queue without taking the element out of the queue. This is done via the element() method. Here is how that looks:

Object firstElement = queueA.element();

To take the first element out of the queue, you use the remove() method which is described later.

You can also iterate all elements of a queue, instead of just processing one at a time. Here is how that looks:

Queue queueA = new LinkedList();

queueA.add("element 0");

queueA.add("element 1");

queueA.add("element 2");

//access via Iterator

Iterator iterator = queueA.iterator();

while(iterator.hasNext(){

String element = (String) iterator.next();

}

//access via new for-loop

for(Object object : queueA) {

String element = (String) object;

}

When iterating the queue via its Iterator or via the for-loop (which also uses the Iterator behind the scene, the sequence in which the elements are iterated depends on the queue implementation.

**Removing Elements** To remove elements from a queue, you call the remove() method. This method removes the element at the head of the queue. In most Queue implementations the head and tail of the queue are at opposite ends. It is possible, however, to implement the Queue interface so that the head and tail of the queue is in the same end. In that case you would have a stack.

Here is a remove example();

Object firstElement = queueA.remove();

**Generic Queue**

By default you can put any Object into a Queue, but from Java 5, Java Generics makes it possible to limit the types of object you can insert into a Queue. Here is an example:

Queue<MyObject> queue = new LinkedList<MyObject>();

This Queue can now only have MyObject instances inserted into it. You can then access and iterate its elements without casting them. Here is how it looks:

MyObject myObject = queue.remove();

for(MyObject anObject : queue){

//do someting to anObject...

}

### Question 1: How to find middle element of linked list in one pass?

One of the most popular question from data structures and algorithm, mostly asked on telephonic interview. Since many programmer know that, in order to find length of linked list we need to first traverse through linked list till we find last node, which is pointing to null, and then in second pass we can find middle element by traversing only half of length. They get confused when interviewer ask him to do same job in one pass. In order to find middle element of linked list in one pass, you need to maintain two-pointer, one increment at each node while other increments after two nodes at a time, by having this arrangement, when first pointer reaches end, second pointer will point to middle element of linked list. See this trick to[find middle element of linked list in single pass](http://javarevisited.blogspot.com/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html)for more details.

### Question 2: How to find if linked list has a loop ?

This question has bit of similarity with earlier algorithm and data structure interview question. I mean we can use two pointer approach to solve this problem. If we maintain two pointers, and we increment one pointer after processing two nodes and other after processing every node, we are likely to find a situation where both the pointers will be pointing to same node. This will only happen if linked list has loop.

### Question 3 : How to find 3rd element from end in a linked list in one pass?

This is another frequently asked linked list interview question. This question is exactly similar to[finding middle element of linked list in single pass](http://javarevisited.blogspot.sg/2012/12/how-to-find-middle-element-of-linked-list-one-pass.html). If we apply same trick of maintaining two pointers and increment other pointer, when first has moved up to 3rd element, than when first pointer reaches to the end of linked list, second pointer will be pointing to the 3rd element from last in a linked list.

### Question 4: In an integer array, there is 1 to 100 number, out of one is duplicate, how to find?

This is a rather simple data structures question, especially for this kind of. In this case you can simply add all numbers stored in array, and total sum should be equal to n(n+1)/2. Now just subtract actual sum to expected sum, and that is your duplicate number. Of course there is a brute force way of checking each number against all other numbers, but that will result in performance of O(n^2) which is not good. By the way this trick will not work if array have multiple duplicates or its not numbers forming arithmetic progression. Here is example of one way to[find duplicate number in array](http://javarevisited.blogspot.com/2012/02/how-to-check-or-detect-duplicate.html).

### Question 6 : How to reverse String in Java ?

This is one of my favorite question. Since String is one of the most important type of programming, you expect lot of question related to String any data structure interview. There are many ways to reverse Sting in Java or any other programming language, and interviewer will force you to solve this problem by using without API i.e. without using reverse() method of StringBuffer. In follow-up he may ask to reverse String using recursion as well. See[3 ways to reverse String in Java](http://javarevisited.blogspot.com/2012/01/how-to-reverse-string-in-java-using.html)to learn reversing String using both loops and[recursion in Java](http://javarevisited.blogspot.com/2012/12/recursion-in-java-with-example-programming.html).

