AP Computer Science A

Java Programming Essentials   [Ver. 3.0]

Unit 3: Basic Data Structures

CHAPTER 7A: ARRAYS

DR. ERIC CHOU   IEEE SENIOR MEMBER

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)

- Bible word count sorted by occurrence project

- Washington High School Project

- Conclusion of Data Structures

ArrayList Processing II

LECTURE 1

ArrayList Processing II

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?

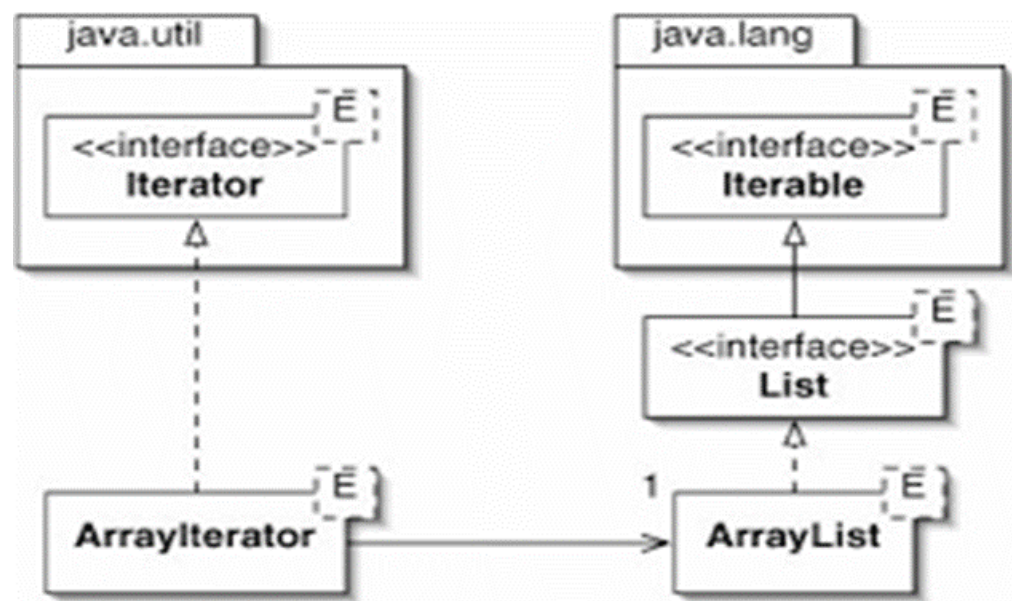(1) By index: (access by **index**)
```
for (int index=0; i<arrayList.size(); i++)
        System.out.println(arrayList.get(i));
```

