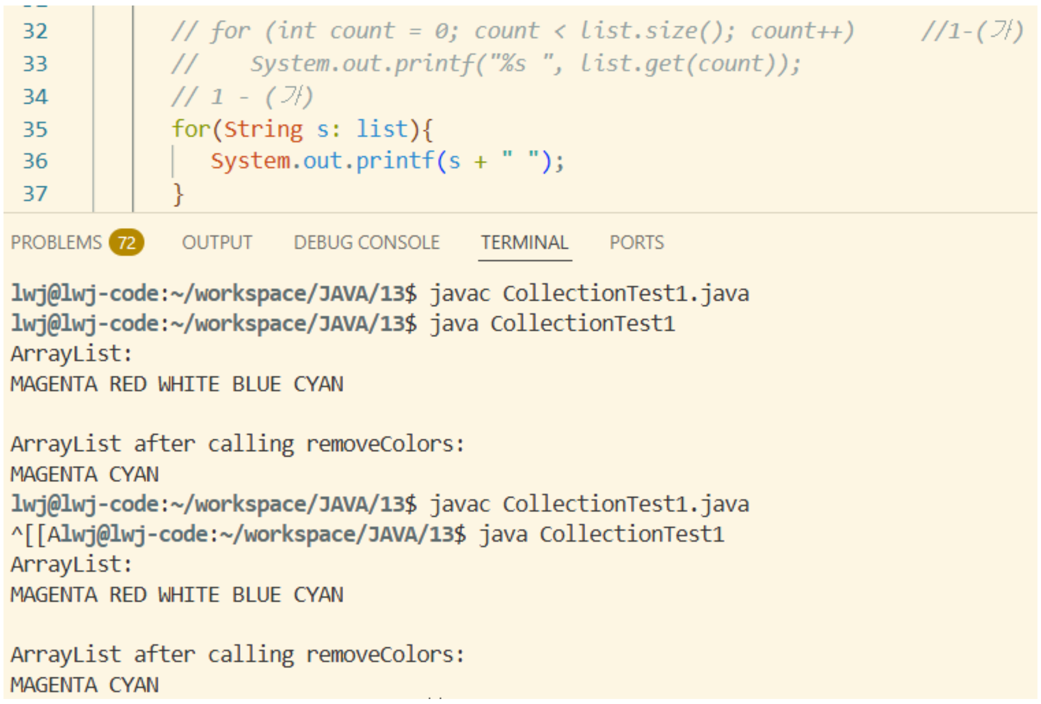
13주차 실습 보고서

강의명	객체지향프로그래밍및실습
담당교수	류기열
학과 학년	소프트웨어학과 2학년
학번	202220209
작성자	이욱준

2025.11.27.

