

CSC508 Data Structures

Topic 2 : Array

Recap

- ▶ Data types
- ▶ Definition of Data Structures
- ▶ Java Collection Framework
- ▶ Review
 - ▶ Java Abstract Classes
 - ▶ Java Interfaces

Topic Structure

- ▶ Collection hierarchy
- ▶ Array definition
- ▶ Array implementation
 - ▶ ArrayList
 - ▶ User-defined

Learning Outcomes

- ▶ At the end of this lesson, students should be able to:
 - ▶ Define the concept array
 - ▶ Describe array characteristics
 - ▶ Implement array operation

Collection Hierarchy

- ▶ The Collection interface specifies a subset of the methods specified in the List interface. Definition of Data Structures
- ▶ Collection interface is the root of the collection hierarchy
 - ▶ Two branches: one rooted by the List interface and the other by the Set interface

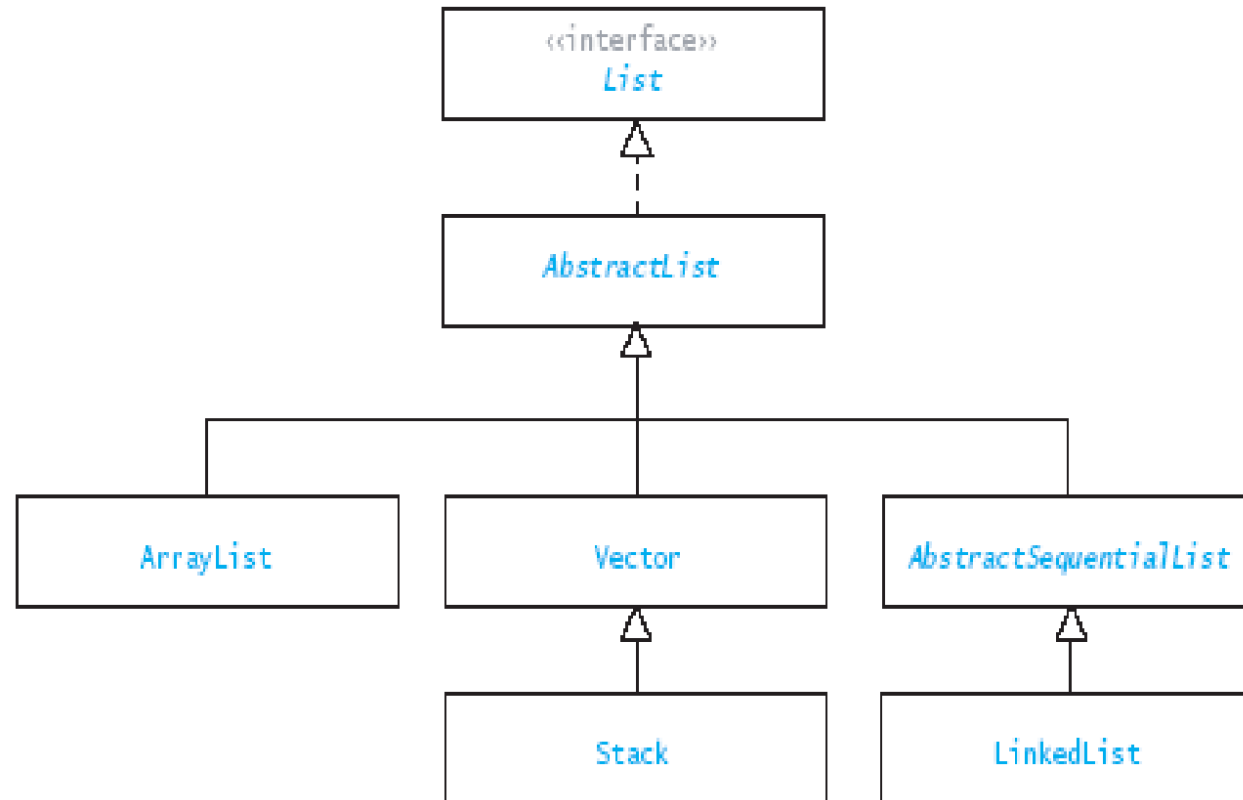
List Interface

- ▶ List: A collection of elements of the same type.
 - ▶ Example: □ A list of students □ A list of books
- ▶ Allowed operations on the List interface include:
 - ▶ Finding a specified target
 - ▶ Adding an element to either end
 - ▶ Removing an item from either end
 - ▶ Traversing the list structure without a subscript

List Interface (cont.)

FIGURE 4.1

The java.util.List
Interface and Its
Implementers



Array

- ▶ A collection of a fixed number of components wherein all of the components have the same data type.
- ▶ Content of an array is referred using index, or subscript, ranging from 0 (the first element in the array) until $n-1$ (the last element in the array)
 - ▶ Where n is the size of an array.

