Java ArrayList and Iterator

1. Array

Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms. Following are the important terms to understand the concept of Array.

- ✓ Element: Each item stored in an array is called an element.
- ✓ Index: Each location of element in an array has a numerical index, which is used to identify the element.

1. 1 Array in Python

Array in Python is created by importing array module to the python program. The array is declared as shown below.

```
arrayName = array(typecode, [Initializers])
```

"typecode" is a character that designates the type of data that will be stored in the array. For example, 'i' designates and array of signed integers.

1. 2 Array in Java

Array in Java is used to store a fixed-size sequential collection of elements of the same type. It must be specified the type that the variable can reference.

The syntax for declaring an array variable is

```
dateType[] arrayName = new datatype[n];
or
datatype arrayName[] = new datatype[n];
```

2. ArrayList

An ArrayList is a re-sizable array, also called a dynamic array. It grows its size to accommodate new elements and shrinks the size when the elements are removed. ArrayList internally uses an array to store the elements. Just like arrays, it allows you to retrieve the elements by their index. Java ArrayList allows duplicate and null values. It is an ordered collection. It maintains the insertion order of the elements.

The syntax for declaring an ArrayList is as follows.

```
ArrayList<E> variable_name = new ArrayList<E>();
```

ArrayList should be declared with generic <E>, which forces to have only specified type of objects in ArrayList. E should be class name. For example, “Integer” instead of “int”.

An example is as follows.

```
ArrayList<String> data = new ArrayList<String>();
```

Figure 1 shows an example program of ArrayList, and Figure 2 shows the output of the example program.

```
/* An example of ArrayList */

import java.util.ArrayList;

public class Test1 {
    ArrayList<String> colors = new ArrayList<String>();

    public static void main(String[] args) {
        Test1 t1 = new Test1();
        t1.example();
    }

    public void example() {
        colors.add("red");                  // add red
        colors.add("green");            // add green
        System.out.println(colors);         // print out colors
        colors.add(1, "yellow");        // insert yellow at position 1
        System.out.println(colors);         // print out colors
        // print out the word at position 0
        System.out.println("get(0): " + colors.get(0));
        // replace the element at position 1 with blue
        colors.set(1, "blue");
        System.out.println(colors);         // print out colors
        // print out the size of colors
        System.out.println("size is " + colors.size());
        // print out each element on each line
        for(int i = 0; i < colors.size(); i++) {
            System.out.println(colors.get(i));
        }
    }
}
```

Figure 1. An Example Program of ArrayList.

```
[red, green]
[red, yellow, green]
get(0): red
[red, blue, green]
size is 3
red
blue
green
```

Figure 2. The Output of The Example Program of ArrayList.

Table 1. The main methods of ArrayList.

Modifier/Type	Method and Description
boolean	add(E element) Appends the specified element to the end of this list
void	add(int index, E element) Inserts the specified element at the specified position
void	clear() Removes all of the elements from this list
boolean	contains(Object o) Returns true if this list contains the specified element
E	get(int index) Returns the element at the specified position
int	indexOf(Object o) Returns the index of the first occurrence of the specified elements, or -1 if this list does not contain the element
boolean	isEmpty() Returns true if this list contains no elements
Iterator<E>	iterator() Returns an iterator over the elements in this list in proper sequence
E	remove(int index) Removes the element at the specified position
boolean	remove(Object o) Removes the first occurrence of the specified element from this list, if it is present
E	set(int index, E element) Replaces the element at the specified position
int	size() Returns the number of elements in this list

3. Iterator

Iterator is used for iterating (looping) various collection classes such as ArrayList. It is used to obtain or remove elements. It obtained by calling the colletion's iterator() method. For example,

```
ArrayList<String> data = new ArrayList<Sring>();
Iterator<String> iterator = data.iterator();
```

Table 2 shows the main method of Iterator.

Table 2. Thein method of Iterator.

Modifier/Type	Method and Description
boolean	hasNext() Returns true if the iteration has more elements
E	next() Returns the next element in the iteration
void	remove() Removes from the underlying collection the last element returned by this iterator

3. 1 Program 1 using Iterator

Figure 3 shows a program that removes “blue” from ArrayList containing two adjacent values of “blue”. This program uses “Iterator”. Figure 4 shows the result of erasing the two blues.

```
import java.util.ArrayList;
import java.util.Iterator;

public class Test2 {
    String[] original = {"red", "blue", "blue", "green", "yellow", "blue"};
    ArrayList<String> colors = new ArrayList<String>(); // global variable
    public static void main(String[] args) {
        Test2 t2 = new Test2();
        t2.example();
    }
    public Test2() { // constructor
        for(int i = 0; i < original.length; i++) {
            colors.add(original[i]);
        }
    }

    public void example(){
        System.out.println(colors);
        Iterator<String> iterator = colors.iterator();
        while(iterator.hasNext()) {
            if(iterator.next().equals("blue")) {
                iterator.remove();
            }
        }
        System.out.println(colors);
    }
}
```

```
[red, blue, blue, green, yellow, blue]
[red, green, yellow]
```

Figure 4. The Result Program 1 using Iterator.

3. 2 Program 2 without using Iterator

Figure 5 shows the method of erasing using the location where “blue” is found without using Iterator. Figure 6 shows the result of program 2. The first “blue” is erased, but the second “blue” is not erased because the position of the element shifts.

```
public void example(){
    System.out.println(colors);

    for(int i = 0; i < colors.size(); i++){
        if(colors.get(i).equals("blue")) {
            colors.remove(i);
        }
    }
    System.out.println(colors);
}
```

Figure 5. Program 2 using Iterator.

```
[red, blue, blue, green, yellow, blue]
[red, blue, green, yellow]
```

Figure 6. The Result Program 2 without using Iterator.

3. 3 Program 3 without using Iterator

Figure 7 shows the method of erasing using the location where “blue” is found without using Iterator. Figure 8 shows the result of program 2. This program erases “blue” as long as “blue” exists, so all “blue” are erased.

```
public void example(){
    System.out.println(colors);

    while(colors.contains("blue")) {
        colors.remove("blue");
    }
    System.out.println(colors);
}
```

Figure 7. Program 3 using Iterator.

```
[red, blue, blue, green, yellow, blue]
[red, green, yellow]
```

Figure 8. The Result Program 3 without using Iterator.