□ 1

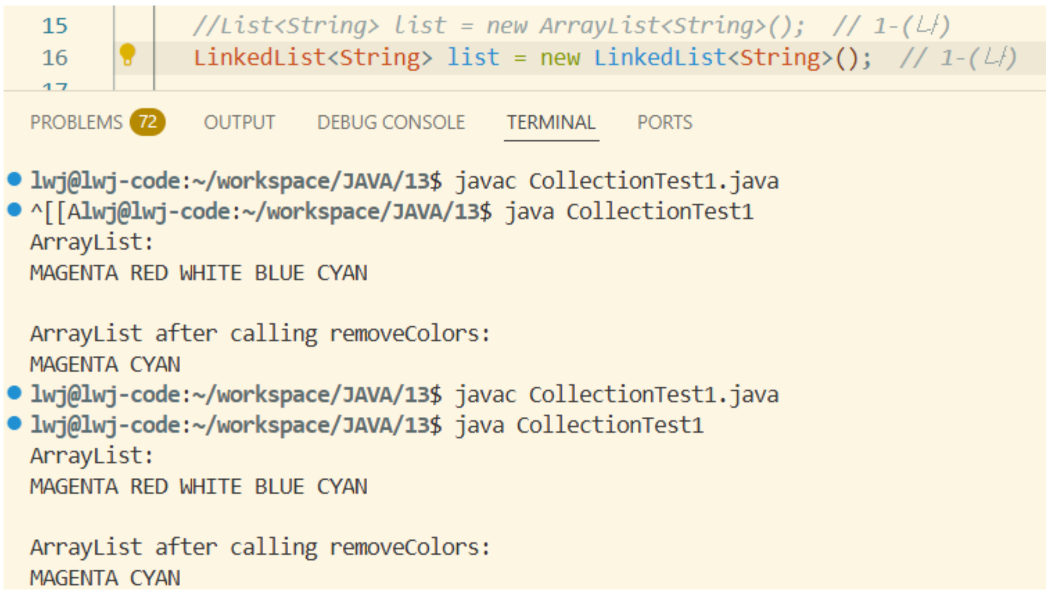
(가) Iterating



```
// for (int count = 0; count < list.size(); count++)     //1-(가)
//     System.out.printf("%s ", list.get(count));
// 1 - (가)
for(String s: list){
    System.out.printf(s + " ");
}
```

```
lwj@lwj-code:~/workspace/JAVA/13$ javac CollectionTest1.java
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
ArrayList:
MAGENTA RED WHITE BLUE CYAN

ArrayList after calling removeColors:
MAGENTA CYAN
lwj@lwj-code:~/workspace/JAVA/13$ javac CollectionTest1.java
^[[Alwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
ArrayList:
MAGENTA RED WHITE BLUE CYAN

ArrayList after calling removeColors:
MAGENTA CYAN
```

(나) LinkedList



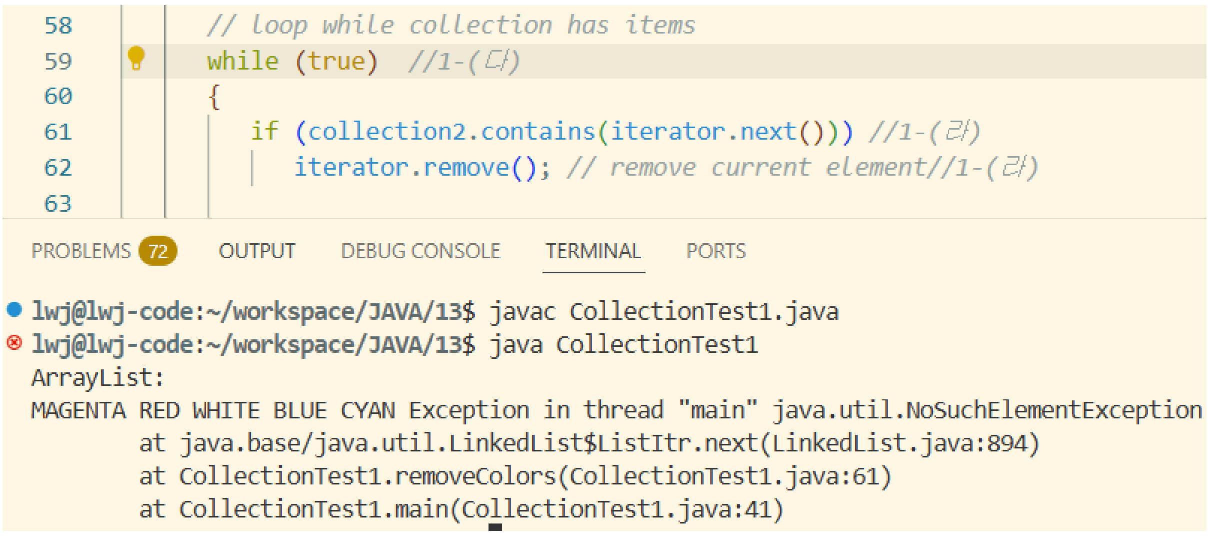
```
//List<String> list = new ArrayList<String>();  // 1-(나)
LinkedList<String> list = new LinkedList<String>();  // 1-(나)
```

```
● lwj@lwj-code:~/workspace/JAVA/13$ javac CollectionTest1.java
● ^[[Alwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
  ArrayList:
  MAGENTA RED WHITE BLUE CYAN

  ArrayList after calling removeColors:
  MAGENTA CYAN
● lwj@lwj-code:~/workspace/JAVA/13$ javac CollectionTest1.java
● lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
  ArrayList:
  MAGENTA RED WHITE BLUE CYAN

  ArrayList after calling removeColors:
  MAGENTA CYAN
```

