**Practical No. 2**

**Name:- Rohit Phatangare.**

**Roll No.:- 44.**

**Div:- A.**

**Aim**: **Assignments On List Interface**

**1:-** Write a Java Program to create a List containing Integer elements and print all the elements using for loop and for-each loop.

**Description:**

This Java code demonstrates how to create a LinkedList of integers and then traverse its elements using both a for loop and a for-each loop. Here's a description of the code:

1. **System.out.println("Write A Java Program Print List Elements :");** : This line prints a message to the console to indicate that the program will print the elements of a list.

2. **LinkedList<Integer> list = new LinkedList<>();** : This line creates a LinkedList of integers named list . LinkedList is a data structure that stores elements in a linked list fashion.

3**. list.add(10); , list.add(20); , list.add(30);** : These lines add three integer elements (10, 20, and 30) to the list .

4. **System.out.println("Traversing Elements With For Loop"**); : This line prints a message to indicate that the program is about to traverse the list elements using a for loop.

5. **for (int i = 0; i < list.size(); i++) { :** This for loop is used to traverse the elements of the list using an index-based approach.

* int i = 0; : Initializes a loop variable i to 0.
* i < list.size(); : Specifies the loop condition to iterate while i is less than the size of the list .
* i++ : Increments i after each iteration.

6. **System.out.println(list.get(i));** : Inside the for loop, this line prints the element at index i of the list using the get method.

7. **System.out.println("Traversing Elements With For Each Loop");** : This line prints a message to indicate that the program is about to traverse the list elements using a for-each loop.

8. **for (Integer t : list) {** : This for-each loop is used to traverse the elements of the list without needing an explicit index.

* Integer t : Declares a temporary variable t of type Integer to hold each element of the list during each iteration.

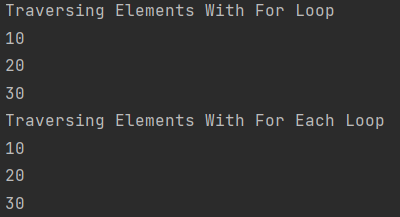
9**. System.out.println(t);** : Inside the for-each loop, this line prints the current element t to the console.

In summary, the code creates a LinkedList, adds integer elements to it, and then demonstrates two ways to traverse and print the elements: using a traditional for loop with an index and using a for-each loop. This provides two different methods for iterating through the list elements.

**Code:-**

System.*out*.println("Write A Java Program Print List Elements :");  
LinkedList<Integer> list = new LinkedList<>();  
list.add(10);  
list.add(20);  
list.add(30);  
System.*out*.println("Traversing Elements With For Loop");  
for (int i = 0; i < list.size(); i++) {  
 System.*out*.println(list.get(i));  
}  
System.*out*.println("Traversing Elements With For Each Loop");  
for (Integer t : list) {  
 System.*out*.println(t);  
}

**Output:-**



**2:-** Write a Java program to create List containing list of items and use Iterator interface to print items present in the list.

**Description:**

This Java code snippet continues to traverse the elements of a LinkedList named list , but this time it uses an Iterator to iterate through the elements. Here's a description of the code:

1. **System.out.println("Traversing Elements With Iterator");** : This line prints a message to indicate that the program is about to traverse the list elements using an Iterator.

2. **Iterator<Integer> itr = list.iterator();** : This line initializes an Iterator of type Integer named itr by obtaining an iterator from the list . An iterator allows you to traverse a collection sequentially.

3. **System.out.println(itr.next());** : This line uses the next method of the Iterator to retrieve and print the first element in the list . It advances the iterator to the next element.

4. **System.out.println(itr.next());** : Similarly, this line retrieves and prints the second element in the list and advances the iterator.

5**. System.out.println(itr.next());** : This line retrieves and prints the third element in the list and advances the iterator.

6. **while (itr.hasNext())** { : This line starts a while loop that continues as long as the Iterator itr has more elements to iterate over. The hasNext method is used to check if there are more elements.

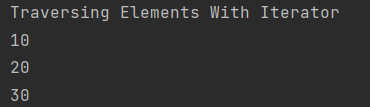
7. **Integer number = itr.next();** : Inside the loop, this line uses the next method to retrieve the next element in the list and assigns it to the number variable of type Integer .

8. **System.out.println(number);** : This line prints the number variable, which represents the current element being iterated over.

**Code:-**

System.*out*.println("Traversing Elements With Iterator");  
Iterator<Integer> itr = list.iterator();  
System.*out*.println(itr.next());  
System.*out*.println(itr.next());  
System.*out*.println(itr.next());  
while (itr.hasNext()) {  
 Integer number = itr.next();  
 System.*out*.println(number);  
}

**Output:-**



**3:-** Write a Java program to display all the String elements in a List in uppercase by using toUppercase() method.

**Description:**

This Java code snippet demonstrates how to create an ArrayList called list2 containing String elements and then iterate through the elements using an Iterator . During the iteration, it converts each String element to uppercase using the toUpperCase() method and prints the uppercase version of each element. Here's a description of the code:

1. **ArrayList<String> list2 = new ArrayList<>();** : This line creates an ArrayList named list2 to store String elements.

