

AP Computer Science A

Java Programming Essentials [Ver.4.0]

Unit 4: Data Collections

CHAPTER 16: ARRAYLIST PROCESSING

DR. ERIC CHOU
IEEE SENIOR MEMBER



AP Computer Science Curriculum

- Implementing ArrayList Algorithms (T4.10)

Objectives:

- ArrayList Processing II: reverse of a list, sorting of a list, ListIterator
- Information List: Occurrence List, Available list, Non-recurring list, interval list, difference list (Generation of special lists)
- Abstract Data Types: (Basic Data Structures)
- Washington High School Project
- Bible word count sorted by occurrence project



ArrayList Processing II

Lecture 1

ArrayList Processing II

ArrayListProcessingII.java

1. Traversal of ArrayList (by index, object, object pointer)
2. Iterator and ListIterator
3. ArrayList of user-defined Class
4. Occurrence List (Char freq and bible.txt
WordCountArrayList.java)
5. Reverse of List
6. Sorting of Array by ArrayList

[1] How to traverse through ArrayList?

- (1) By index: (access by **index**)
 - `for (int index=0; i<arrayList.size(); i++)`
 - `System.out.println(arrayList.get(i));`
- (2) for-each loop: (access by **object**)
 - `for (String e: arrayList)`
 - `System.out.println(e);`
- (3) Iterator: (access by **object pointer**)
 - `Iterator<String> itr = arrayList.iterator();`
 - `while (itr.hasNext())`
 - `System.out.println(itr.next());`

[2] Index versus Iterator

(Primitive type pointer versus Object Type Pointer)

- You may have heard of me talking about input stream handler(Scanner), file handler(File), and XYZ handlers. Handler is a pointer to an object. It is an object itself. It is an **object-type pointer**.

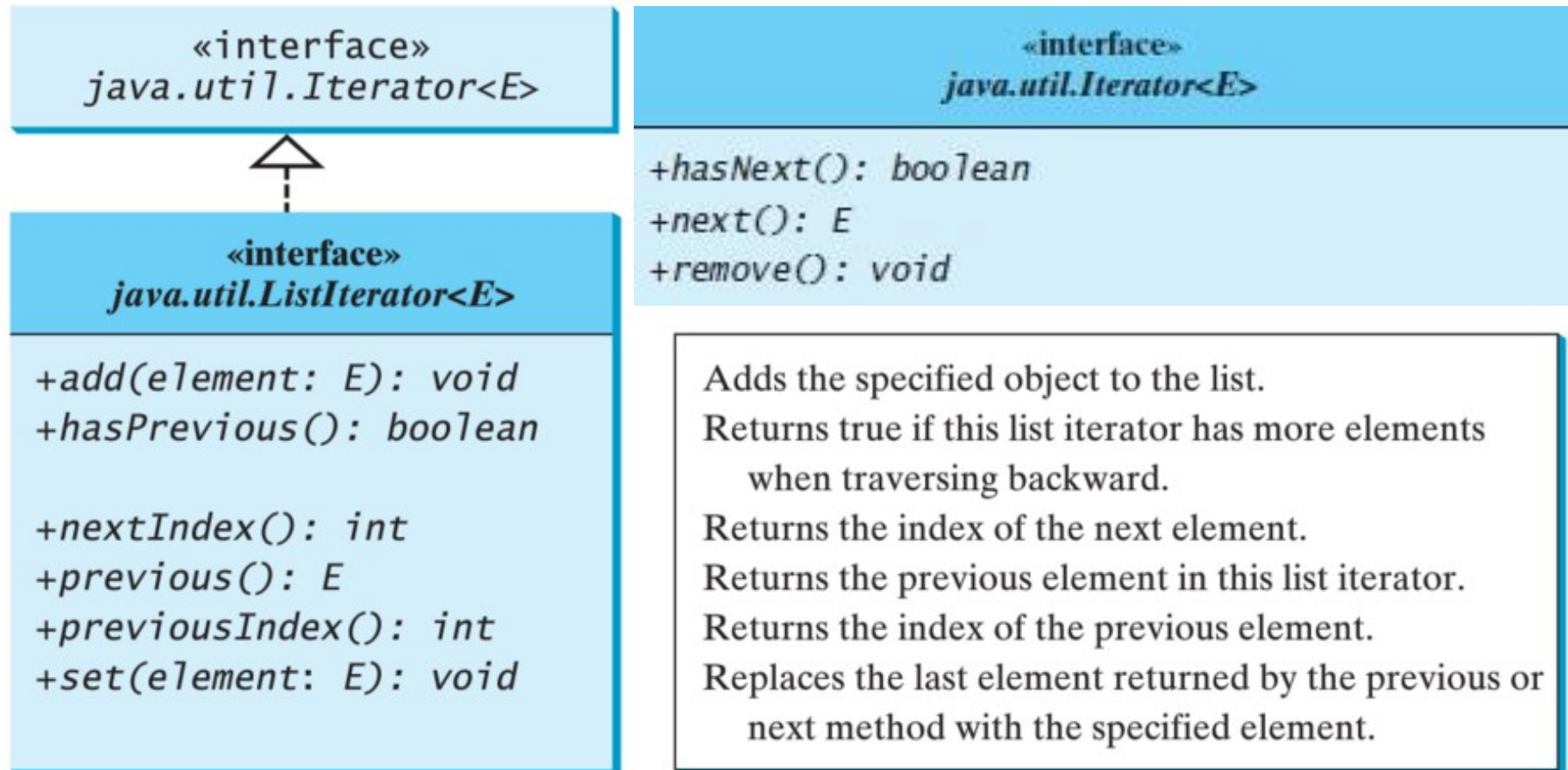
itr **boolean hasNext()**



Iterator of ArrayList (Iterator and ListIterator)

`Iterator<ElementType> itr = arraylist.iterator();`

`ListIterator<ElementType> itr = arraylist.iterator();`



ListIterator Versus Iterator

```
public static void iteratorExample() {
    System.out.println("ArrayList Iterator Examples:.....");
    ArrayList<String> al = new ArrayList<String>();
    al.add("C"); al.add("A"); al.add("E");
    al.add("B"); al.add("D"); al.add("F");
    System.out.print("Original contents of al: ");

    Iterator<String> itr = al.iterator();
    while (itr.hasNext()) {
        String element = itr.next();
        System.out.print(element + " ");
    }
    System.out.println();
}
```

ListIterator Versus Iterator

```
ListIterator<String> litr = al.listIterator();  
while (litr.hasNext()) {  
    String element = litr.next();  
    litr.set(element + "+");  
}  
  
// Now, display the list backwards.  
System.out.print("Modified list backwards: ");  
while (litr.hasPrevious()) {  
    String element = litr.previous();  
    System.out.print(element + " ");  
}  
}
```

ArrayList Iterator Examples:.....

Original contents of al: C A E B D F

Modified list backwards: F+ D+ B+ E+ A+ C+

[3] ArrayList of User-Defined Class

```
static class Student{  
    int rollno;  
    String name;  
    int age;  
    Student(int rollno,String name,int age) {  
        this.rollno=rollno;  
        this.name=name;  
        this.age=age;  
    }  
}
```

```
public static void userDefinedClass(){
    System.out.println("ArrayList of User-defined Class Examples:.....");
    //Creating user-defined class objects
    Student s1=new Student(101,"Sonoo",23);
    Student s2=new Student(102,"Ravi",21);
    Student s3=new Student(103,"Hanumat",25);
    ArrayList<Student> al=new ArrayList<Student>(); //creating arraylist
    al.add(s1); //adding Student class object
    al.add(s2);
    al.add(s3);
    Iterator itr=al.iterator();
    //traversing elements of ArrayList object
    while(itr.hasNext()){
        Student st=(Student)itr.next();
        System.out.println(st.rollno+" "+st.name+" "+st.age);
    }
}
```

ArrayList of User-defined Class Examples:.....

