Java Interview Question and Answers

➢ Topic – Java LinkedList:

Java Linked List Interview Questions and Answers

---

1. What is LinkedList in Java?
Answer: LinkedList in Java is a linear data structure that uses a doubly linked list internally to store a group of elements or data.

A doubly linked list consists of a group of nodes that together represents a sequence in the list. It stores the group of elements in the sequence of nodes.

2. What is the initial capacity of Java LinkedList?
Answer: The initial capacity of linked list is zero because the size of linked list automatically grows with the addition of elements. When the element is removed from the list, its size automatically shrinks.

3. In which scenario, LinkedList is better to use than ArrayList in Java?
Answer: Java LinkedList is better to use than ArrayList when the frequent operations are addition or deletion of elements in the middle of the list.

No shifting of elements takes place after removal. Only the reference of the next and previous nodes will change.

But in the case of ArrayList, if we remove an element from the middle of ArrayList, a lot of shifting of elements will take place after removal that will reduce the performance of ArrayList.

---

4. Suppose we have a list of elements whose size will grow with time. We may also need to insert elements in between existing elements but we will very rarely need random access. Which List implementation will you prefer for it?
Answer: Java LinkedList is the best choice for the above scenario.

5. What are the interfaces that are implemented by LinkedList class in Java?
Answer: LinkedList class in Java implements List and Deque interfaces. It does not implement Random Access Interface.

---

6. What are the key features of Java LinkedList?
Answer: The key features of Java LinkedList are as follows:

a) The underlying data structure of LinkedList is a doubly LinkedList data structure.

b) LinkedList allows storing duplicate elements.

c) Null elements can be added to the linked list.

d) Heterogeneous elements are allowed in the linked list.

e) LinkedList is not synchronized. Therefore, it is not thread-safe.

f) Since LinkedList is not synchronized. Hence, its operation is faster.

g) Insertion and removal of elements in the LinkedList are fast.

h) Retrieval (getting) of elements is very slow in LinkedList.

7. What is the main difference between ArrayList and LinkedList data structures in Java?
Answer: The main differences between ArrayList and LinkedList in java are as follows:

a) An ArrayList is an index-based dynamic array whereas LinkedList is a doubly-linked list data structure.

b) ArrayList internally uses a dynamic array to store elements whereas, LinkedList internally uses a doubly linked list to store the elements.

c) ArrayList is not efficient for manipulation because it provides slow manipulation due to a lot of shifting taking place. Whereas,

LinkedList provides faster manipulation as compared to ArrayList. Therefore, it is efficient for manipulation.

d) ArrayList is better for storing and fetching (or accessing) elements (or data). Whereas, LinkedList is better for manipulating data.

e) ArrayList uses random access but LinkedList does not.

f) Since ArrayList stores only object, therefore, it takes less memory. But, LinkedList stores the object and the address of that object, so it takes more memory.

g) ArrayList implements only the List interface. So, it acts as a list only. Whereas, LinkedList implements both List and Deque interfaces. Therefore, it acts as list and queue both.

---

8. How does Insertion and deletion work in Java LinkedList?
Answer: Go to this tutorial for understanding: LinkedList in Java
9. What will be the output of the following program?

```
import java.util.LinkedList;

import java.util.List;

public class LinkedListTest {

public static void main(String[] args)

{

  List<String> list = new LinkedList<String>();

   list.add("John");

   list.add(1, "Herry");

   list.add(3, "Ivaan");

   list.add("Deep");

   System.out.println(list);

  }

}
```

Answer: Output: IndexOutOfBoundsException because the size of LinkedList is 2.

10. What is the output of the following code snippet?

```
import java.util.LinkedList;

import java.util.List;

public class LinkedListTest {

public static void main(String[] args)

{

  List<String> list = new LinkedList<String>();

    list.add("John");

    list.add(1, "Herry");

    list.add(0, "Ivaan");

    list.add(2, "Deep");

  System.out.println(list);

  }

}
```

Answer: Output: [Ivaan, John, Deep, Herry]

11. Can we traverse elements of Linked List using ListIterator?
Answer: Yes, we can traverse elements of linked list using ListIterator.

12. What the following code snippet will print?

```
import java.util.LinkedList;

import java.util.List;

public class LinkedListTest {

public static void main(String[] args)

{
 List<String> list = new LinkedList<String>();

   list.add("Herry");

   list.add(0, "John");

   list.add("null");

   list.add(1, "John");

  System.out.println(list.indexOf("Herry"));

  System.out.println(list.indexOf("null"));

  System.out.println(list.lastIndexOf("John"));

  System.out.println(list.indexOf(list));

 }

}
```

Answer: Output: 2, 3, 1, -1.

---

13. What will the following code snippet display output?

```
import java.util.LinkedList;

import java.util.List;

public class LinkedListTest {

public static void main(String[] args)

{

  List<String> list = new LinkedList<String>();

     list.add("Orange");

     list.add(0, "Red");
```

```
    list.add("null");

    list.add(1, "Green");

    list.add("White");

   System.out.println(list.remove(3));

   System.out.println(list.get(2));

  }

}
```

Answer: Output: null, Orange.

14. What are the ways to iterate LinkedList in Java?
Answer: There are five ways by which we can iterate LinkedList in java. They are as follows:

- For loop
- Advanced For loop
- While loop
- Iterator
- ListIterator

For more details, go to this tutorial: How to iterate LinkedList in Java

---

15. What is the best-case scenario of using LinkedList in Java applications?
Answer: LinkedList is the best choice when our common operation is to add or remove elements in the middle of the list.