결과는 List와 LinkedList로 구현한 방법 둘 모두 같다. 하지만 removeColors() 메소드에서 list의 원소를 삭제하는 과정에서 List.remove()는 해당하는 원소를 삭제한 후 해당 원소보다 index가 뒤에 위치한 원소들을 앞으로 당겨오는 과정에서 메모리를 모두 탐색하

고 수정하는 만큼의 시간복잡도를 필요로 한다. 반면에 LinkedList의 경우에는 삭제를 하더라도 연결점에서 가리키는 주소값을 바꿔주기만 하면 되기 때문에 List에 비해 더 적은 시간을 필요로 한다.

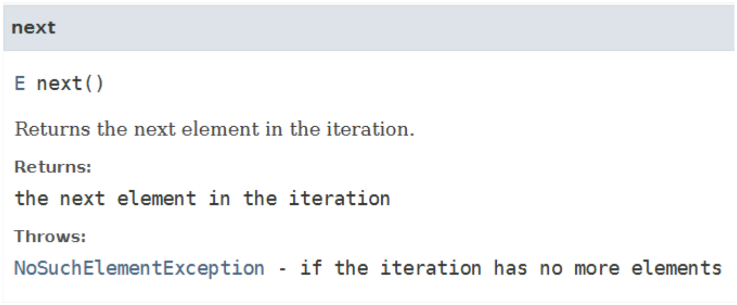
(다)



```
58    // loop while collection has items
59    while (true)  //1-(다)
60    {
61       if (collection2.contains(iterator.next())) //1-(라)
62          iterator.remove(); // remove current element//1-(라)
63
```

```
lwj@lwj-code:~/workspace/JAVA/13$ javac CollectionTest1.java
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
ArrayList:
MAGENTA RED WHITE BLUE CYAN Exception in thread "main" java.util.NoSuchElementException
        at java.base/java.util.LinkedList$ListItr.next(LinkedList.java:894)
        at CollectionTest1.removeColors(CollectionTest1.java:61)
        at CollectionTest1.main(CollectionTest1.java:41)
```

NoSuchElementException이 발생한다. list를 받고 있는 iterator에서 while(true)일 경우 마지막 반복에서 호출되는 iterator.next()의 경우 해당하는 원소가 없기 때문이다. 실제로 Oracle 공식 문서에서 iteration 중 element가 더 없는 경우에 next()를 호출할 경우 NoSuchElementException을 throw한다고 나와있다.



next

`E next()`

Returns the next element in the iteration.

Returns:

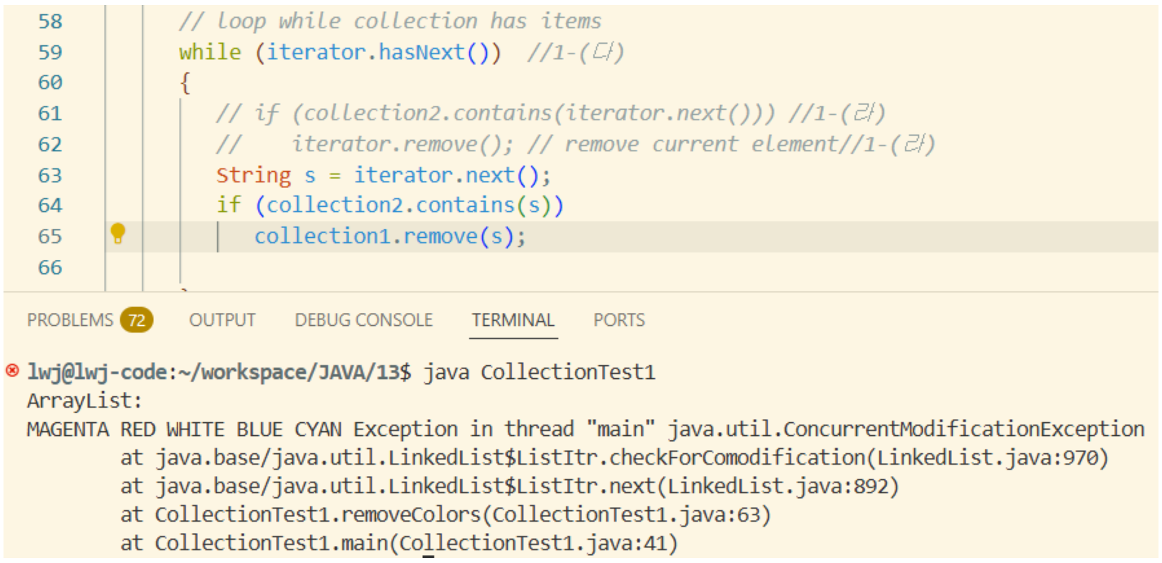
`the next element in the iteration`

Throws:

`NoSuchElementException - if the iteration has no more elements`

https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html

(라)



```
        // loop while collection has items
        while (iterator.hasNext())  //1-(다)
        {
            // if (collection2.contains(iterator.next())) //1-(라)
            //    iterator.remove(); // remove current element//1-(라)
            String s = iterator.next();
            if (collection2.contains(s))
                collection1.remove(s);
```