101 Sonoo 23

102 Ravi 21

103 Hanumat 25

[4] Character Occurrence Counting

```
char[] cc = {A,J,A,H,F,A,E,I,R,U,I,J,F,K,S,A,J,F,K,D,S,A,K,F,H,J,D,S,H,F,D,S,N,M,  
             A,N,R,W,E,M,H,F,J, H,F,J,K,L,D,S,A,H,F,J,H,D,S,A,J,K,H,F,J,K,D,S,  
             A,H,J,R,E,Y,W,O,H,F,D,S,J,F,L,K,A,J,F,K, J,D,S,L,K,A,F,J,S,A,U,O,  
             R,E,U,W,O,F,J,L,S,A,J,F,K,L,D,S,J,A,F,K,L,S,D,J,R,Q,P,U,R,I,Q,E,J,  
             T,J,Z,M,M,V,Z,C,X,N,V,A,D,J,K,A,S,J,F,S,P,R,E,W,Q,F,K,A,K,F,D,L,S};  
  
ArrayList<Character> cccc = new ArrayList<Character>(); // dictionary list  
  
ArrayList<Integer> freq = new ArrayList<Integer>(); // occurrence count list
```


[4] Character Occurrence Counting

```
for (char c: cc){
    boolean found = false;
    for (int i=0; i<cccc.size(); i++){
        // found c in the cccc (dictionary)
        if (cccc.contains(new Character(c))){
            int j = cccc.indexOf(new Character(c));
            int k =freq.get(j); k++; freq.set(j, k);    found = true;    break;
        }
    }
    //new, add c to dictionary, not 1 (int type)
    if (!found) {cccc.add(c); freq.add(new Integer(1));}
}
```

ArrayList of Character Occurence Counting Example:.....

A=17	J=22	H=10	F=19	E=6	I=3	R=7	U=4	K=14	S=16
D=12	N=3	M=4	W=4	L=7	Y=1	O=3	Q=3	P=2	T=1
Z=2	V=2	C=1	X=1						

Word Occurrence Count

WordCountArrayList.java

```
// ArrayList Version: New Dictionary Representation
static class Word {
    String name = "";
    int count = 0;
}
static ArrayList<Word> dict = new ArrayList<Word>();
```

Note:

(1)**getter** method (**get()**) does not only return an object in the arraylist but also work as an **object pointer**. Using it can access the data in the object.

(2)dict.get(i) works like words[i]

Word Occurrence Count

WordCountArrayList.java

```
for (int i =0; i<words.length; i++) {
    found = false;
    words[i] = words[i].trim();
    if (!words[i].equals("")){ // for non-empty strings
for (int j=0; j<dict.size() && !found; j++)
        if (words[i].equals(dict.get(j).name)){
            dict.get(j).count++;    found = true;
            } // try to find new word in dictionary
If (!found){    Word a = new Word();
            a.name = words[i];
            a.count++;
            dict.add(a);
            } // word not found in current dictionary.
        }
    }
```

WordCountArrayList.java Output

```
biblecount - Notepad
File Edit Format View Help
1532 genesis
12667 in
63924 the
106 beginning
4472 god
45 created
583 heaven
51696 and
987 earth
4522 was
426 without
24 form
24 void
162 darkness
2748 upon
416 face
34734 of
65 deep
505 spirit
75 moved
287 waters
3999 said
1511 let
2299 there
```

```
BlueJ: Terminal Window - Chapter08
Options
1 scorch
1 gnawed
1 armageddon
1 coloured
1 martyrs
1 delicacies
2 deliciously
1 thyme
1 slaves
1 sailors
1 costliness
1 musicians
1 pipers
4 alleluia
1 omnipotent
1 chalcedony
1 sardonyx
1 chrysolyte
1 chrysoprasus
1 transparent
1 proceeding
Total Word Count : 1093544
Total Different Word Count: 12608
```



Demonstration Program

WordCountArrayList.java

[5] ArrayList Reverse Example:

```
ArrayList<Character> original = new ArrayList<Character>(Arrays.asList(new
Character[] {A, B, C, D, E}));
ArrayList<Character> reverse = new ArrayList<Character>();
// perform reverse
for (int i=original.size()-1; i>=0; i--) reverse.add(original.get(i));
// print out
System.out.println("Original="+original+" Reverse="+reverse);
```

ArrayList Reverse Example:.....

Original=[A, B, C, D, E] Reverse=[E, D, C, B, A]

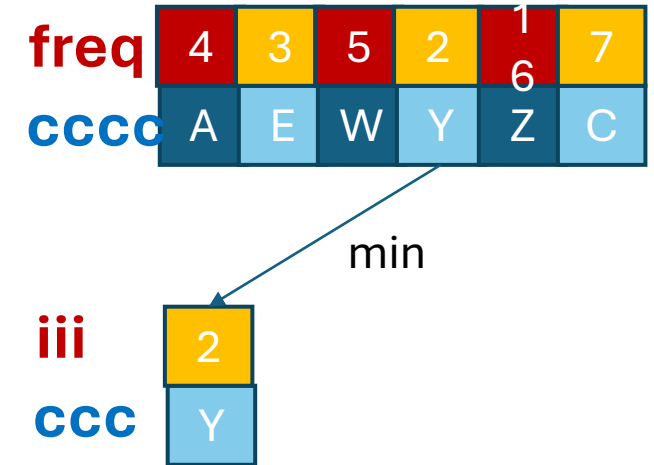
[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

freq	4	3	5	2	1	7
cccc	A	E	W	Y	Z	C

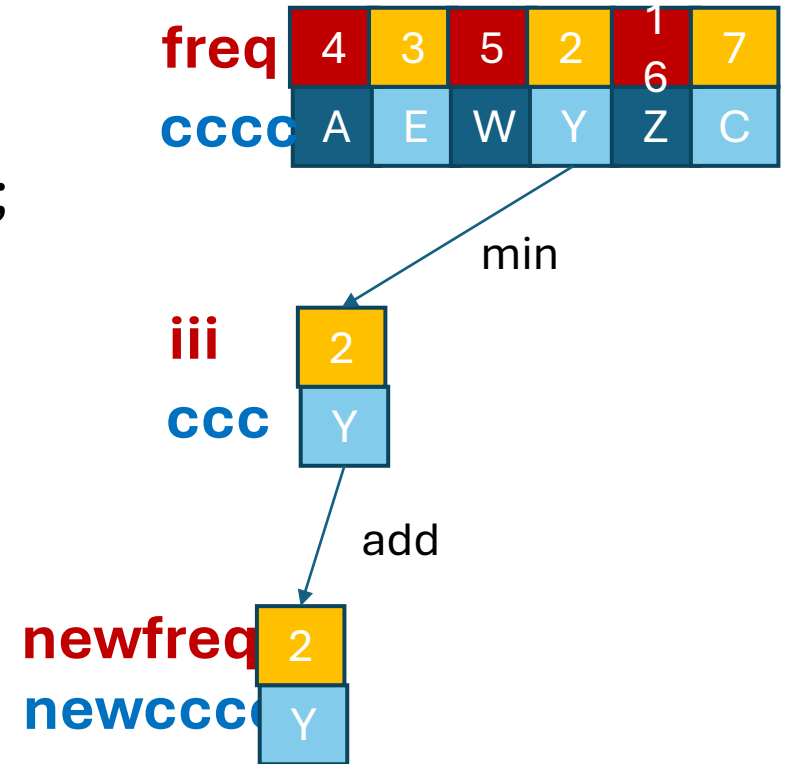
[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