(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());
```

```
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
```

[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**.
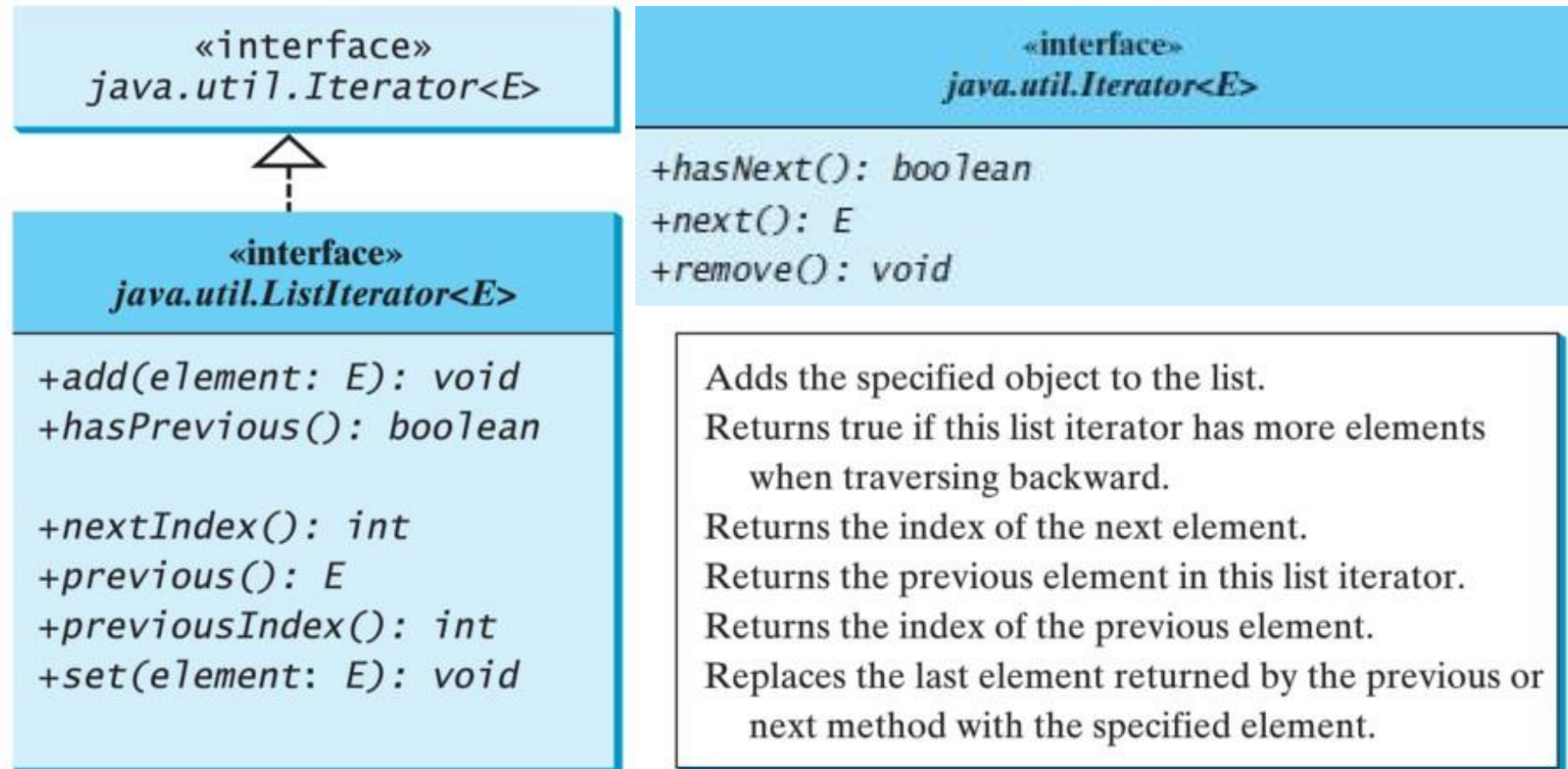
itr

boolean hasNext()

next()

ArrayList

0   1   2   3   4   5   6   7   8   9

Iterator of ArrayList (Iterator and ListIterator)

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

«interface»
java.util.Iterator<E>

«interface»
java.util.ListIterator<E>

+add(element: E): void
+hasPrevious(): boolean

+nextIndex(): int
+previous(): E
+previousIndex(): int
+set(element: E): void

«interface»
java.util.Iterator<E>

+hasNext(): boolean
+next(): E
+remove(): void

Adds the specified object to the list.
Returns true if this list iterator has more elements
    when traversing backward.
Returns the index of the next element.
Returns the previous element in this list iterator.
Returns the index of the previous element.
Replaces the last element returned by the previous or
    next method with the specified element.

ListIterator Versus Iterator

```java
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

```java
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

```java
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;
    }
}
```

```java
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

```java
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

```java
// ArrayList Version:  New Dictionary Represenation
    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

```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       thyine
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

[5] ArrayList Reverse Example:

```java
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)

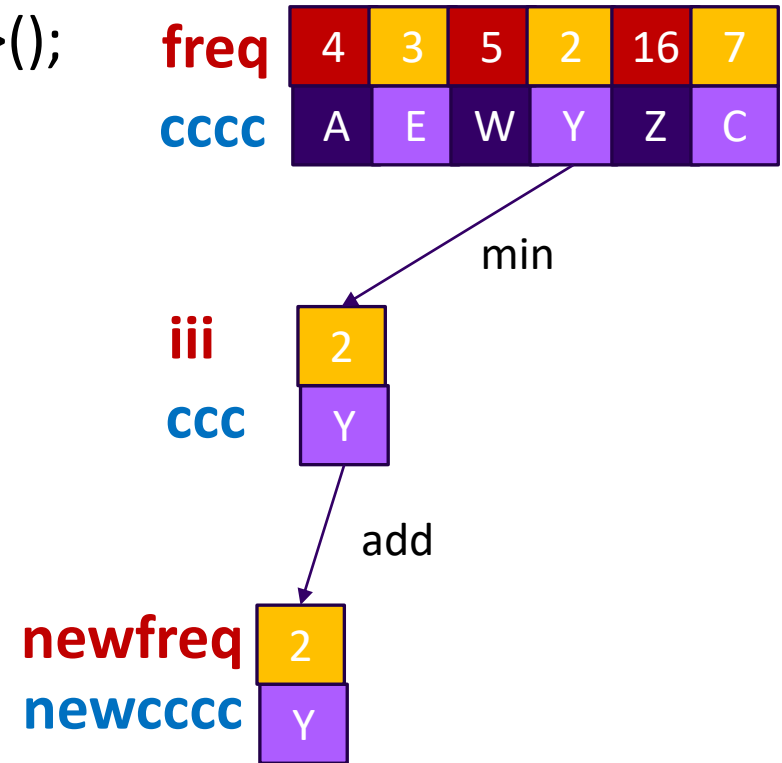| freq	4	3	5	2	16	7
cccc	A	E	W	Y	Z	C

```
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 | 2 | 16 | 7

cccc | A | E | W | Y | Z | C

min

iii | 2

ccc | 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));
}
```
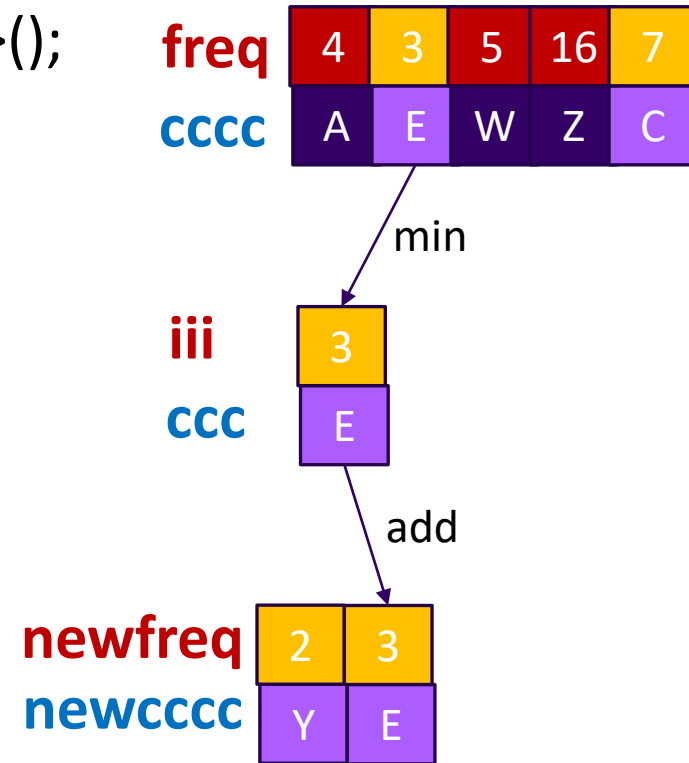
freq