## Question 7: Write a Java program to sort an array using Bubble Sort algorithm?

I have always send couple of questions from searching and sorting in data structure interviews. Bubble sort is one of the simplest sorting algorithm but if you ask anyone to implement on the spot it gives you an opportunity to gauge programming skills of a candidate. See[How to sort array using Bubble Sort in Java](http://java67.blogspot.com/2012/12/bubble-sort-in-java-program-to-sort-integer-array-example.html)for complete solution of this datastrucutre interview question.

### Question 8: What is the difference between Stack and Queue data structure?

One of the classical data structure interviews question. I guess every one know, No? Any way main difference is that Stack is LIFO(Last In First Out) data structure while Queue is a FIFO(First In First Out) data structure.

### Question 9: How do you find duplicates in an array if there is more than one duplicate?

Sometime this is asked as follow-up question of earlier data structure interview question, related to finding duplicates in Array. One way of solving this problem is using a[Hashtable or HashMap](http://javarevisited.blogspot.com/2013/02/how-to-get-key-from-value-in-hashtable.html)data structure. You can traverse through array, and store each number as key and number of occurrence as value. At the end of traversal you can find all duplicate numbers, for which occurrence is more than one. In Java if a number already exists in[HashMap](http://java67.blogspot.com/2013/02/10-examples-of-hashmap-in-java-programming-tutorial.html)then calling get(index) will return number otherwise it return null. this property can be used to insert or update numbers in HashMap.

### Question 10 : What is difference between Singly Linked List and Doubly Linked List data structure?

This is another classical interview question on data structure, mostly asked on telephonic rounds. Main difference between singly linked list and doubly linked list is ability to traverse. In a single linked list, node only points towards next node, and there is no pointer to previous node, which means you can not traverse back on a singly linked list. On the other hand doubly linked list maintains two pointers, towards next and previous node, which allows you to navigate in both direction in any linked list.

### Question 11 : Write Java program to print Fibonacci series ?

This is not a data structures question, but a programming one, which many times appear during data structure interview. Fibonacci series is a mathematical series, where each number is sum of previous two numbers e.g. 1,1, 2, 3, 5, 8, 13, 21. Interviewer is often interested in two things, a function which returns nth number in Fibonacci series and solving this problem using recursion in Java. Though, its easy question, recursion part often confuses beginners. See this link to[find nth Fibonacci number in Java](http://java67.blogspot.sg/2012/07/java-program-fibonacci-series-with.html).

### Question 12: Write Java program to check if a number is a palindrome or not?

This is similar to previous question, not directly related to data structures, but quite popular along with other questions. A number is called palindrome, if reverse of number is equal to number itself. Interviewer ask to solve this problem without taking help from Java API or any open source library. Any way it’s simple question, you can use division operator (/) and remainder operator (%) to solve this question. Just remember, division operator can be used to get rid of last digit e.g. 1234/10 will give you 123, and modulus operator can give you last digit e.g. 1234%10 will return 4. By the way, here is a[Java program check if number is palindrome or not](http://javarevisited.blogspot.com/2012/12/how-to-check-if-number-is-palindrome-or-not-example.html).

### Question 13 : What is binary search tree?

This is a data structure question from Tree data structures. Binary Search Tree has some special properties e.g. left nodes contains items whose value is less than root , right sub tree contains keys with higher node value than root, and there should not be any duplicates in the tree. Apart from definition, interview can ask you to implement binary search tree in Java and questions on tree traversal e.g. IN order, preorder, and post order traversals are quite popular data structure question.

### Question 14 : How to reverse linked list using recursion and iteration?

This is another good question on data structures. There are many algorithms to reverse linked list and you can search for them using google. I am thinking of writing another blog post to explain linked list reversal and will share with you later.

### Question 15: Write a Java program to implement Stack in Java?

You can implement Stack by using array or linked list. This question expect you to implement standard method provided by stack data structure e.g. push() and pop().  Both push() and pop() should be happen at top of stack, which you need to keep track. It’s also good if you can implement utility methods like contains(),isEmpty() etc. By the way JDK has java.util.Stack class and you can check it’s code to get an idea. You can also check[Effective Java book](http://www.amazon.com/dp/0321356683/?tag=javamysqlanta-20), where Josh Bloch has explains how an incorrect implementation of stack can cause memory leak in Java.