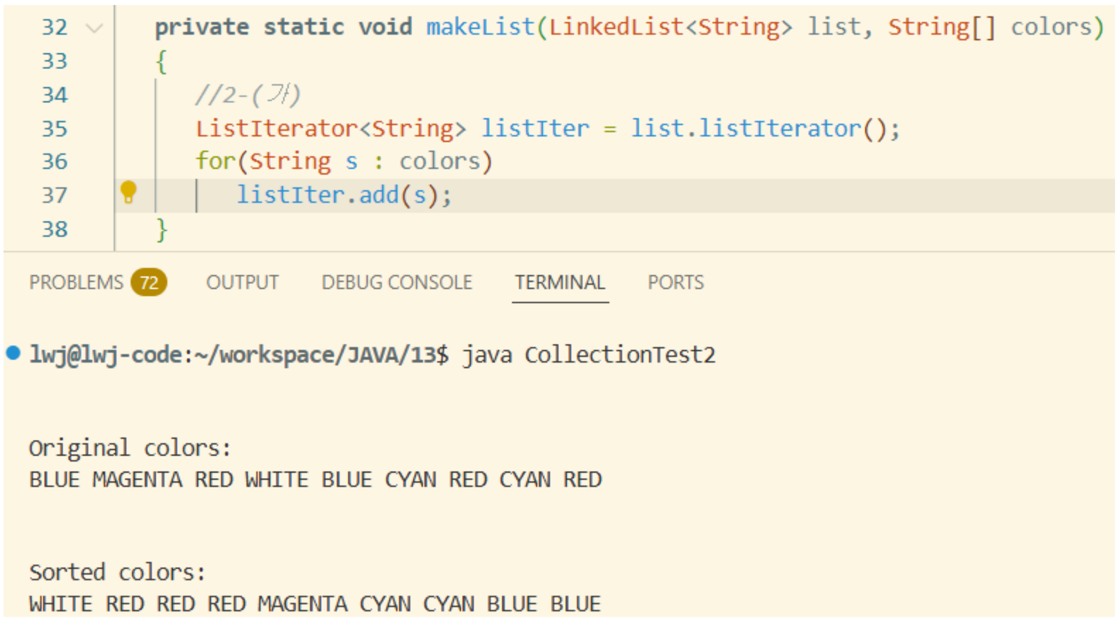
PROBLEMS 72   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest1
ArrayList:
MAGENTA RED WHITE BLUE CYAN Exception in thread "main" java.util.ConcurrentModificationException
        at java.base/java.util.LinkedList$ListItr.checkForComodification(LinkedList.java:970)
        at java.base/java.util.LinkedList$ListItr.next(LinkedList.java:892)
        at CollectionTest1.removeColors(CollectionTest1.java:63)
        at CollectionTest1.main(CollectionTest1.java:41)
```

ConcurrentModificationException, Collection1에 대한 iterator를 이미 만들어서 사용하고 있는 상황에서 collector1에 대한 직접적인 수정이 가해질 경우 List 내부의 modCount에 대한 변형이 생기고 LinkedList의 경우 연결이 끊어질 수 있기 때문에 이를 막기 위해 Java에서 자체적으로 하나의 Collection에 대해 2개의 서로 다른 iterator가 modification을 한다면 해당 에러를 throw한다.

☐ 2

(가)

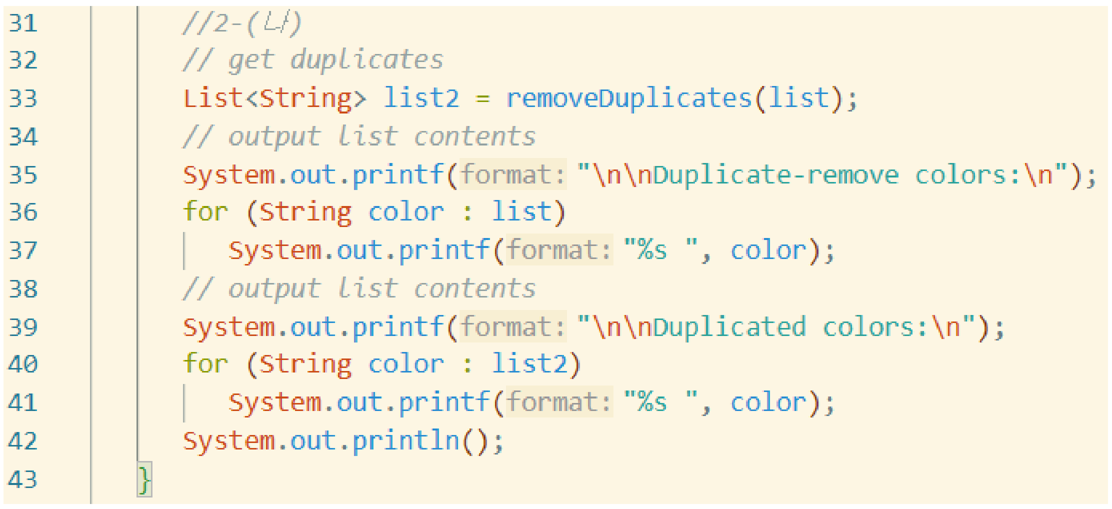


```
    private static void makeList(LinkedList<String> list, String[] colors)
    {
        //2-(가)
        ListIterator<String> listIter = list.listIterator();
        for(String s : colors)
            listIter.add(s);
    }
```

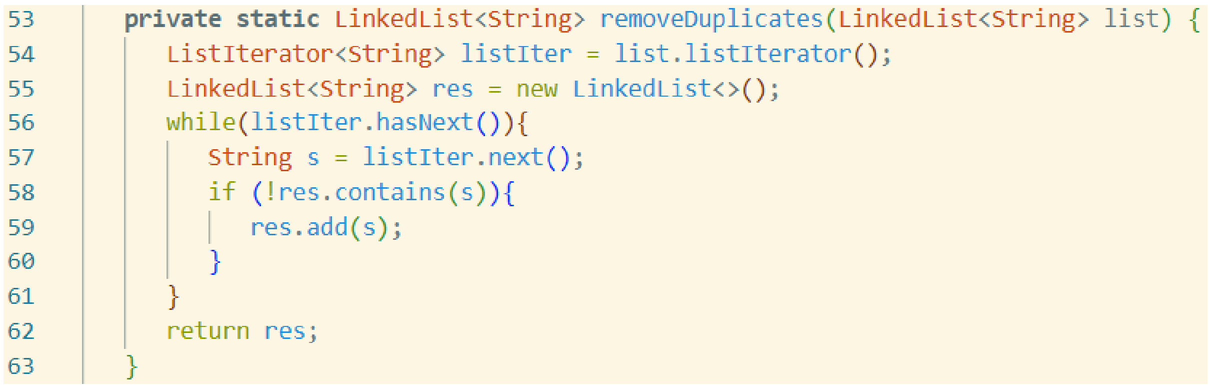
PROBLEMS 72   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest2


Original colors:
BLUE MAGENTA RED WHITE BLUE CYAN RED CYAN RED


Sorted colors:
WHITE RED RED RED MAGENTA CYAN CYAN BLUE BLUE
```