4	3	5	2	16	7

cccc

A	E	W	Y	Z	C

min

iii | 2 |

ccc | Y |

add

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));
}
```

freq

4	3	5	16	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));
}
```
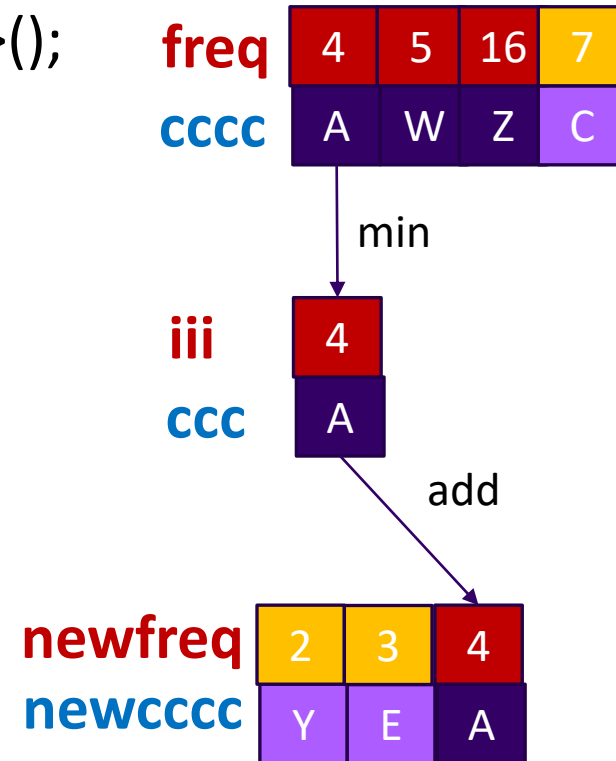
freq

4	3	5	16	7

cccc

A	E	W	Z	C

min

iii

3

ccc

E

add

newfreq

2	3

newcccc

Y	E

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

```java
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));
}
```
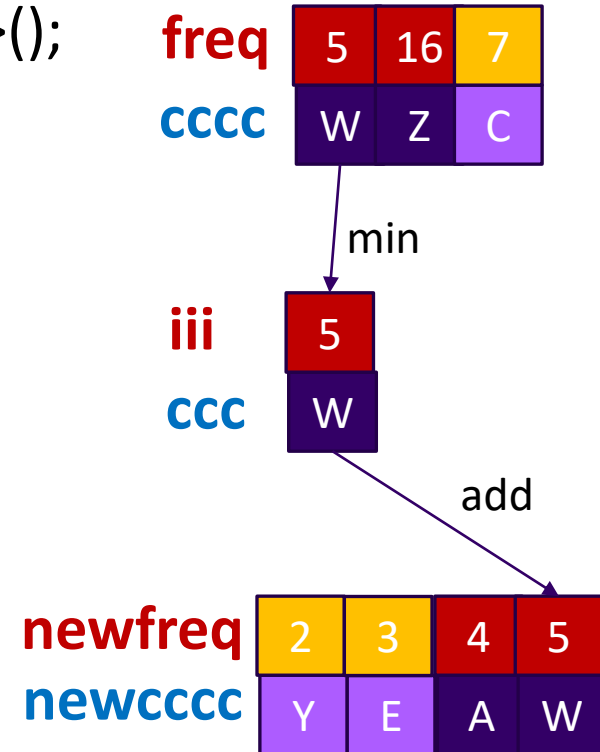
freq