2. **list2.add("Rohit"); , list2.add("Yash"); , list2.add("Devank");** : These lines add three String elements ("Rohit," "Yash," and "Devank") to the list2 .

3. **System.out.println("Display All The String Elements In A List In Uppercase");** : This line prints a message to indicate that the program is about to display the String elements in uppercase.

4. **Iterator<String> itr2 = list2.iterator();** : This line initializes an Iterator of type String named itr2 to iterate through the elements in list2 . The iterator() method is called on list2 to obtain the Iterator.

5. **while (itr2.hasNext()) {** : This line starts a while loop that continues as long as the Iterator itr2 has more elements to iterate over. The hasNext() method is used to check if there are more elements.

6. **String s = itr2.next().toUpperCase();** : Inside the loop, this line retrieves the next String element from the list2 using the next() method and immediately converts it to uppercase using the toUpperCase() method. The result is stored in the String variable s .

7**. System.out.println(s);** : This line prints the String variable s , which contains the uppercase version of the current element being iterated over.

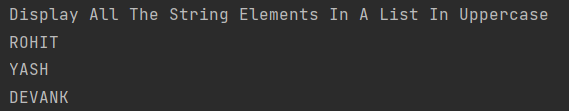
The code effectively demonstrates how to iterate through a collection of String elements and manipulate each element during iteration by converting it to uppercase

**Code:-**

ArrayList<String> list2 = new ArrayList<>();  
list2.add("Rohit");  
list2.add("Yash");  
list2.add("Devank");  
System.*out*.println("Display All The String Elements In A List In Uppercase");  
Iterator<String> itr2 = list2.iterator();

while (itr2.hasNext()) {  
 String s = itr2.next().toUpperCase();  
 System.*out*.println(s);  
}

**Output:-**



4:- Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backword direction.

**Description:**

This Java code snippet demonstrates how to create an ArrayList named list3 containing Integer elements and then traverse the elements using a ListIterator . The code performs both forward and backward iterations using the ListIterator . Here's a description of the code:

1. **List<Integer> list3 = new ArrayList<>();** : This line creates an ArrayList named list3 to store Integer elements.

2. **list3.add(80); , list3.add(90); , list3.add(60);** : These lines add three Integer elements (80, 90, and 60) to the list3 .

3**. System.out.println("Travesing String Using Iterator"); :** This line prints a message to indicate that the program is about to traverse the Integer elements using an Iterator.

4. **ListIterator<Integer> itr3 = list3.listIterator();** : This line initializes a ListIterator of type Integer named itr3 to iterate through the elements in list3 . The listIterator() method is called on list3 to obtain the ListIterator.

5. **System.out.println("Forward Direction:");** : This line prints a message to indicate that the program is about to traverse the Integer elements in the forward direction.

6. The while loop with **itr3.hasNext()** :

* This loop iterates through the elements in the forward direction as long as the ListIterator itr3 has more elements to traverse (determined using hasNext() ).
* Integer p = itr3.next(); retrieves the next Integer element from list3 using the next() method and assigns it to the Integer variable p .
* System.out.println(p); prints the Integer element p to the console.

7. **System.out.println("Backward Direction:");** : This line prints a message to indicate that the program is about to traverse the Integer elements in the backward direction.

8. The while loop with itr3.hasPrevious() :

* This loop iterates through the elements in the backward direction as long as the ListIterator itr3 has previous elements to traverse (determined using hasPrevious() ).
* Integer r = itr3.previous(); retrieves the previous Integer element from list3 using the previous() method and assigns it to the Integer variable r .
* System.out.println(r); prints the Integer element r to the console.

The code demonstrates how to use a ListIterator to traverse a collection of Integer elements in both forward and backward directions. It is particularly useful when you need to iterate through a list bidirectionally and perform operations on the elements.

**Code:-**

List<Integer> list3 = new ArrayList<>();  
list3.add(80);  
list3.add(90);  
list3.add(60);  
System.*out*.println("Travesing String Using Iterator");  
ListIterator<Integer> itr3 = list3.listIterator();

// Iterate through the list of lists using ListIterator  
 System.*out*.println("Forward Direction:");  
 while (itr3.hasNext()) {  
 Integer p = itr3.next();  
 System.*out*.println(p);  
 }  
 // Print the list in reverse/backward direction  
 System.*out*.println("Backward Direction:");  
 while(itr3.hasPrevious()) {  
 Integer r=itr3.previous();  
 System.*out*.println(r);  
}

**Output:-**