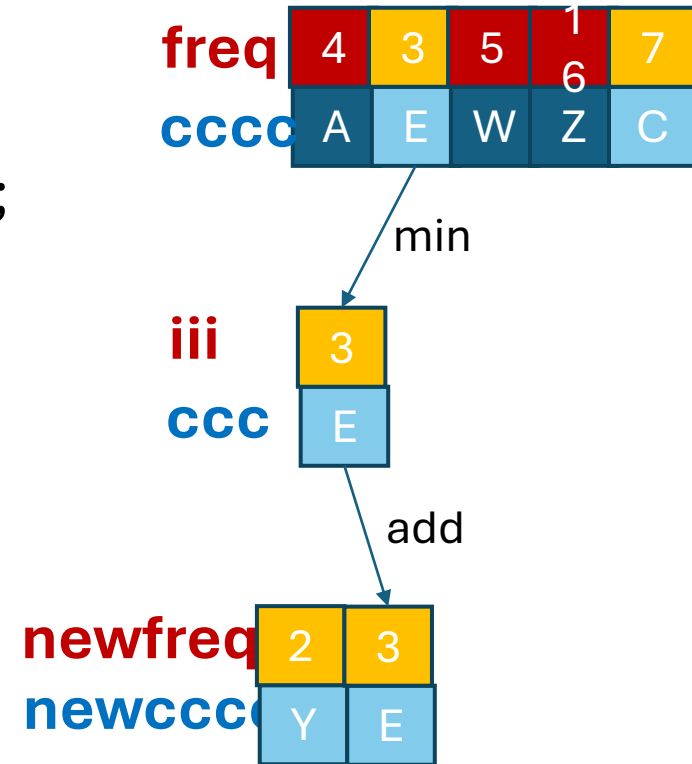
freq	4	3	5	1	7
cccc	A	E	W	Z	C

iii	2
ccc	Y

newfreq	2
newcccc	Y

[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

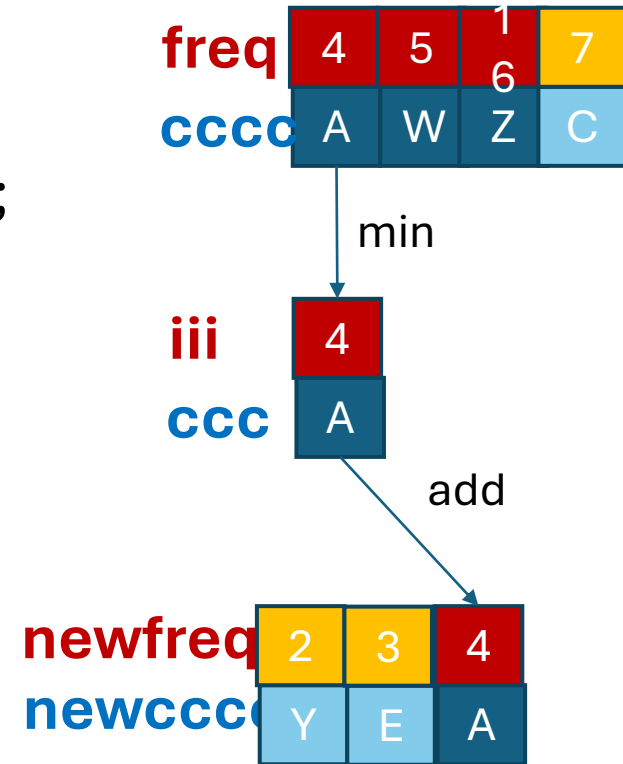
freq	4	5	1	7
cccc	A	W	Z	C

iii	3
ccc	E

newfreq	2	3
newcccc	Y	E

[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

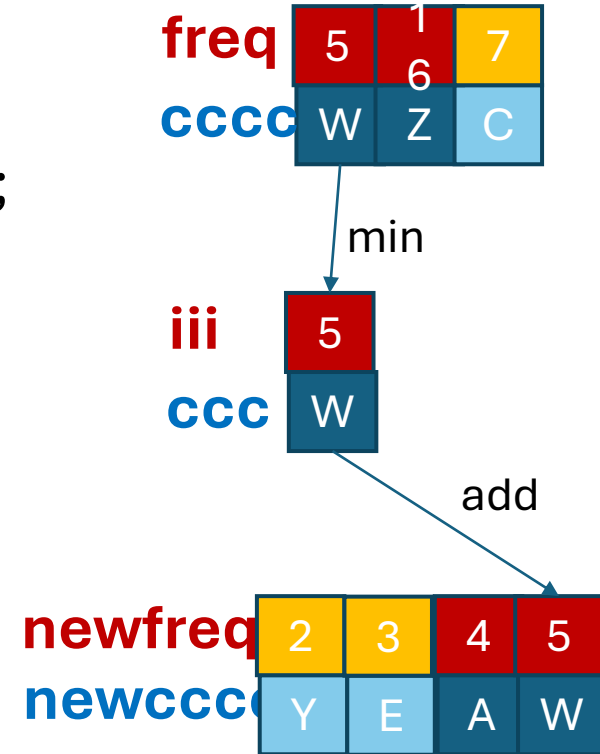
freq	5	1	7
cccc	W	Z	C

iii	4
ccc	A

newfreq	2	3	4
newcccc	Y	E	A

[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

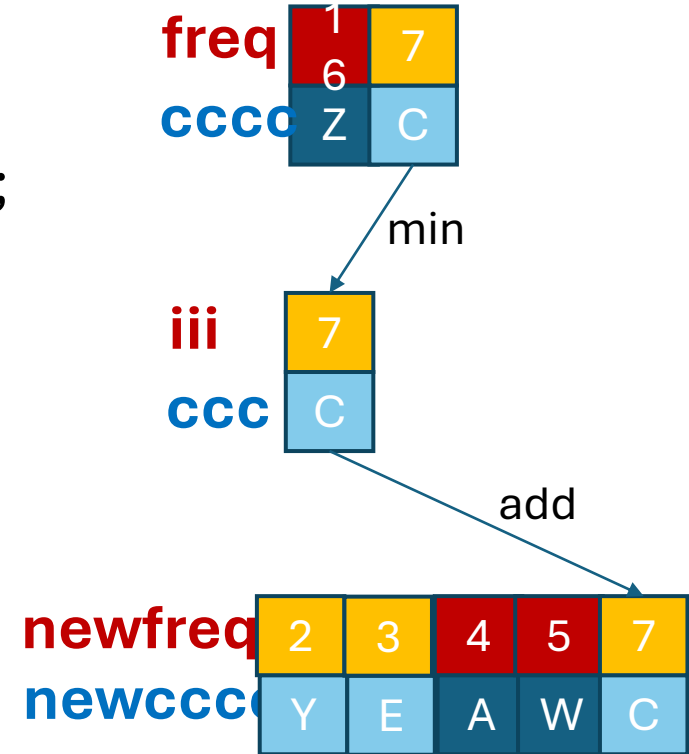
freq	1	7
cccc	Z	C

iii	5
ccc	W

newfreq	2	3	4	5
newcccc	Y	E	A	W

[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

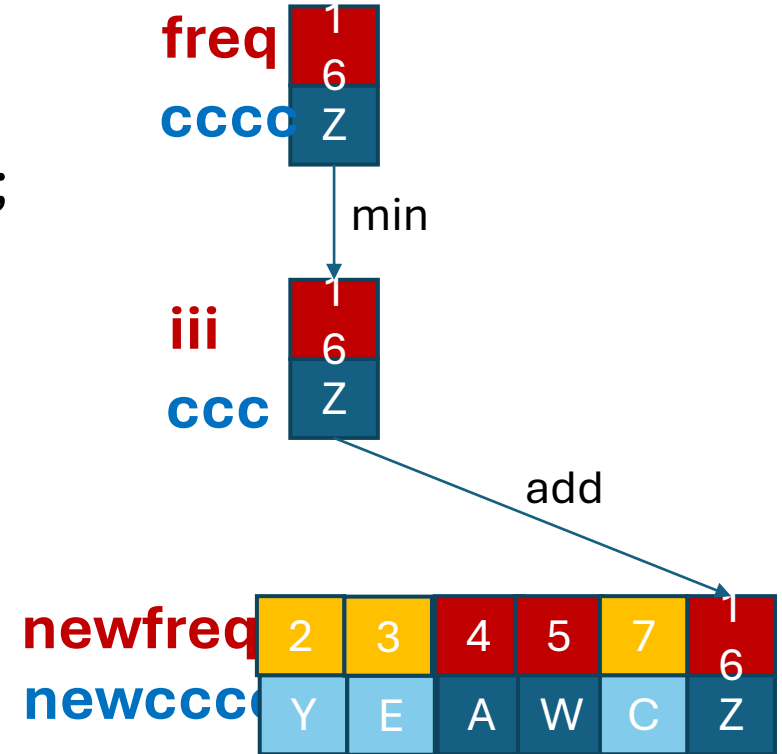
freq	1
	6
cccc	Z

iii	7
ccc	C

newfreq	2	3	4	5	7
newcccc	Y	E	A	W	C

[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```



[6] Sorting on Occurrence of Character (follow-up of Counting the Occurrence of Character)

```
ArrayList<Character> newcccc = new  
ArrayList<Character>();  
ArrayList<Integer> newfreq = new ArrayList<Integer>();  
int len = cccc.size();  
for (int i=0; i<len; i++){  
    int min = min(freq);  
    Character ccc = cccc.get(freq.indexOf(min));  
    Integer iii = freq.get(freq.indexOf(min));  
    newcccc.add(ccc);  
    newfreq.add(iii);  
    cccc.remove(freq.indexOf(min));  
    freq.remove(freq.indexOf(min));  
}
```

freq
cccc

iii
ccc

1
6
Z

newfreq
newcccc

2	3	4	5	7	1
Y	E	A	W	C	Z

[6] Sorting by ArrayList

Advantage:

- Easiest to understand.
- Easy to use.
- Less than 10 lines of code.

Disadvantage:

- $O(n^2)$ Algorithm. Slow in performance.

Download

ArrayProcessingII.java

WordCountArrayList.java

bible.txt

In ArrayProcessingII.zip

```
BlueJ: Terminal Window - Chapter08
Options
ArrayList Traversal Examples:.....
#1 normal for loop
Text 1 Text 2 Text 3
#2 advance for loop
Text 1 Text 2 Text 3
#3 while loop
Text 1 Text 2 Text 3
#4 iterator
Text 1 Text 2 Text 3
ArrayList Iterator Examples:.....
Original contents of al: C A E B D F
Modified list backwards: F+ D+ B+ E+ A+ C+
ArrayList of User-defined Class Examples:.....
101 Sonoo 23
102 Ravi 21
103 Hanumat 25
ArrayList of Character Occurence Counting Example:.....
A=17 J=22 H=10 F=19 E=6 I=3 R=7 U=4 K=14 S=16
D=12 N=3 M=4 W=4 L=7 Y=1 O=3 Q=3 P=2 T=1
Z=2 V=2 C=1 X=1
ArrayList Reverse Example:.....
Original=[A, B, C, D, E] Reverse=[E, D, C, B, A]
ArrayList of Character Occurence Counting Example:.....
Y=1 T=1 C=1 X=1 P=2 Z=2 V=2 I=3 N=3 O=3
Q=3 U=4 M=4 W=4 E=6 R=7 L=7 H=10 D=12 K=14
S=16 A=17 F=19 J=22
```



Parallel Lists Using ArrayLists

Lecture 2

Parallel Array/Parallel ArrayList

- When two or more array or arraylist are used to represent a same set of data. Each index represents data of a same entity across the different array or arraylist. We call them parallel array or parallel arraylist.

```
double[] x = new double[20];
```

```
double[] y = new double[20];
```

- (x, y) represents a point in the Cartesian coordinate.

Design Patterns with ArrayLists