4	5	16	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));
}
```

freq

4	5	16	7

cccc

A	W	Z	C

min

iii

4

ccc

A

add

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));
}
```
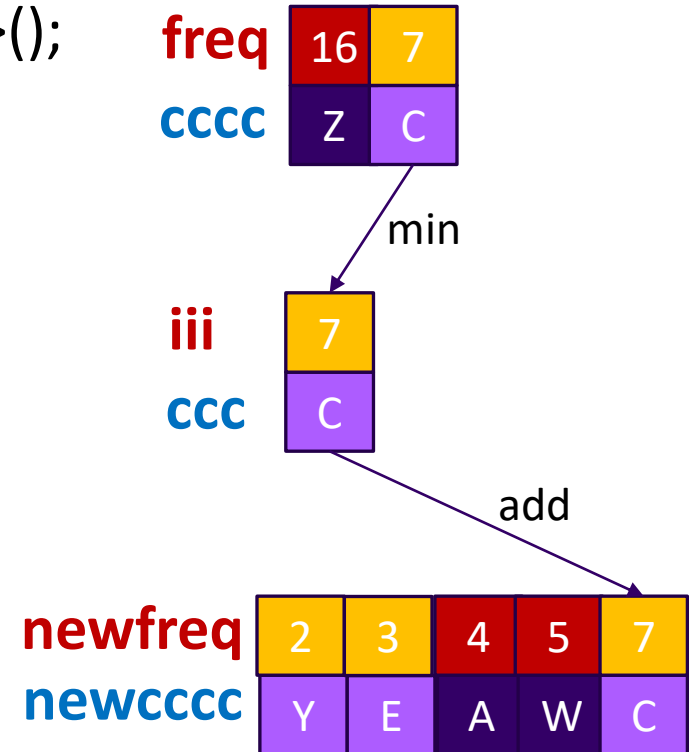
freq | 5 | 16 | 7

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));
}
```

freq

5	16	7

cccc

W	Z	C

min

iii

5

ccc

W

add

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));
}
```
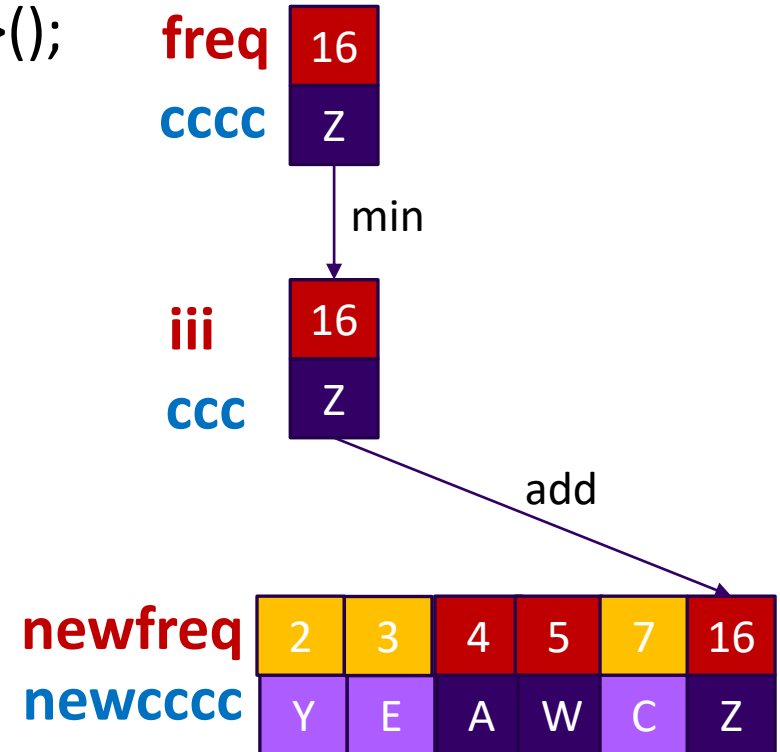
freq

16	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));
}
```

freq | 16 | 7
cccc | Z | C

min

iii | 7
ccc | C

add

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));
}
```

freq | 16
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));
}
```

freq 16

cccc Z

min

iii 16

ccc Z

add

newfreq 2 3 4 5 7 16

newcccc Y E A W C Z

[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 | 16 |

ccc | Z |

newfreq | 2 | 3 | 4 | 5 | 7 | 16 |

newcccc | 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
```