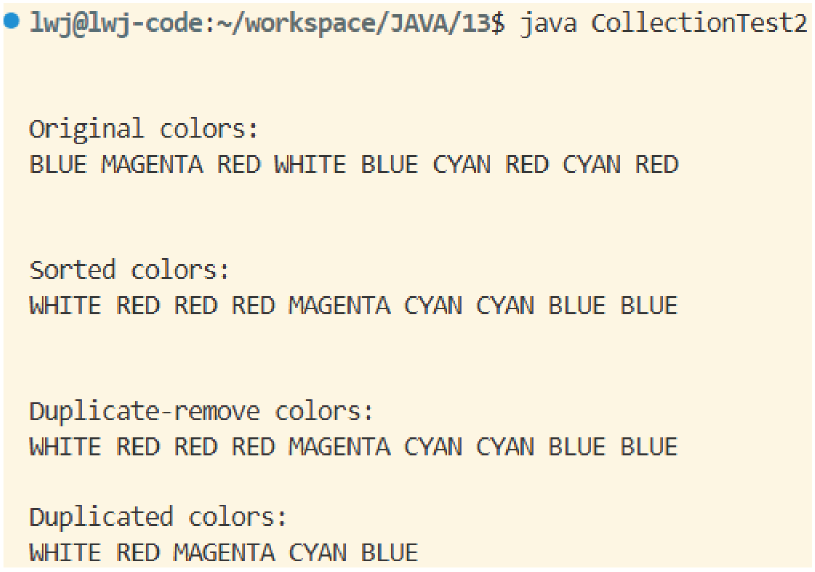
(나)



```java
        //2-(나)
        // get duplicates
        List<String> list2 = removeDuplicates(list);
        // output list contents
        System.out.printf(format: "\n\nDuplicate-remove colors:\n");
        for (String color : list)
            System.out.printf(format: "%s ", color);
        // output list contents
        System.out.printf(format: "\n\nDuplicated colors:\n");
        for (String color : list2)
            System.out.printf(format: "%s ", color);
        System.out.println();
    }
```