- 1. Available List
- 2. SelectList
- 3. Non-RecurringList
- 4. OccurrenceList
- 5. DifferenceList

Available List

AvailableList.java

- Two ways to implement available list.
 - (1) Use a single list to list all the available element or available indice.
 - (2) Use a separate parallel boolean list to keep track of whether an element is available or not.



Demonstration Program

AvailableList.java

Selection List

SelectionList.java

- Using arraylist for selection sort.
- Selection and remove is the core operations.



Demonstration Program

SelectionList.java

Non Recurring List

NonRecurring.java

- Using ArrayList as a set.
- When an item is added to a set, it will be added only if the list does not contain the item.
- Discussion:
 - (1) Ordered non-recurring list.
 - (2) Ordered non-recurring list.



Demonstration Program

NonRecurring.java

Occurrence List

Occurrence.java

- Using ArrayList as a histogram to keep track of the frequency of each item in the list. This list sometime used along with the non-recurring list.



Demonstration Program

Occurrence.java

Difference List

DifferenceList.java

- The difference list can be used to predict the growth rate for the original list.
- It is quite useful.

```
Before Occurrence Count:
```

```
[8, 7, 9, 5, 9, 8, 9, -3, 2, 0, 17, 5]
```

```
The difference list :
```

```
[-1, 2, -4, 4, -1, 1, -12, 5, -2, 17, -12]
```



Demonstration Program

DifferenceList.java



Comparison of String, Array, ArrayList

Lecture 3

Study of Programming

(Object-Oriented Programming is One Programming Paradigm to Handle All)

Program Structure:

- Study of Algorithm, Programming Paradigm, and Software Engineering.
- **Data Structure:**
- Study of Data Structure, Object, and Classes, Data Bases, and Data Science.

Data structure (From Wikipedia)

- In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently.
- Data structures can implement one or more particular **abstract data types** (ADT), which are the means of specifying the contract of operations and their complexity. In comparison, a data structure is a concrete implementation of the contract provided by an **ADT**.

Data type and data Structure in Programming Language (Not Specific for Java)

Java:

Primitive types

- Boolean, true or false
- Character
- Floating-point, single-precision real number values
- Double, a wider floating-point size
- Integer, integral or fixed-precision values
- Enumerated type, a small set of uniquely named values

Composite types

- Array
- Record (also called tuple or struct)
- Union
- Tagged union (also called variant, variant record, discriminated union, or disjoint union)

Primitive Data Type

byte, char, short, int, float, double

Reference Data Type:

String, Array (Built-in)
Math

Advanced Data Types:

Class (packages, Java API)

Abstract data types

- Container
- List
- Associative array
- Multimap
- Set
- Multiset
- Stack
- Queue
- Double-ended queue
- Priority queue
- Tree
- Graph