ArrayList Removal Issue

LECTURE 2

Concurrent Modification Issue

```java
static String[] a = {"A", "A", "A", "A", "B", "C", "D", "A", "E", "A", "A", "F", "G"};

public static void main(String[] args){
    System.out.print("\f");

    ArrayList<String> alist = new ArrayList<String>(Arrays.asList(a));
    System.out.println("Original: "+alist);

    // remove all A
    for (int i=0; i<alist.size(); i++){
        if (alist.get(i).equals("A")) alist.remove(i);
    }

    System.out.println("After removal: "+alist);
}
```

```
Original: [A, A, A, A, B, C, D, A, E, A, A, F, G]
After removal: [A, A, B, C, D, E, A, F, G]
```

Backward Traversal

```java
public class ArrayListRemoval_S1
{
    static String[] a = {"A", "A", "A", "A", "B", "C", "D", "A", "E", "A", "A", "F", "G"};

    public static void main(String[] args){
        System.out.print("\f");

        ArrayList<String> alist = new ArrayList<String>(Arrays.asList(a));
        System.out.println("Original: "+alist);

        // remove all A
        for (int i=alist.size()-1; i>=0; i--){
            if (alist.get(i).equals("A")) alist.remove(i);
        }

        System.out.println("After removal: "+alist);
    }
}
```

Forward Traversal with Conditional Advancement of Index

```java
public class ArrayListRemoval_S2{
    static String[] a = {"A", "A", "A", "A", "B", "C", "D", "A", "E", "A", "A", "F", "G"};

    public static void main(String[] args){
        System.out.print("\f");
        ArrayList<String> alist = new ArrayList<String>(Arrays.asList(a));
        System.out.println("Original: "+alist);

        // remove all A
        for (int i=0; i<alist.size(); ){
            if (alist.get(i).equals("A")) {
                alist.remove(i);
            }
            else i++;
        }

        System.out.println("After removal: "+alist);
    }
}
```

Maximum Run Problem

LECTURE 3

Maximum Run of Same Numbers in an Array

- How to figure out the largest run of a same number in an array?

```
static int[] a = {1,1, 2, 3, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 10, 10, 10, 10, 11, 12};
```

- Find the cuts (where the number changes), and add two more cuts (0, and a.length).

- The difference between cuts are the sizes of the runs.

```java
static int[] a = {1,1, 2, 3, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 10, 10, 10, 10, 11, 12};

public static int maxRun(int[] a){
    ArrayList<Integer> cuts = new ArrayList<Integer>();

    for (int i=1; i<a.length; i++){
        if (a[i] != a[i-1]) cuts.add(i);
    }

    cuts.add(0, 0);
    cuts.add(a.length);

    int max = 1;
    for (int i=1; i<cuts.size(); i++){
        int span = cuts.get(i) - cuts.get(i-1);
        if (span > max) max = span;
    }
    return max;
}
```

Demonstration Program

MAXRUN.JAVA

Interval List Generation

LECTURE 4

Interval List Generation

- How to figure out the largest run of a same number in an array?

```
static int[] a = {1,1, 2, 3, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8, 9, 10, 10, 10, 10, 10, 11, 12};
```

- Find the cuts (where the number changes), and add two more cuts (0, and a.length).

- Each pair of the cuts represent the start of a run of same numbers and the end of the same run of same numbers.

Interval Class

```java
static class Interval{
    int start;
    int end;
    Interval(int t, int e){ start = t; end = e; }
    public String toString(){ return "<"+start+","+end+">"; }
}
```

Create Interval

```java
public static ArrayList<Interval> createInterval(int[] a){
    ArrayList<Integer> cuts = new ArrayList<Integer>();

    for (int i=1; i<a.length; i++){
        if (a[i] != a[i-1]) cuts.add(i);
    }

    cuts.add(0, 0);
    cuts.add(a.length);

    ArrayList<Interval> ilist = new ArrayList<Interval>();
    for (int i=1; i<cuts.size(); i++){
        int s = cuts.get(i-1);
        int e = cuts.get(i);
        ilist.add(new Interval(s, e));
    }
    return ilist;
}
```

```java
public static void main(String[] args){
    System.out.print("\f");

    System.out.println(createInterval(a));
}
```

[<0,2>, <2,3>, <3,4>, <4,7>, <7,9>, <9,13>, <13,15>, <15,16>, <16,17>, <17,22>, <22,23>, <23,24>]

Demonstration Program

INTERVALLIST.JAVA

Parallel Lists Using ArrayLists

LECTURE 5

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. SelectionList

3. Non-RecurringList

4. OccurrenceList

5. DifferenceList

Available List

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

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

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 Occurrenc 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

Lab Project:
Bible Word Count

LECTURE 6

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.

