Virtusa Interview Questions and Answers

Introduction

If you ask seniors about interviews at Virtusa, you will hear mixed experiences. Some say the panel was friendly. Others say the questions went deep into basics. Both can be true.

One candidate shared that his interview began casually with a discussion about his final year project. Within minutes, the panel started asking why he selected a particular architecture and how the system would behave under heavy load. The tone stayed calm, but the questions demanded clarity.

That is usually how it works. They move from simple to specific. They check whether you understand what you claim to know. Not just theory, but application.

The sections that follow cover practical interview questions with clear, straightforward answers so you can prepare without overcomplicating your approach.

Virtusa Interview Process for Freshers

Freshers who attend a campus drive with Virtusa often expect a quick screening and a final discussion. In reality, the process moves in small stages, and each one filters candidates for a reason.

- **Assessment Round:** Most drives begin with an online test. You may see aptitude questions, basic reasoning, and a few technical problems. For developer roles, a coding section is common. Some students finish early. Others struggle with time. Managing pressure matters here.
- **Technical Conversation:** Those who clear the test are invited for a technical discussion. It rarely feels like a rapid fire round. The interviewer may start with your project and slowly explore deeper areas. If you mention a tool or concept, be ready to explain how it works in practice.
- **Final Discussion:** The last round is usually more about you as a person. Communication, adaptability, and willingness to learn are observed closely. Questions about relocation or teamwork can come up naturally in conversation.

The structure stays fairly consistent, but the experience can vary. Solid basics and calm explanation make a real difference.

Virtusa Eligibility Criteria for Students and Freshers

Before sending your resume to Virtusa, pause for a moment and check whether you match the basic requirements. Many students rush to apply and only later realise they missed a simple condition.

- **Degree Requirement:** For most entry level technical roles, a degree in Engineering, Computer Science, IT, or a related stream is expected. The exact branch depends on the project needs.
- **Academic Scores:** There is usually a minimum percentage mentioned for 10th, 12th, and graduation. It does not mean you must be a topper. Still, consistent scores make shortlisting easier.
- **Backlogs Status:** Active arrears at the time of interview are generally not preferred. Cleared backlogs may be accepted, but policies differ by drive.
- **Year of Passing:** Campus drives normally target specific batches. If you belong to an earlier year, eligibility may vary.
- **Basic Skill Readiness:** Knowing one programming language, understanding fundamentals, and being able to explain your project clearly often matter more than just meeting the minimum marks.

Eligibility rules can shift slightly depending on the hiring campaign. Reading the official notification carefully saves confusion later.

List of Virtusa Interview Questions for Freshers

1. Describe Abstraction.

2. What does encapsulation mean?

3. Define view.

4. How do void pointers work?

5. Define a copy constructor.

6. Describe a virtual function.

7. Define a transaction.

8. Explain access specifiers.

9. What qualities do ACIDs have?

10. Describe data warehousing.

Virtusa Technical Interview Questions and Answers for Freshers

1. Describe Abstraction.

The feature of Java called Data Abstraction allows the user to see only the most important information.

2. What does encapsulation mean?

Encapsulation, which in Java refers to the grouping of data and methods that alter that data into a single unit known as a class, is a fundamental concept in object-oriented programming (OOP).

3. Define view.

In SQL, views are a type of virtual table. Rows and columns in a view are identical to those in an actual database table. By choosing fields from one or more database tables, we can create a view. A view may contain every row in a table or just certain rows determined by predefined criteria.

4. How do void pointers work?

A pointer without a corresponding data type is called a void pointer. Any type of address can be stored in a void pointer, and it can be typecast to any type.

5. Define a copy constructor.

An object of the same class is used by a member function called a copy constructor to initialize another object. To put it simply, a copy constructor is a constructor that uses an already-generated object of the same class as its initialization when building an object.

6. Describe a virtual function.

Member functions that are declared inside base classes and redefined (overridden) with derived classes are referred to as virtual functions, also called virtual methods.

7. Define a transaction.

A collection of database actions known as a database transaction must be handled as a whole, meaning that either every operation is carried out

or not. A bank transaction from one account to another can serve as an illustration. It is necessary to do either all of the credit and debit operations or none of them.

8. Explain access specifiers.

Special keywords called “access specifiers” are used to define or regulate an entity’s accessibility, such as that of classes, methods, and so forth. Access modifiers or specifiers include things like Private, Public, and Protected.

Encapsulation and data hiding, two essential elements of OOPs, are mostly accomplished via these access specifiers.

