JAVA - AT&T Interview Question And Answers

Q1. Difference between a Pointer and a Reference?

ANS:
We cannot get the address of a reference like a pointer. Moreover we cannot perform pointer arithmetic with references.

Q2. Difference between TCP and UDP?

Q3. What are RESTful Web Services?

ANS:
REST or Representational State Transfer is a flexible architecture style for creating web services that recommends the following guidelines -

1. http for client server communication,
2. XML / JSON as formatting language,
3. Simple URI as address for the services and,
4. stateless communication.

Q4. What is bean auto wiring?

ANS:
The Spring container is able to auto wire relationships between collaborating beans. This means that it is possible to automatically let Spring resolve collaborators (other beans) for your bean by inspecting the contents of the BeanFactory without usingand elements

Q5. What bean scopes does Spring support? Explain them.

ANS:
The Spring Framework supports following five scopes -

1. Singleton
2. prototype
3. request

4. session
5. global-session

Q6. What are the pre-requisite for the collection to perform Binary Search?

ANS:
1. Collection should have an index for random access.
2. Collection should have ordered elements.

Q7. What is the difference between namenode and datanode in Hadoop?

ANS:
NameNode stores MetaData (No of Blocks, On Which Rack which DataNode is stored etc) whereas the DataNode stores the actual Data.

Q8. What things you would care about to improve the performance of Application if its identified that its DB communication that needs to be improved?

ANS:
1. Query Optimization (Query Rewriting , Prepared Statements)
2. Restructuring Indexes.
3. DB Caching Tuning (if using ORM)
4. Identifying the problems (if any) with the ORM Strategy (If using ORM)

Q9. What is Singleton?

ANS:
Its a Design Pattern.

Q10. What is a Singleton Class?

ANS:

Class using which only one object can be created.

Q11. What is the use of such a class?

ANS:

There could be situations where we need not create multiple objects and hence Singleton can help in saving resources by avoiding creating new objects every time a request is made. Moreover these classes can also be helpful if we want a object to be shared among threads.

Q12. Are you using Singleton in your code?

ANS:

Yes, for Database connection and Property files.

Q13. Write the code for a Singleton Class?

ANS:

```
class Singleton {
private static volatile Singleton instance = null;
private Singleton(){}
public static Singleton getInstance() {
if (instance == null) {
synchronized(Singleton.class) {
if (instance== null)
instance = new Singleton();
}
}
return instance;
}
}
```

Q14. Why have we used synchronized here?

ANS:

getInstance method can be accessed from two points simultaneously and in such case 2 instances may get created. Synchronization will make sure that the method gets accessed one by one for each call and the same object is returned for the second call.

Q15. Why have we declared the instance reference volatile?

ANS:

Thats an instruction to JVM that the variable is getting accessed by multiple locations and hence do not cache it.

Q16. Can we make the reference instance non static?

ANS:

No, as non static variables cannot be accessed through static methods.

Q17. Can we have this pattern implemented using Static Class?

ANS:

Though we can implement this behavior using static class, but we should never do it.

Q18. What are the problems in implementing this patterns using static Class?

ANS:

a. We cannot achieve runtime Polymorphism or late binding as Java does not allow overriding static methods.
b. We cannot do lazy initialization as Static members are loaded during class loading only.
c. We cannot serialize as Java does not serialize static members.

Q19. Write a program to see if the number is prefect number or not?

ANS:

```
#include
int main() {
int num, i = 1, sum = 0;
printf(Enter a number: );
scanf(%d, &num);
while (i < num) {
if (num % i == 0) {
sum = sum + i;
}
i++;
}
if (sum == num)
printf(%d is a Perfect Number, i);
else
printf(%d is Non Perfect Number, i);

return 0;
}
```

Q20. Explain and Write Program for Selection Sort.

ANS:

Selection Sort: Selection sort is to repetitively pick up the smallest element and put it into the right position:

- Find the smallest element, and put it to the first position.
- Find the next smallest element, and put it to the second position.
- Repeat until all elements are in the right positions.

A loop through the array finds the smallest element easily. After the smallest element is put in the first position, it is fixed and then we can deal with the rest of the array. The following implementation uses a nested loop to repetitively pick up the smallest element and swap it to its final position. The swap() method exchanges two elements in an array.

```
public static void selectionSort(int[] arr)
{
   // find the smallest element starting from position i
for (int i = 0; i < arr.length - 1; i++)
{
int min = i;  // record the position of the smallest
for (int j = i + 1; j < arr.length; j++)
{
          // update min when finding a smaller element
if (arr[j] < arr[min])
min = j;
}
// put the smallest element at position i
swap(arr, i, min);
}
}
public static void swap (int[] arr, int i, int j)
{
int temp = arr[i];
arr[i] = arr[j];
arr[j] = temp;
}
```

The following tracks the code on an array with elements {38, 27, 43, 3, 9, 82, 10}.

```
{38, 27, 43,  3,  9, 82, 10},   i: 0,   min: 3, minValue:  3
{ 3, 27, 43, 38,  9, 82, 10},   i: 1,   min: 4, minValue:  9
{ 3,  9, 43, 38, 27, 82, 10},   i: 2,   min: 6, minValue: 10
{ 3,  9, 10, 38, 27, 82, 43},   i: 3,   min: 4, minValue: 27
{ 3,  9, 10, 27, 38, 82, 43},   i: 4,   min: 4, minValue: 38
{ 3,  9, 10, 27, 38, 82, 43},   i: 5,   min: 6, minValue: 43
{ 3,  9, 10, 27, 38, 43, 82},   i: 6
```

Suppose the input array has n elements. The outer loop runs n-1 rounds, roughly one for each position. Inside each outer loop, the inner loop goes through the unsorted part of the array. On average, each inner loop scans through n/2 elements. The total time is roughly t

*(n) = (n – 1) * (n/2) *k = O(n^2),*

where k denotes the number of basic operations inside each inner loop; the constants are absorbed in the big-Oh notion. Note that in the big-Oh notion, we only care about the dominating factor (which is n^2 in this case).