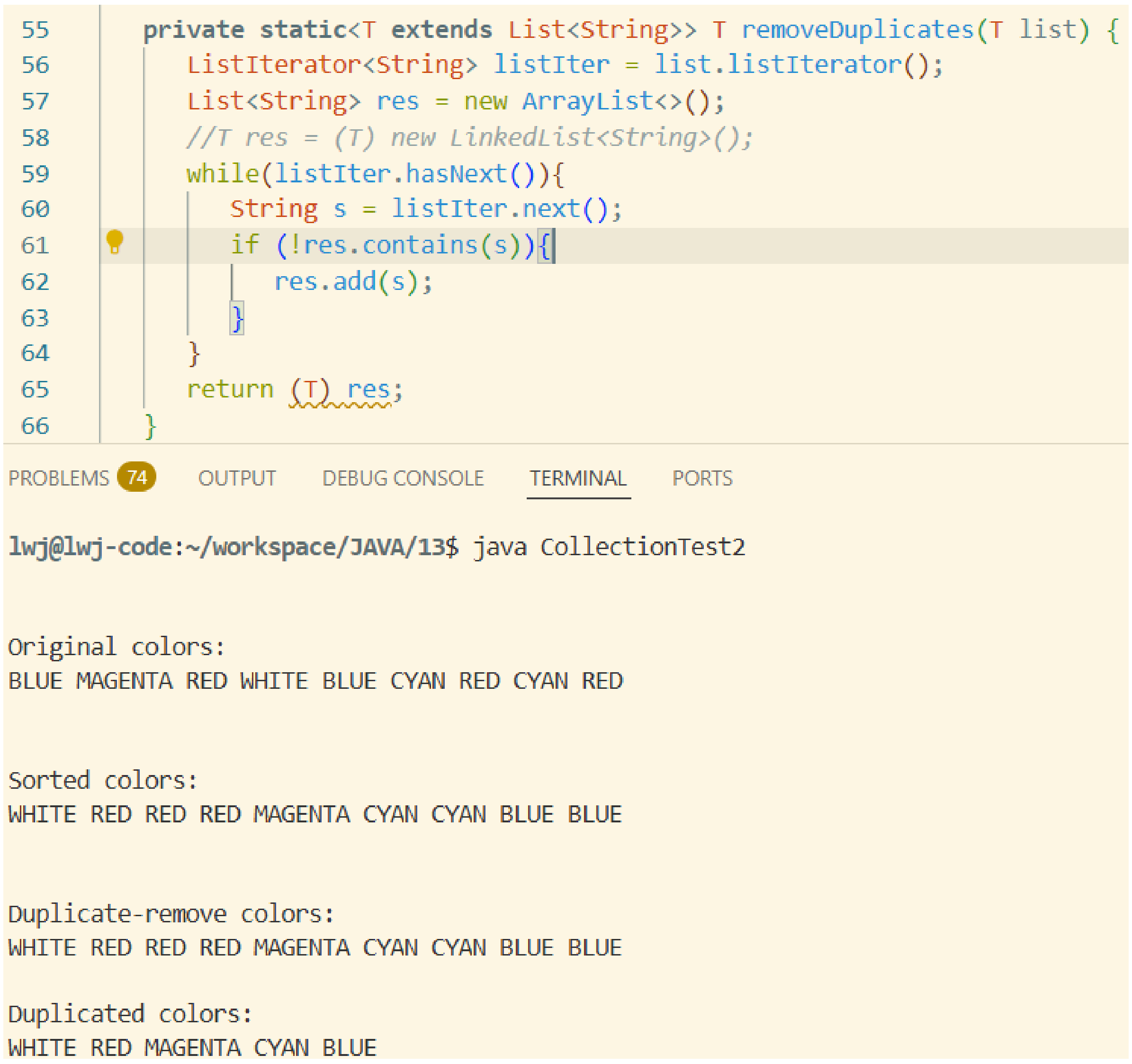


```java
    private static LinkedList<String> removeDuplicates(LinkedList<String> list) {
        ListIterator<String> listIter = list.listIterator();
        LinkedList<String> res = new LinkedList<>();
        while(listIter.hasNext()){
            String s = listIter.next();
            if (!res.contains(s)){
                res.add(s);
            }
        }
        return res;
    }
```



```
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest2


Original colors:
BLUE MAGENTA RED WHITE BLUE CYAN RED CYAN RED


Sorted colors:
WHITE RED RED RED MAGENTA CYAN CYAN BLUE BLUE


Duplicate-remove colors:
WHITE RED RED RED MAGENTA CYAN CYAN BLUE BLUE

Duplicated colors:
WHITE RED MAGENTA CYAN BLUE
```

(다)



```java
    private static<T extends List<String>> T removeDuplicates(T list) {
        ListIterator<String> listIter = list.listIterator();
        List<String> res = new ArrayList<>();
        //T res = (T) new LinkedList<String>();
        while(listIter.hasNext()){
            String s = listIter.next();
            if (!res.contains(s)){
                res.add(s);
            }
        }
        return (T) res;
    }
```

```
lwj@lwj-code:~/workspace/JAVA/13$ java CollectionTest2


Original colors:
BLUE MAGENTA RED WHITE BLUE CYAN RED CYAN RED


Sorted colors:
WHITE RED RED RED MAGENTA CYAN CYAN BLUE BLUE


Duplicate-remove colors:
WHITE RED RED RED MAGENTA CYAN CYAN BLUE BLUE

Duplicated colors:
WHITE RED MAGENTA CYAN BLUE
```