9. What qualities do ACIDs have?

A collection of characteristics known as ACID (Atomicity, Consistency, Isolation, Durability) ensures the dependable processing of database transactions.

10. Describe data warehousing.

A data warehouse is distinct from a database management system (DBMS). It houses enormous volumes of data that are usually gathered from several heterogeneous sources, such as files, DBMSs, etc.

List of Virtusa Interview Questions for Experienced

11. Decode a string that is encoded as a count that follows a substring recursively.

12. Reverse a Linked List

13. Sort an array of 0s, 1s and 2s

14. Return the missing number from an array called num that contains n distinct numbers in the range [0, n].

15. Two non-empty linked lists representing non-negative integers are provided to you. Every list has a single digit at each node, arranged in the least important digit's reverse order. To retrieve the total as a linked list, add the two numbers.

16. Construct a program that reverses x's digits so that the result falls inside the bounds of a signed 32-bit integer. Return 0 if reversing x makes the value jump outside of the range.

17. Create a program that will return true if any element in the array nums appears at least twice. Return false if each element is unique.

18. Find out if a binary tree is a BST, given its root. Next, ascertain if it is a legitimate BST. A binary search tree (BST) is a tree whose subtrees are binary search trees in both cases, as well as BSTs in each node's subtrees.

19. Implement a binary search with Java.

20. When should a function throw an exception?

21. Assuming the array is initially positioned in that direction, rotate it by k degrees to the right. For that, create a program.

Virtusa Technical Interview Questions and Answers for Experienced

11. Decode a string that is encoded as a count that follows a substring recursively.

Utilizing two stacks. One for letters and another for integers is the plan.

Proceed to traverse the string.

- Put any number we come across into the integer stack, and any letter (a to z) or open bracket (‘[‘) should go into the character stack.
- Pop the character from the character stack if a close bracket (‘]’) is detected, and continue doing so until the character stack contains no more open brackets (‘[‘). Pop the top element, let’s say n, from the integer stack as well. The popped character should now be repeated n times in a string. Proceed to push every character in the string into the stack.

12. Reverse a Linked List

To solve the issue, take the following actions:

Set up three-pointers first. curr as head, next as NULL, and prev as NULL.

Go through the linked list iteratively. Repeat the following in a loop:

Store the next node before altering the next of curr.

next = curr -> next

Update the following curr pointer to the previous

curr -> next = prev

Update the following curr pointer to the previous

prev = curr

curr = next

13. Sort an array of 0s, 1s and 2s

The problem is comparable to resolving.

Three colors were used to pose the dilemma in this case: 0', 1', and 2'. There are four sections to the array:

arr[1] to arr[low – 1]

arr[low] to arr[mid – 1]

arr[mid] to arr[high – 1]

arr[high] to arr[n]

Change the element to the low range if the ith element is 0.

In a similar vein, leave it alone if the element is 1.

Replace the element with one from the high range if it is 2.

14. Return the missing number from an array called num that contains n distinct numbers in the range [0, n].

Example: [0,6,8,5,1,2,3,4] 7 is the missing number.

This difficulty can be resolved with the aid of a hash table that records numbers between 0 and n. And we will return any number that remains after marking for marking. However, this requires more room. Therefore, we can use mathematics to calculate the sum of numbers from an array to the actual sum that must exist from (1 – n) to optimize this. Thus, the code fix will be:

```
public int missingNumber(int[] nums) {

    int sumArray = 0;

    int n = nums.length;

    for(int i = 0; i<n; i++)

        sumArray += nums[i];

    int ActualSum = (n*n + n)/2;

    return ActualSum – sumArray;

}
```

15. Two non-empty linked lists representing non-negative integers are provided to you. Every list has a single digit at each node, arranged in the least important digit's reverse order. To retrieve the total as a linked list, add the two numbers.

```
class Solution {
```

```
private int val = 0;

private int carry = 0;

private ListNode reverse(ListNode head){

    if(head == null || head.next == null)

        return head;

    ListNode t = reverse(head.next);

    head.next.next = head;

    head.next = null;

    return t;

}

private ListNode addNumber(ListNode l1, ListNode l2){

    if(l1 == null && l2 != null){

        val = l2.val + carry;

        carry = val/10;

        ListNode n = new ListNode((val%10), addNumber(l1, l2.next));
```

```
    return n;
}
else if(l1 != null && l2 == null){
    val = l1.val + carry;
    carry = val/10;
    ListNode n = new ListNode((val%10), addNumber(l1.next, l2));
    return n;
}
else if(l1 != null && l2 != null){
    val = l1.val + l2.val + carry;
    carry = val/10;
    ListNode n = new ListNode((val%10), addNumber(l1.next, l2.next));
    return n;
}
else{
```

```
            if(carry != 0)
                return new ListNode(carry);
            return null;
        }
    }
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        l1 = reverse(l1);
        l2 = reverse(l2);
        l1 = addNumber(l1, l2);
        return reverse(l1);
    }
}
```

16. Construct a program that reverses x's digits so that the result falls inside the bounds of a signed 32-bit integer. Return 0 if reversing x makes the value jump outside of the range.