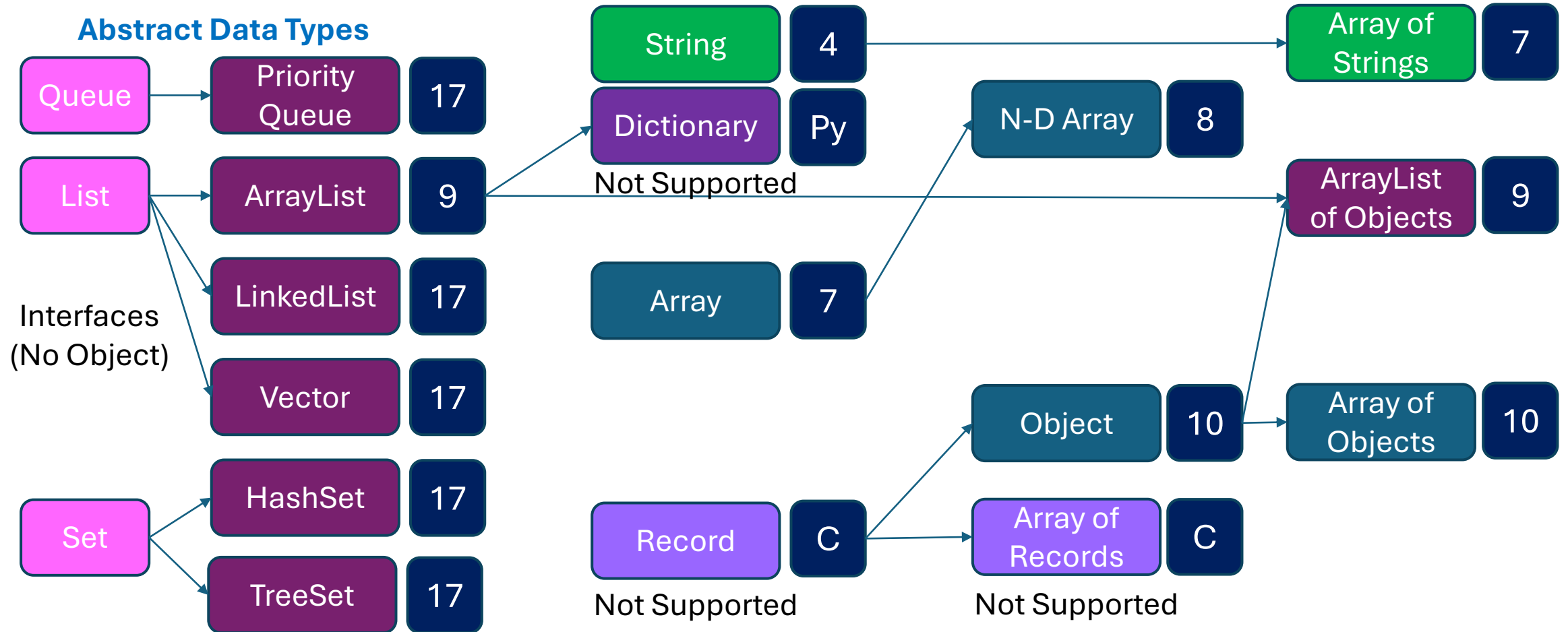
Java Supports for ADT:

Object-Oriented Programming Paradigm.

Classes and Objects (Ch. 9-13)

More Data Structures (Ch. 16)

Data Structure Supported by Java



Comparison between Array and String

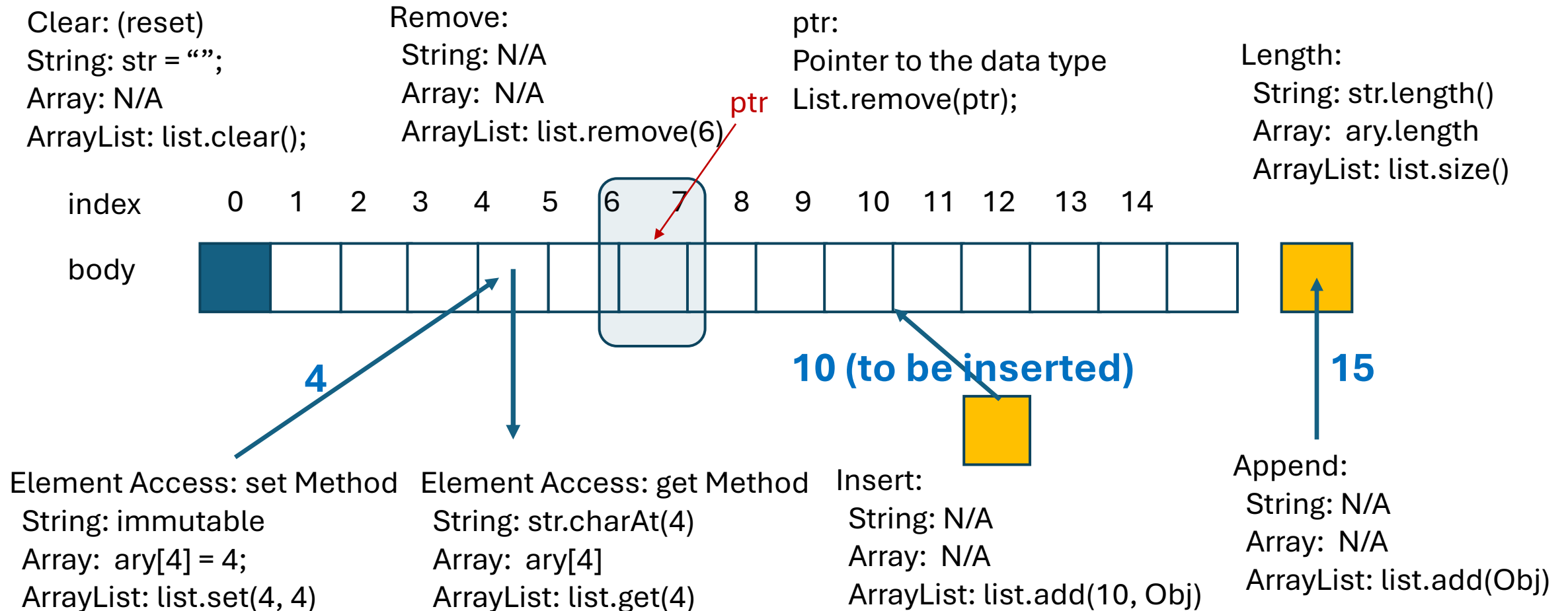
(Using Array of Char as example)

	Array of Character	String
Declaration	<code>char[] chary = {'A', 'B', 'C'};</code>	<code>String str = "ABC";</code>
New Object	<code>char[] chary = new char[3];</code>	<code>String str = new String("ABC");</code>
Access to Elements	<code>chary[2]</code>	<code>str.charAt(2)</code>
Change Content?	Yes	Immutable
Length	<code>chary.length</code>	<code>str.length()</code>
Partial elements	none	<code>substring(1,3)</code>
Easy Indexing	<code>chary[(a+b)/3*4-1+5/2]</code>	<code>str.charAt((a+b)/3*4-1+5/2)</code> can only fetch data
Object Traversal	Yes	No
Easy for <code>println()</code> ?	No	Yes
Concatenation?	No	Yes <code>System.out.println(str+str1+str2)</code>
Application	Tabularize data	Message Processing
Sorting of Elements	Yes	No
Adding new elements	No	No, but allow concatenation to create new string