1	3	5	7	9	11	13	15	17	19
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Figure 1.1 Array 1

A	L	G	O	R	I	T	H	M	S
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Figure 1.2 Array 2

Array Characteristic

- ▶ **Homogenous**
 - ▶ Array only stores data of the same type.
- ▶ **Fixed-size**
 - ▶ In general, the size of array is fixed upon creation.
- ▶ **Random access**
 - ▶ Element of an array can access in random

Operation on Arrays

- ▶ Create and initialize
- ▶ Determine whether an array is full or empty
- ▶ Insert an element in the list at the specified location
- ▶ Remove an element from the list at the specified location
- ▶ Retrieve an element from the list at the specified location
- ▶ Replace an item at the specified location with another item
- ▶ Search the array for a given element

ArrayList Class

- ▶ The Java API provides the ArrayList class, which uses an array as the underlying structure to implement the List.
 - ▶ An ArrayList class is the class that implements the List interface.
- ▶ An ArrayList object is an improved version of a one-dimensional array, where;
 - ▶ It can select its elements in arbitrary order (random) using a subscript value.
 - ▶ The size automatically increases or decreases as new elements are added or removed.

Methods in ArrayList

TABLE 4.1

Methods of Class `java.util.ArrayList`

Method	Behavior
<code>public Object get(int index)</code>	Returns a reference to the element at position <code>index</code> .
<code>public Object set(int index, Object anEntry)</code>	Sets the element at position <code>index</code> to reference <code>anEntry</code> . Returns the previous value.
<code>public int size()</code>	Gets the current size of the <code>ArrayList</code> .
<code>public boolean add(Object anEntry)</code>	Adds a reference to <code>anEntry</code> at the end of the <code>ArrayList</code> . Always returns <code>true</code> .
<code>public void add(int index, Object anEntry)</code>	Adds a reference to <code>anEntry</code> , inserting it before the item at position <code>index</code> .
<code>int indexOf(Object target)</code>	Searches for <code>target</code> and returns the position of the first occurrence, or <code>-1</code> if it is not in the <code>ArrayList</code> .
<code>public Object remove(int index)</code>	Returns and removes the item at position <code>index</code> and shifts the items that follow it to fill the vacated space.

Sample Implementation ArrayList

```
import java.util.ArrayList; //Import ArrayList from java.util

public class Topic2Array {
    public static void main(String[] args) {
        //Create an ArrayList object
        ArrayList<String> customer = new ArrayList<String>();

        //Adding three elements into the array
        customer.add("John");
        customer.add("Mike");
        customer.add("Sue");
        customer.add("Joanna");

        //Print the array content
        for(int i = 0; i < customer.size(); i++) {
            System.out.println("Index " + i + " : " + customer.get(i));
        }
    }
}
```

```
Index 0 : John
Index 1 : Mike
Index 2 : Sue
Index 3 : Joanna
```

Sample Implementation ArrayList (cont.)

```
customer.remove(2); //Remove element at index 2
```

```
//Print the array content
```

```
for(int i = 0; i < customer.size(); i++) {  
    System.out.println("Index " + i + " : " + customer.get(i));  
}
```

```
customer.set(1, "Hendry"); //Update the content at index 1
```

```
//Print the array content
```

```
for(int i = 0; i < customer.size(); i++) {  
    System.out.println("Index " + i + " : " + customer.get(i));  
}
```

```
}
```

```
}
```

Index 0 : John
Index 1 : Mike
Index 2 : Joanna

Index 0 : John
Index 1 : Hendry
Index 2 : Joanna

User-defined array

- Create your own class myArray.

```
class myArray{  
    private int size;  
    public int[] newArray;  
  
    public myArray() {  
        newArray = new int[10]; //Create array of 10 elements  
        size = 0;  
        System.out.println("Array Created.");  
    }  
}
```

Define the variable to update the array size

Reference variable to the array

Constructor to initialize the array

User-defined array (cont.)

- Methods to add an element at the end of the array.

```
public void addElement(int a) {  
    newArray[size] = a;  
    size++;  
}
```

- Methods to remove an element from a specific index.

```
public void deleteElement(int index) {  
    if (size == 0 || index >= size)  
        System.err.println("Invalid index");  
    else {  
        for (int i = index; i < size; i++)  
            newArray[i] = newArray[i+1];  
        size --;  
    }  
}
```

Check the array size
first before remove

Shift the array element
one by one

User-defined array (cont.)

► Print array element

```
public void printArray() {  
    if (size == 0)  
        System.err.println("Array is Empty.");  
    else {  
        for (int i = 0; i < size; i++)  
            System.out.println(newArray[i]);  
    }  
}
```

► Get array size

```
public int getSize() {  
    return size;  
}
```

Testing myArray class

```
public static void main(String[] args) {  
    myArray arr1 = new myArray();  
    //arr1.printArray();  
    arr1.addElement(4);  
    arr1.addElement(6);  
    arr1.addElement(78);  
    arr1.printArray();  
    arr1.deleteElement(0);  
    arr1.printArray();  
    System.out.println(arr1.getSize());  
}
```

Array Created.

4

6

78

6

78

2

Summary

- ▶ Array is a collection of a fixed number of components wherein all of the components have the same data type.
- ▶ Build-in array implementation using ArrayList class.
- ▶ Self-defined array implementation
 - ▶ Create array
 - ▶ Insert element
 - ▶ Removing element
 - ▶ Print element

Next Topic...

- ▶ Linear list
 - ▶ Linked List
 - ▶ Concept
 - ▶ Implementation
 - ▶ Application

References

- ▶ Carrano, F. & Savitch, W. 2005. *Data Structures and Abstractions with Java, 2nd ed. Prentice-Hall.*
- ▶ Malik D.S, & Nair P.S., Data Structures Using Java, Thomson Course Technology, 2003.
- ▶ Rada Mihalcea, CSCE 3110 Data Structures and Algorithm Analysis notes, U of North Texas.