Q21. Explain and Write Program for Insertion Sort.

ANS:

Insertion Sort: Insertion sort maintains a sorted sub-array, and repetitively inserts new elements into it. The process is as following:

- Take the first element as a sorted sub-array.
- Insert the second element into the sorted sub-array (shift elements if needed).
- Insert the third element into the sorted sub-array.
- Repeat until all elements are inserted.

The following insertionSort() method implements insertion sort. It uses a nested loop to repetitively insert elements into the sorted sub-array.

```
public static void insertionSort(int[] arr)
{
 for (int i = 1; i < arr.length; i++)
 {
 // a temporary copy of the current element
 int tmp = arr[i];
int j;
// find the position for insertion
for (j = i; j > 0; j--)
{
if (arr[j - 1] < tmp)
break;
// shift the sorted part to right
arr[j] = arr[j - 1];
}
 // insert the current element
arr[j] = tmp;
}
}
```

The following tracks the code on an array with elements {38, 27, 43, 3, 9, 82, 10}.

```
{38, 27, 43,  3,  9, 82, 10}
{27, 38, 43,  3,  9, 82, 10},   i: 1
{27, 38, 43,  3,  9, 82, 10},   i: 2
{ 3, 27, 38, 43,  9, 82, 10},   i: 3
{ 3,  9, 27, 38, 43, 82, 10},   i: 4
{ 3,  9, 27, 38, 43, 82, 10},   i: 5
{ 3,  9, 10, 27, 38, 43, 82},   i: 6
```

Suppose the array length is n. The outer loop runs roughly n times, and the inner loop on average runs n/2 times. The total time is about

*t(n) = n * (n/2) = O(n^2).*

In terms of the efficiency, this is the same as selection sort.

Q22. Explain and Write Program for Bubble Sort.

ANS:

Bubble Sort: Bubble sort repetitively compares adjacent pairs of elements and swaps if necessary.

- Scan the array, swapping adjacent pair of elements if they are not in relative order. This bubbles up the largest element to the end.
- Scan the array again, bubbling up the second largest element.
- Repeat until all elements are in order.

The following bubbleSort() method implements bubble sort. It uses a nested loop to repetitively swap elements and bubble up the largest elements one by one.

```
public static void bubbleSort (int[] data)
{
```

```
for (int i = data.length - 1; i >= 0; i--)
{
 // bubble up
for (int j = 0; j <= i - 1; j++)
{
if (data[j] > data[j + 1])
swap(data, j, j + 1);
}
}
}
```

The following tracks the code on an array with elements {38, 27, 43, 3, 9, 82, 10} for three rounds of bubbling.

```
{38, 27, 43,  3,  9, 82, 10},  i: 6
{27, 38, 43,  3,  9, 82, 10},   j: 0
{27, 38, 43,  3,  9, 82, 10},   j: 1
{27, 38,  3, 43,  9, 82, 10},   j: 2
{27, 38,  3,  9, 43, 82, 10},   j: 3
{27, 38,  3,  9, 43, 82, 10},   j: 4
{27, 38,  3,  9, 43, 10, 82},   j: 5, i: 5
{27, 38,  3,  9, 43, 10, 82},   j: 0
{27,  3, 38,  9, 43, 10, 82},   j: 1
{27,  3,  9, 38, 43, 10, 82},   j: 2
{27,  3,  9, 38, 43, 10, 82},   j: 3
{27,  3,  9, 38, 10, 43, 82},   j: 4, i: 4
{3,  27,  9, 38, 10, 43, 82},   j: 0
{3,  9,  27, 38, 10, 43, 82},   j: 1
{3,  9,  27, 38, 10, 43, 82},   j: 2
{3,  9,  27, 10, 38, 43, 82},   j: 3
```

Suppose the array length is n. The outer loop runs roughly n times, and the inner loop on average runs n/2 times. The total time is about

$t(n) = n * (n/2) = O(n^2)$.

In terms of the efficiency, this is the same as selection sort and insertion sort.

Q23. Explain and Write Program for Binary search?

ANS:

```
import java.util.Arrays;
public class BinarySearch {

// return the index of the key in the sorted array a[]; -1 if not found
public static int search(String key, String[] a) {
return search(key, a, 0, a.length);
}
public static int search(String key, String[] a, int lo, int hi) {
// possible key indices in [lo, hi)
if (hi <= lo) return -1;
int mid = lo + (hi - lo) / 2;
int cmp = a[mid].compareTo(key);
if (cmp > 0) return search(key, a, lo, mid);
else if (cmp < 0) return search(key, a, mid+1, hi);
else return mid;
}


// whitelist, exception filter
public static void main(String[] args) {
In in = new In(args[0]);
String s = in.readAll();
String[] words = s.split(s+);
System.err.println(Done reading words);
// sort the words (if needed)
Arrays.sort(words);
System.err.println(Done sorting words);
// prompt user to enter a word and check if its there
while (!StdIn.isEmpty()) {
String key = StdIn.readString();
if (search(key, words) < 0) StdOut.println(key);
}
}
}
```