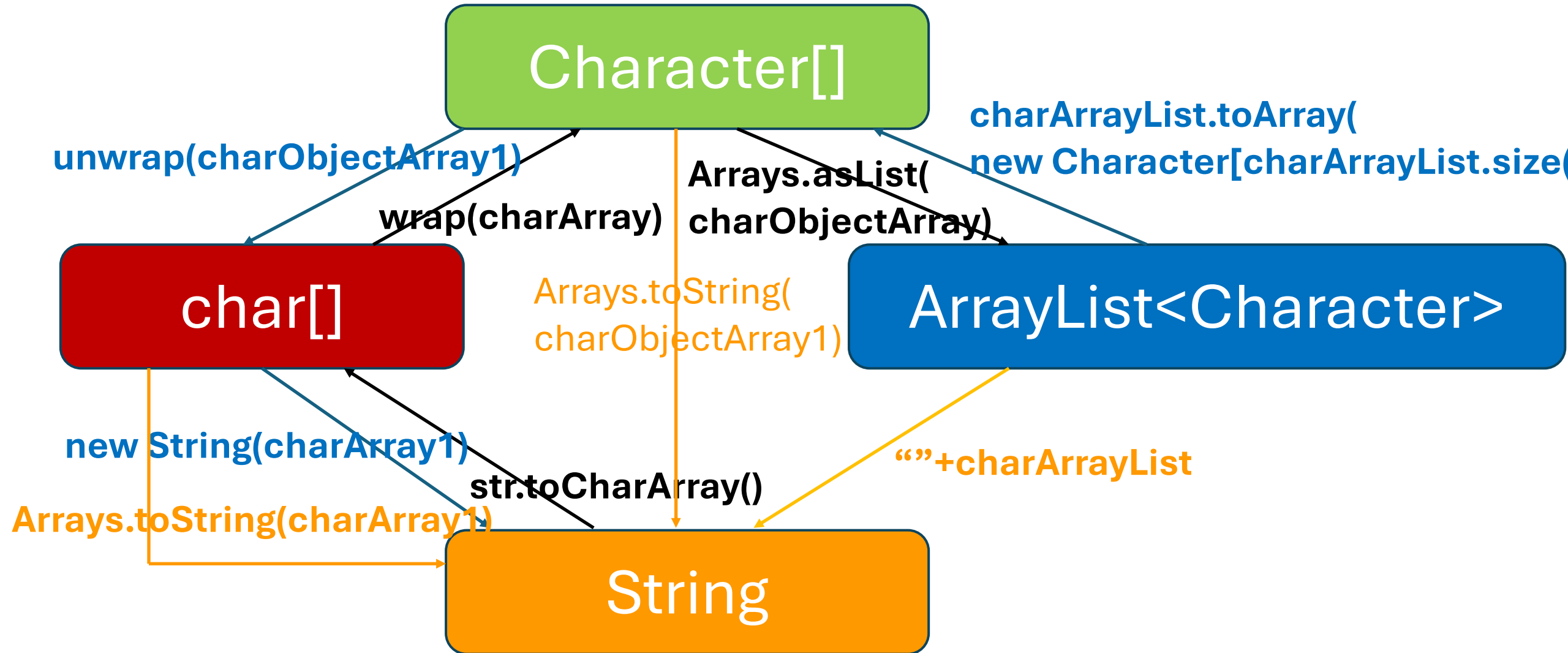
Differences and Similarities between Arrays and ArrayList

<i>Operation</i>	<i>Array</i>	<i>ArrayList</i>
Creating an array/ArrayList <code>ArrayList<>();</code>	<code>String[] a = new String[10]</code>	<code>ArrayList<String> list = new</code>
Accessing an element	<code>a[index]</code>	<code>list.get(index);</code>
Updating an element	<code>a[index] = "London";</code>	<code>list.set(index, "London");</code>
Returning size	<code>a.length</code>	<code>list.size();</code>
Adding a new element		<code>list.add("London");</code>
Inserting a new element		<code>list.add(index, "London");</code>
Removing an element		<code>list.remove(index);</code>
Removing an element		<code>list.remove(Object);</code>
Removing all elements		<code>list.clear();</code>

Operation for String, Array and ArrayList



Conversion Among String, char[], Character[] and ArrayList<Character>





Demonstration Program

Comparison.java

Conversion among String, char[], Character[] and ArrayList<Character>

```
public static void main(String[] args){
    String str = "Java Good!";
    char[] charArray = str.toCharArray();
    System.out.println("String=\""+ str + "\"" to char Array=" + Arrays.toString(charArray));
    Character[] charObjectArray = wrap(charArray);
    System.out.println("char Array=" + Arrays.toString(charArray) + " to Character Array=" + Arrays.toString(charObjectArray));
    ArrayList<Character> charArrayList = new ArrayList<Character>(Arrays.asList(charObjectArray));
    System.out.println("Character Array=" + Arrays.toString(charObjectArray) + " to ArrayList=" + charArrayList);
    Character[] charObjectArray1 = charArrayList.toArray(new Character[charArrayList.size()]);
    System.out.println("ArrayList=" + charArrayList + " to New Character Array=" + Arrays.toString(charObjectArray1));
    char[] charArray1 = unwrap(charObjectArray1);
    System.out.println("New Character Array=" + Arrays.toString(charObjectArray1) + " to New char Array=" + Arrays.toString(charArray1));
    String str1 = new String(charArray1);
    System.out.println("New char Array=" + Arrays.toString(charArray1) + " to New String=\""+ str1+ "\"");
}
```


wrap() and unwrap() to convert between char[] and Character[]

```
public static Character[] wrap(char[] charArray){  
    Character[] charObjectArray = new Character[charArray.length];  
    for (int i=0; i<charArray.length; i++) charObjectArray[i] = Character.valueOf(charArray[i]);  
    return charObjectArray;  
}
```

```
public static char[] unwrap(Character[] charObjectArray){  
    char[] charArray = new char[charObjectArray.length];  
    for (int i=0; i<charObjectArray.length; i++) charArray[i] = charObjectArray[i].charValue();  
    return charArray;  
}
```

Comparison.java

Execution Result for Comparison.java

BlueJ: Terminal Window - Chapter09

Options

```
String="Java Good!" to char Array=[J, a, v, a, , G, o, o, d, !]  
char Array=[J, a, v, a, , G, o, o, d, !] to Character Array=[J, a, v, a, , G, o, o, d, !]  
Character Array=[J, a, v, a, , G, o, o, d, !] to ArrayList=[J, a, v, a, , G, o, o, d, !]  
ArrayList=[J, a, v, a, , G, o, o, d, !] to New Character Array=[J, a, v, a, , G, o, o, d, !]  
New Character Array=[J, a, v, a, , G, o, o, d, !] to New char Array=[J, a, v, a, , G, o, o, d, !]  
New char Array=[J, a, v, a, , G, o, o, d, !] to New String="Java Good!"
```

Object-Oriented Programming

- Welcome to the 2nd Part of Java Programming ...
- Chapter 10: Objects and Classes
- Chapter 11-14: Object-Oriented Programming
 - Object-Thinking
 - Inheritance and Polymorphism
 - Abstract Class and Interfaces
 - File and I/O
- Chapter 15-17 Algorithms



Lab Project:

Student List of Washington High
School

Lecture 4

Background Information

- `ArrayList<E> alist = new ArrayList<E>();`
- `<E>`: generic type.
- ArrayList itself can also be an element to another arraylist. In this way, we can create some sort of 2-D arraylist. That is arraylist of arraylists.

Lab Project:

Washington.java (sample answer)