This is because no shifting of elements take place after addition or removal in the linked list. Only the reference of the next and previous nodes will change.

16. What is the worst case scenario of using LinkedList in Java?
Answer: LinkedList is the worst choice when our regular operation is to retrieve (getting) of elements from the linked list because retrieval of elements is very slow in the LinkedList as compared to ArrayList.

17. Why Linked List is not the best choice for getting elements from the list?

Answer: Linked list is not the best choice for getting elements from the list because LinkedList class does not implement Random-access interface. So, we can retrieve elements from the linked list very fast from any arbitrary position.

---

18. What will be the output of the following code snippet after successfully being compiled and run?

```
import java.util.LinkedList;

public class LinkedListTest {

public static void main(String[] args)

{

  LinkedList<String> list = new LinkedList<String>();

    list.addFirst("Harry");

    list.addFirst("Tom");

    list.addFirst("Diana");

   System.out.println(list.removeFirst());

   System.out.println(list.removeFirst());

   System.out.println(list.removeFirst());

  }

}
```

Answer: Output: Diana, Tom, Harry

19. What the following code will print the output?

```
import java.util.LinkedList;
```

```
public class LinkedListTest {

public static void main(String[] args)

{

   LinkedList<String> list = new LinkedList<String>();

    list.addFirst("Harry");

    list.addFirst("Tom");

    list.addFirst("Diana");

   System.out.println(list.removeLast());

   System.out.println(list.removeFirst());

   System.out.println(list.removeLast());

  }

}
```

Answer: Output: Harry, Diana, Tom

20. What will be the output of the following code snippet after being compiled and run?
a)

```
import java.util.LinkedList;

public class LinkedListTest {

public static void main(String[] args)

{

  LinkedList<String> list = new LinkedList<String>();

    list.addFirst("Harry");

    list.addLast("Tom");
    list.addFirst("Diana");

   System.out.println(list.removeLast());

   System.out.println(list.removeFirst());

   System.out.println(list.removeLast());

   }

}
```

Answer: Output: Tom, Diana, Harry

---

b)

```
public class LinkedListTest {

public static void main(String[] args)

{

  LinkedList<String> list = new LinkedList<String>();

    list.addFirst("Harry");

    list.addLast("Tom");

    list.addFirst("Diana");

    list.offerFirst("John");

    list.offerLast("Deep");

  System.out.println(list.removeLast());

  System.out.println(list.removeFirst());

  System.out.println(list.removeLast());

  }
}
```

Answer: Output: Deep, John, Tom

---

c)

```
public class LinkedListTest {

public static void main(String[] args)

{

   LinkedList<String> list = new LinkedList<String>();

    list.addFirst("Harry");

    list.offerFirst("John");

    list.addLast("Tom");

    list.offerLast("Deep");

     list.addFirst("Diana");



    System.out.println(list.getLast());

    System.out.println(list.getFirst());

    System.out.println(list.removeLast());

  }

}
```

Answer: Output: Deep, Diana, Deep

---

d)

```
public class LinkedListTest {

public static void main(String[] args)

{

   LinkedList<String> list = new LinkedList<String>();

  ListIterator<String> litr = list.listIterator();

    list.add("Harry");

    list.add("John");

    list.add("Tom");

    list.add("Deep");



   litr = list.listIterator();

   if(litr.next().equals("John"))

      litr.remove();

   while(litr.hasNext())

      System.out.println(litr.next());

   }

}
```

Answer: Output: John, Tom, Deep

e)

```
public class LinkedListTest {

public static void main(String[] args)
{
  LinkedList<Integer> list = new LinkedList<Integer>();
    list.add(20);
    list.add(1,30);
    list.add(0, 40);
    list.add(50);
    list.add(2, 60);
   System.out.println(list.peekFirst());
   System.out.println(list.peekLast());
   System.out.println(list.peek());
  }
}
```

Answer: Output: 40, 50, 40

---

f)

```
public class LinkedListTest {
public static void main(String[] args)
{
   LinkedList<Integer> list = new LinkedList<Integer>();
      list.add(20);
      list.add(1,30);
      list.add(0, 40);
      list.add(2, 60);

    System.out.println(list.pollFirst());

    System.out.println(list.pollLast());

    System.out.println(list.poll());

  }

}
```

Answer: Output: 40, 30, 20

---

g)

```
public class LinkedListTest {

public static void main(String[] args)

{

   LinkedList<Integer> list = new LinkedList<Integer>();

      list.add(20);

      list.add(1,30);

      list.add(0, 40);

      list.add(2, 60);

    System.out.println(list.element());

    System.out.println(list.lastIndexOf(20));

   }

}
```

Answer: Output: 40, 1

h)

```
import java.util.LinkedList;

import java.util.ListIterator;

public class LinkedListTest {

public static void main(String[] args)

{

  LinkedList<Integer> list = new LinkedList<Integer>();

      list.add(20);

      list.add(1,30);

      list.add(0, 40);

      list.add(2, 60);

  ListIterator<Integer> litr = list.listIterator(1);

  while(litr.hasNext())

        System.out.println(litr.next());

  }

}
```

Answer: Output: 20, 60, 30

Hope that this tutorial has covered almost all the important linked list interview questions in java with answers. I hope that you will have understood the answers of all the above linked list interview questions and practice the coding questions based on Java LinkedList.
All the best!!!