Example:

54678 is the input. 87645 is the output.

2147483647 is the input. Because reversing it goes out of bounds to an integer, the output is 0.

The integer data type does not allow us to store the maximum integer because we also need to store it. as it crosses the boundary. Thus, we can employ easily recognized data types, either long or short. The char data type will be used in this solution to do that. This is the answer to that, then.

```
public int reverse(int x) {

    char [] ans = Integer.toString(x).toCharArray();

    int maxAllowedPos = 2147483647;

    int maxAllowedNeg = -2147483648;

    if(x > 0){

        int i = 0, j = ans.length-1;

        while(i < j){

            char c = ans[i];

            ans[i++] = ans[j];

            ans[j–] = c;

        }
```

```
    String an = new String(ans);
    double t = Double.parseDouble(an);
    if(t > maxAllowedPos)
        return 0;
    return (int)t;
}else{
    int i = 1, j = ans.length-1;
    while(i < j){
        char c = ans[i];
        ans[i++] = ans[j];
        ans[j–] = c;
    }
    String an = new String(ans);
    double d = Double.parseDouble(an);
    if(d < maxAllowedNeg)
```

```
            return 0;

        return (int)d;

    }

}
```

17. Create a program that will return true if any element in the array nums appears at least twice. Return false if each element is unique.

By leaving the element in the hash map, we can find a solution to this issue. Furthermore, we must return true if any of the items seem to already exist. And HashMap assists us in doing so in this case.

```
public boolean containsDuplicate(int[] arr) {

    int n = arr.length;

    if(n == 1){return false;

    HashSet<Integer> set = new HashSet<>();

    for(int i =0; i<n; i++)

    {

        if(set.contains(arr[i])){return true;}
```

```
        set.add(arr[i]);
    }
    return false;
  }
```

18. Find out if a binary tree is a BST, given its root. Next, ascertain if it is a legitimate BST. A binary search tree (BST) is a tree whose subtrees are binary search trees in both cases, as well as BSTs in each node's subtrees.

With recursion, we can solve this problem with ease. All we have to do is make sure that each subtree is a binary search tree. Thus, this is how recursion is used to approach the solution.

```
 class Solution {
   private boolean solution(TreeNode root, long left, long right){
     if(root == null)
        return true;
     if(!(root.val < right && root.val > left))
        return false;
     return ((solution(root.left, left, root.val)) &&
```

```
            (solution(root.right, root.val, right)));
    }
    public boolean isValidBST(TreeNode root) {
        return solution(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }
}
```

19. Implement a binary search with Java.

```
public boolean binarySearch(int[] arr, int low, int high, key){
        if(low > high)
            return false;
        int mid = (low+high)/2
        if(arr[mid] == key)
            return true
        else if(arr[mid] < key)
            return binarySearch(arr, mid+1, high, key);
```

```
    else

        return binarySearch(arr, low, mid-1, key);

}
```

20. When should a function throw an exception?

The keywords try and catch are used to manage exceptions.

- However, we can use the keyword throw in the method declaration if we don't want to handle the exception in the method on which the exception may arise.
- The method that is calling this one must have handled the exception, according to this.

Example:

- Think of a procedure that takes two integers, divides them, and outputs the outcome.
- Thus, the procedure will raise an exception if the caller function returns 0 in the argument.
- Therefore, we can use the throw keyword to declare the add method in this instance, indicating that it may throw an exception.
- You must deal with it if you are the one making the call.

21. Assuming the array is initially positioned in that direction, rotate it by k degrees to the right. For that, create a program.

```
public int[] rotate(int[] nums, int k) {

    int n = nums.length;
```

```
if(n == 1)
    return;
if(k > n)
    k = k%n;
int i = n-k, j = 0;
int [] auxArray = new int[k];
while(i < n){
    auxArray[j++] = nums[i++];
}
i = n-1;
while(i >= k){
    nums[i] = nums[i-k];
    i–;
}
return auArray;
```

```
}
```


Conclusion

These Virtusa interview questions and answers will be updated frequently. Get in touch to update your skills according to the requirements of top companies through our **placement training institute in Chennai.**