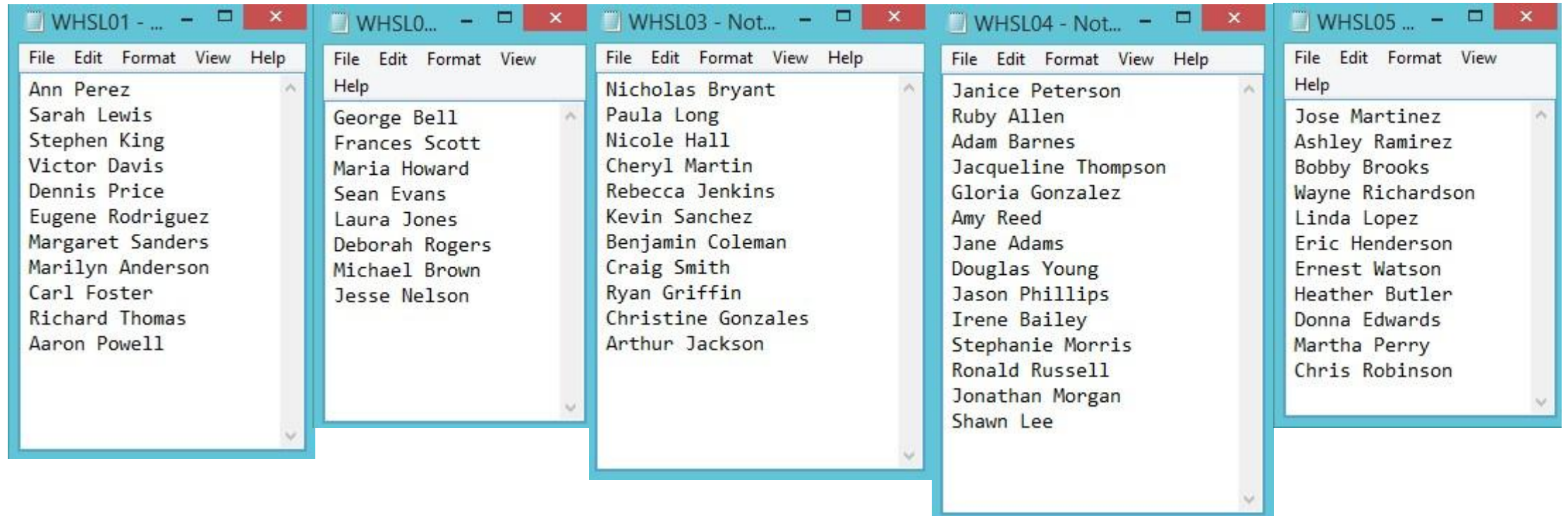
- Write a program to create an arraylist of arraylists. Five files of student names are given (WHSL01.txt, WHSL02.txt, WHSL03.txt, WHSL04.txt, WHSL05.txt). Each file contains a list of student names. Each student name is a line (use input.nextLine to read it in as String is fine).

Then, put these arraylists to another arraylist.

```
ArrayList<ArrayList> schoolList = new  
ArrayList<ArrayList>();
```

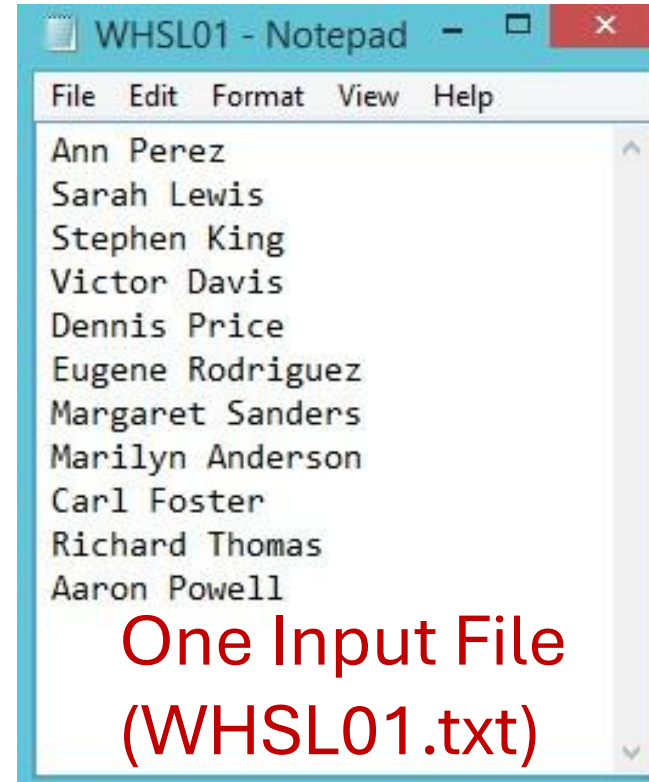
```
ArrayList<String> classList = new ArrayList<String>();
```

Five Student List Files



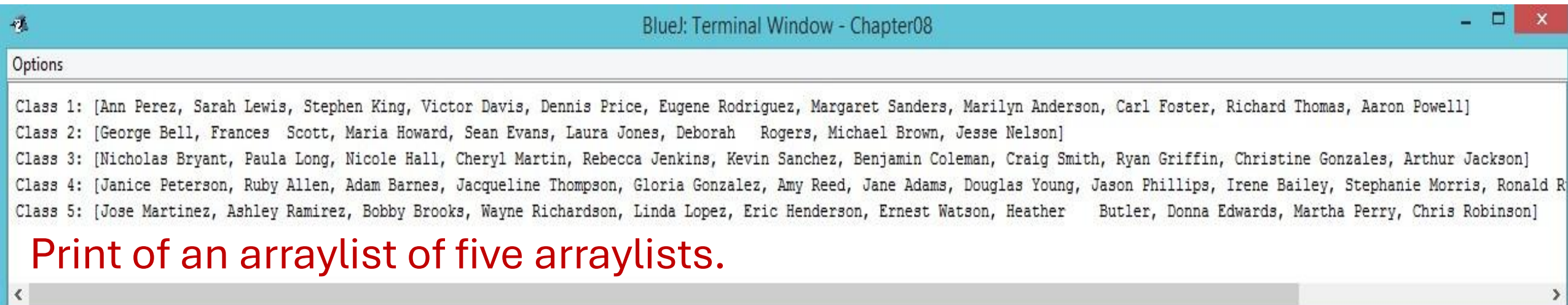
The number of Files can change, and the number of student in a file can also change.

Expected Results:



```
WHSLO1 - Notepad
File Edit Format View Help
Ann Perez
Sarah Lewis
Stephen King
Victor Davis
Dennis Price
Eugene Rodriguez
Margaret Sanders
Marilyn Anderson
Carl Foster
Richard Thomas
Aaron Powell
```

One Input File
(WHSLO1.txt)



```
BlueJ: Terminal Window - Chapter08
Options
Class 1: [Ann Perez, Sarah Lewis, Stephen King, Victor Davis, Dennis Price, Eugene Rodriguez, Margaret Sanders, Marilyn Anderson, Carl Foster, Richard Thomas, Aaron Powell]
Class 2: [George Bell, Frances Scott, Maria Howard, Sean Evans, Laura Jones, Deborah Rogers, Michael Brown, Jesse Nelson]
Class 3: [Nicholas Bryant, Paula Long, Nicole Hall, Cheryl Martin, Rebecca Jenkins, Kevin Sanchez, Benjamin Coleman, Craig Smith, Ryan Griffin, Christine Gonzales, Arthur Jackson]
Class 4: [Janice Peterson, Ruby Allen, Adam Barnes, Jacqueline Thompson, Gloria Gonzalez, Amy Reed, Jane Adams, Douglas Young, Jason Phillips, Irene Bailey, Stephanie Morris, Ronald R
Class 5: [Jose Martinez, Ashley Ramirez, Bobby Brooks, Wayne Richardson, Linda Lopez, Eric Henderson, Ernest Watson, Heather Butler, Donna Edwards, Martha Perry, Chris Robinson]
```

Print of an arraylist of five arraylists.