occurrence list

| 9 | → | 8 | → | 4 | → | 3 |

number = 100
token = "lord"

i=0   i=1   i=2   i=3   *!found*

number= 1
token="belief"

word list

"the"   "a"   "at"   "we"
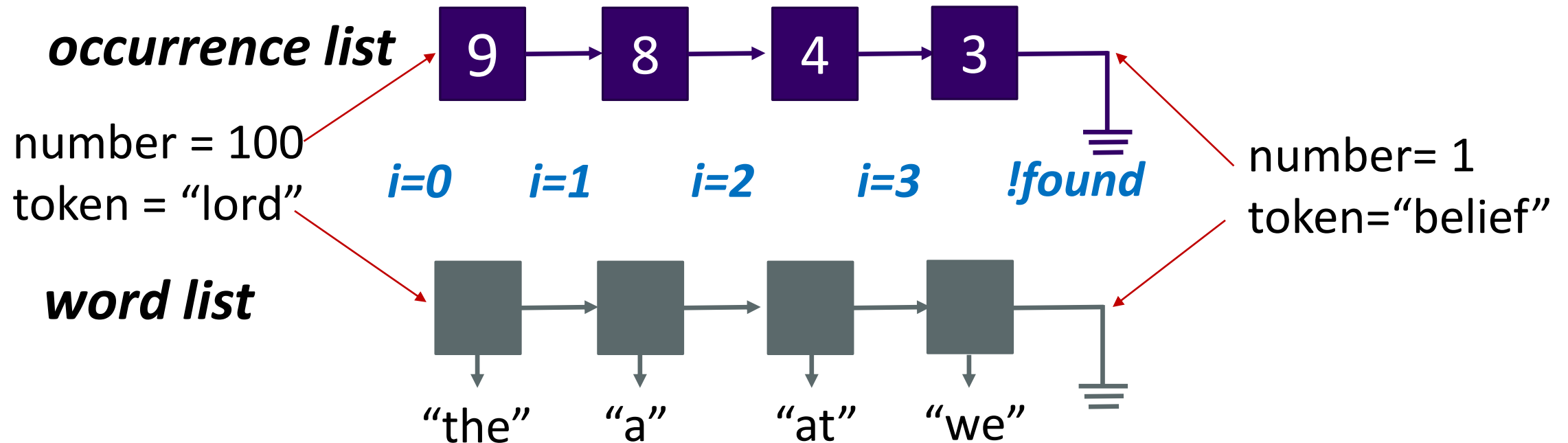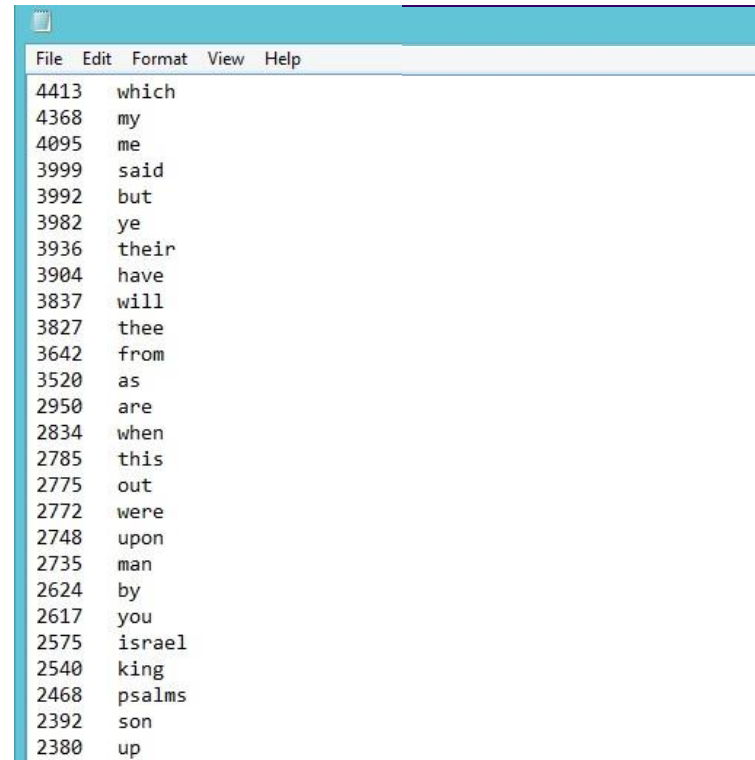
Pseudo code

```
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.
```
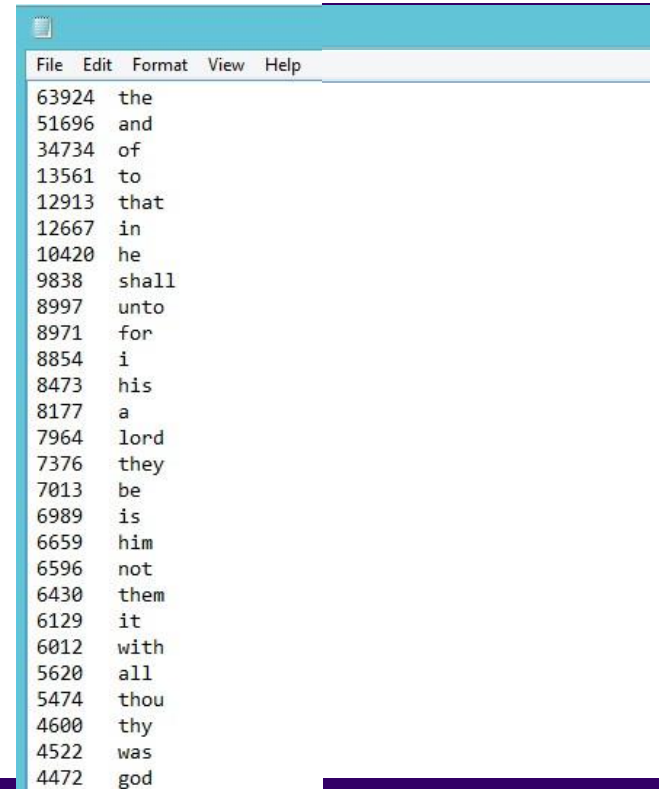
(sample answer:
WordCountBibleSorted.java)