□ 코드 전문

CollectionTest1.java

```java
// Fig. 16.2: CollectionTest.java
// Collection interface demonstrated via an ArrayList object.
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class CollectionTest1
{
   public static void main(String[] args)
   {
      // add elements in colors array to list
      String[] colors = {"MAGENTA", "RED", "WHITE", "BLUE", "CYAN"};
      //List<String> list = new ArrayList<String>();  // 1-(나)
      LinkedList<String> list = new LinkedList<String>();  // 1-(나)

      for (String color : colors)
         list.add(color); // adds color to end of list

      // add elements in removeColors array to removeList
      String[] removeColors = {"RED", "WHITE", "BLUE"};
      List<String> removeList = new ArrayList<String>(); // 1-(나)


      for (String color : removeColors)
         removeList.add(color);

      // output list contents
      System.out.println("ArrayList: ");


      // for (int count = 0; count < list.size(); count++)     //1-(가)
      //    System.out.printf("%s ", list.get(count));
      // 1 - (가)
      for(String s: list){
         System.out.printf(s + " ");
      }

      // remove from list the colors contained in removeList
      removeColors(list, removeList);

      // output list contents
      System.out.printf("%n%nArrayList after calling removeColors:%n");

      for (String color : list)
         System.out.printf("%s ", color);
      System.out.println();
   }

   // remove colors specified in collection2 from collection1
   private static void removeColors(Collection<String> collection1,
      Collection<String> collection2)
   {
      // get iterator
      Iterator<String> iterator = collection1.iterator();

      // loop while collection has items
      while (iterator.hasNext())  //1-(다)
      {
         // if (collection2.contains(iterator.next())) //1-(라)
         //    iterator.remove(); // remove current element//1-(라)
         String s = iterator.next();
         if (collection2.contains(s))
            collection1.remove(s);

      }
   }
} // end class CollectionTest
```

CollectionTest2.java

```java
// Fig. 16.2: CollectionTest.java
// Collection interface demonstrated via an ArrayList object.
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
public class CollectionTest2
{
      public static void main(String[] args)
   {
     // add elements in colors array to list
     String[] colors = {"BLUE", "MAGENTA", "RED", "WHITE", "BLUE", "CYAN",
"RED", "CYAN", "RED" };
```

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

     makeList(list, colors);

     // output list contents
     System.out.printf("%n%nOriginal colors:%n");
      for (String color : list)
        System.out.printf("%s ", color);
      System.out.println();

      list.sort(Comparator.reverseOrder());

      // output list contents
      System.out.printf("%n%nSorted colors:%n");
      for (String color : list)
         System.out.printf("%s ", color);
      System.out.println();

      //2-(나)
      // get duplicates
      List<String> list2 = removeDuplicates(list);
      // output list contents
      System.out.printf("\n\nDuplicate-remove colors:\n");
      for (String color : list)
         System.out.printf("%s ", color);
      // output list contents
      System.out.printf("\n\nDuplicated colors:\n");
      for (String color : list2)
         System.out.printf("%s ", color);
      System.out.println();
   }

   private static void makeList(LinkedList<String> list, String[] colors)
   {
      //2-(가)
      ListIterator<String> listIter = list.listIterator();
      for(String s : colors)
         listIter.add(s);
   }

   private static<T extends List<String>> T removeDuplicates(T list) {
      ListIterator<String> listIter = list.listIterator();
      List<String> res = new ArrayList<>();
      //T res = (T) new LinkedList<String>();
      while(listIter.hasNext()){
         String s = listIter.next();
         if (!res.contains(s)){
            res.add(s);
            }
        }
        return (T) res;
    }
```