Chapter Project:

Sorting the Words in Bible by
their Occurrence

Lecture 5

Data to be sorted:

biblecountunsorted.txt

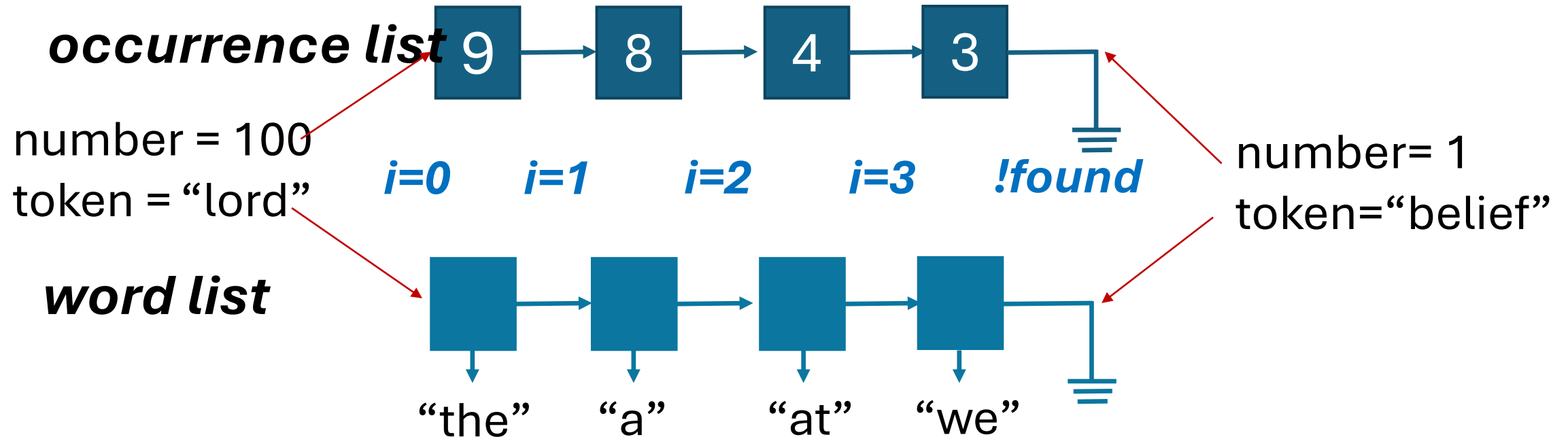
- The biblecountunsorted.txt is copied from biblecount.txt which is generated by WordCountArrayList.java.
- Only the occurrence information of each word in Bible is used to be sorted. (The data field that is used to perform sorting is called **key** field.) So, that we can have a complete listing of words in Bible (in descending order of their occurrence).

Chapter Project:

- Write a program to read in the biblecountunsorted.txt file which has the occurrence and word listing of the whole bible. Read in this file line by line and put the occurrence in one arraylist and the word in another arraylist in a sorted format.
- Then, print the sorted occurrence and word information back to a file named **biblecountsorted.txt**

Pseudo code

- When you add the element, check the existing arraylist for the right location that the occurrence and the word should be inserted by traversing through the occurrence list.

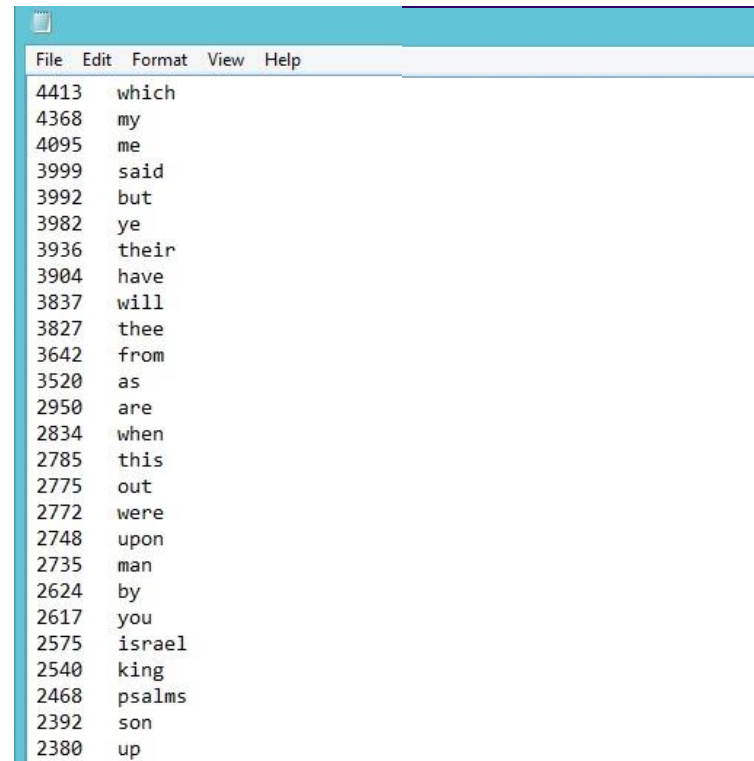


Pseudo code

```
while (input.hasNext()){  
    number is a occurrence number read from file.  
    token is the word read in from file.  
    found = false  
    for (int i = 0; i<list.size() && !found; i++){  
        i is the index number to insert the number and token into the lists.  
        if (number > bible_word_occurrence.get(i) && !found) {  
            add the number at word_occurrence arraylist with index i;  
            also add the token to bible_arraylist word at the same index i.  
            found = true;  
        } // this index i has anything before this location is greater than number  
    }  
    if (!found) add the occurrence and word to the end of the arraylists.  
}
```

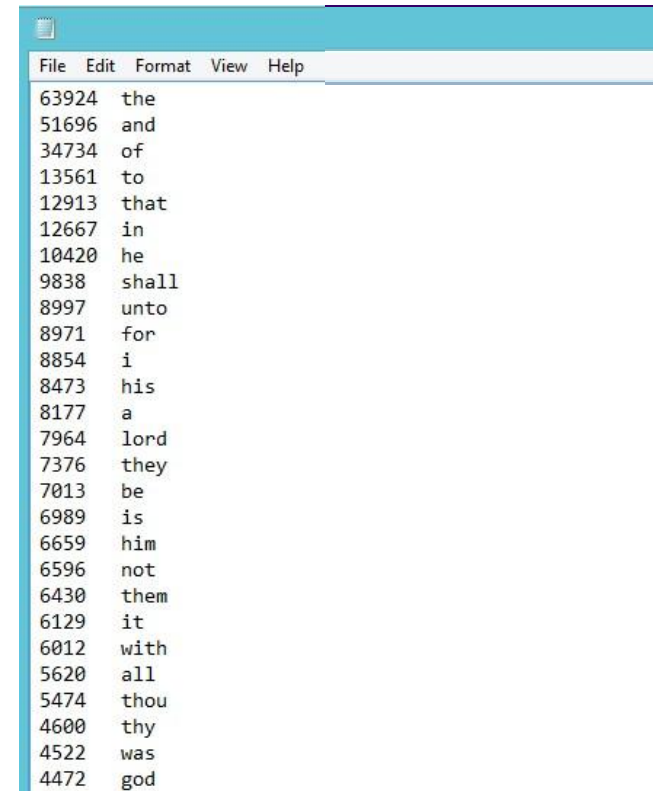
Traversing through the two arraylists and print them out.

(sample answer:
WordCountBibleSorted.java
)



A screenshot of a text editor window with a menu bar (File, Edit, Format, View, Help). The text is a list of words and their counts, sorted by count in descending order. The list starts with 'which' (4413) and ends with 'up' (2380).

Count	Word
4413	which
4368	my
4095	me
3999	said
3992	but
3982	ye
3936	their
3904	have
3837	will
3827	thee
3642	from
3520	as
2950	are
2834	when
2785	this
2775	out
2772	were
2748	upon
2735	man
2624	by
2617	you
2575	israel
2540	king
2468	psalms
2392	son
2380	up



A screenshot of a text editor window with a menu bar (File, Edit, Format, View, Help). The text is a list of words and their counts, sorted by count in descending order. The list starts with 'the' (63924) and ends with 'god' (4472).

Count	Word
63924	the
51696	and
34734	of
13561	to
12913	that
12667	in
10420	he
9838	shall
8997	unto
8971	for
8854	i
8473	his
8177	a
7964	lord
7376	they
7013	be
6989	is
6659	him
6596	not
6430	them
6129	it
6012	with
5620	all
5474	thou
4600	thy
4522	was
4472	god



Lab

WordCountBibleSorted.java