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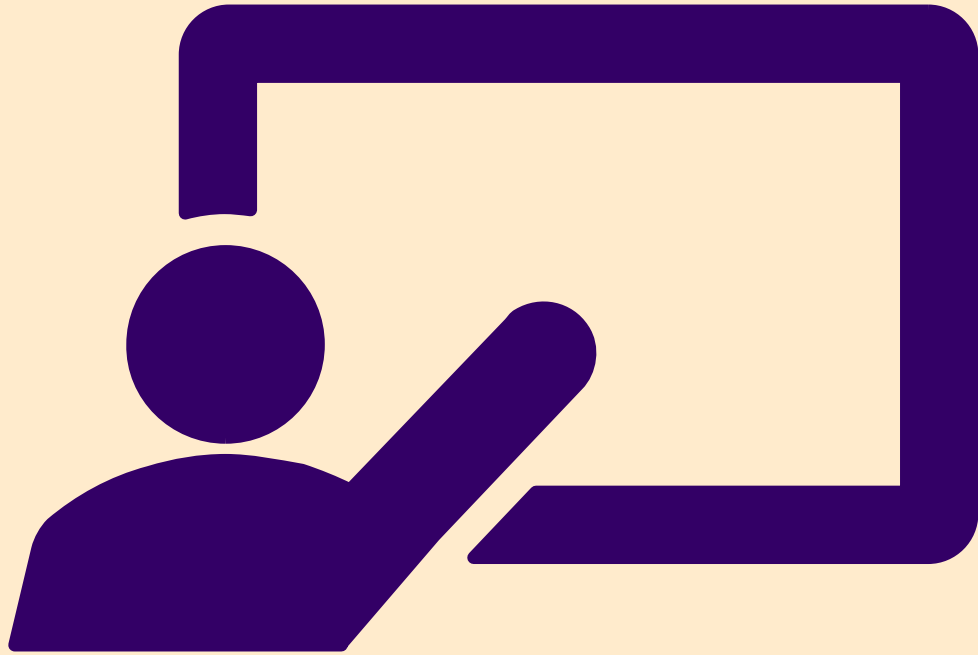
```
63924   the
51696   and
34734   of
13561   to
12913   that
12667   in
10420   he
9838    shall
8997    unto
8971    for
8854    i
8473    his
8177    a
7964    lord
7376    they
7013    be
6989    is
6659    him
6596    not
6430    them
6129    it
6012    with
5620    all
5474    thou
4600    thy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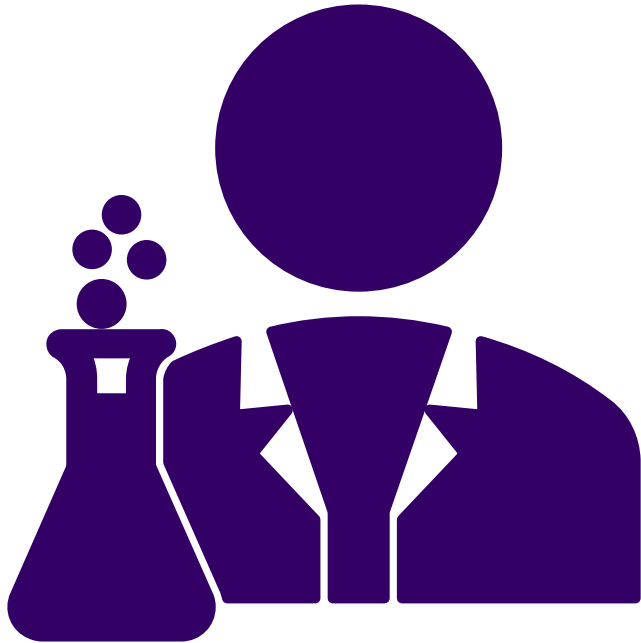
Lab

WORDCOUNTBIBLESORTED.JAVA

Lab Project:
Student List of Washington High School

LECTURE 7

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).
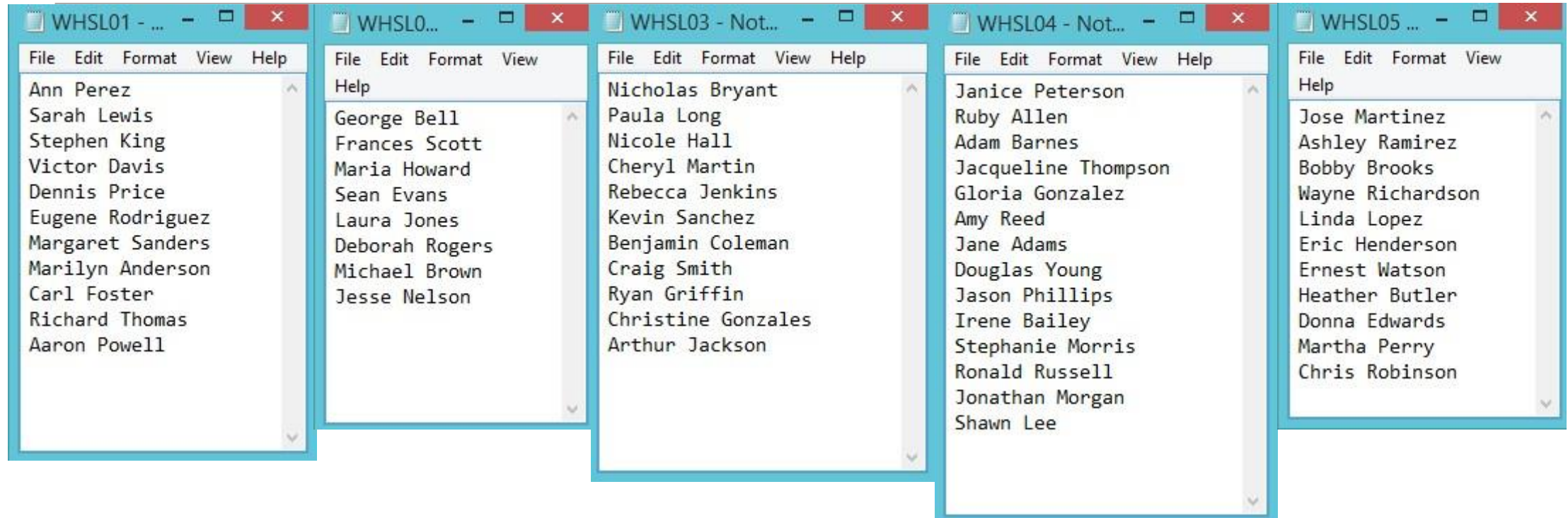
Then, put these arraylists to another arraylist.

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:

WHSL01 - 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
(WHSL01.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]
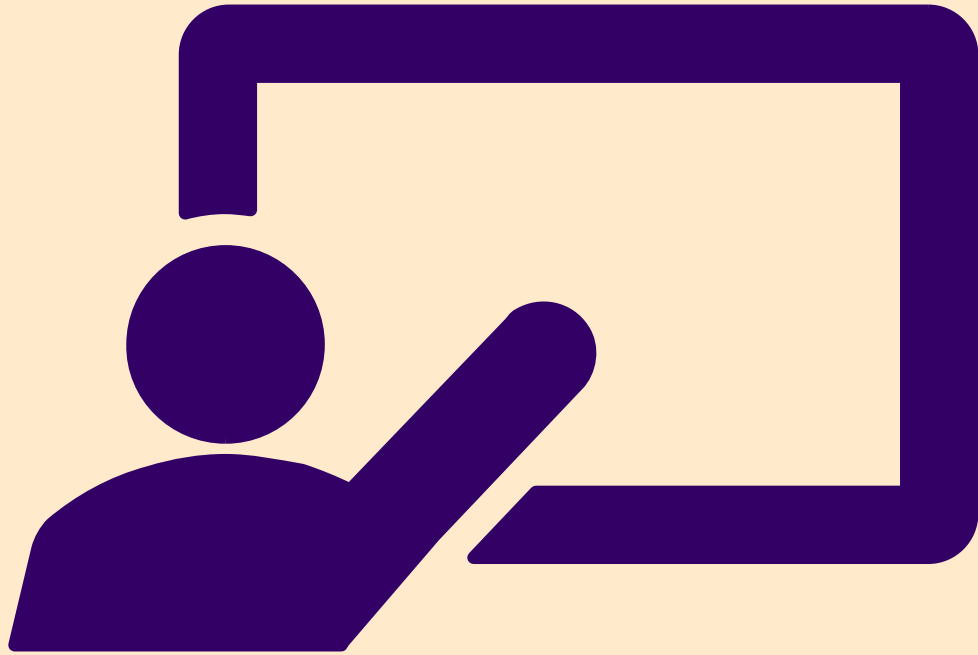
Print of an arraylist of five arraylists.

Chapter Project:
Sorting the Words in Bible by their Occurrence

LECTURE 6

Conclusion of Data Structure Unit

LECTURE 8

Study of Programming

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)

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)

Java:

Primitive Data Type
byte, char, shot, 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
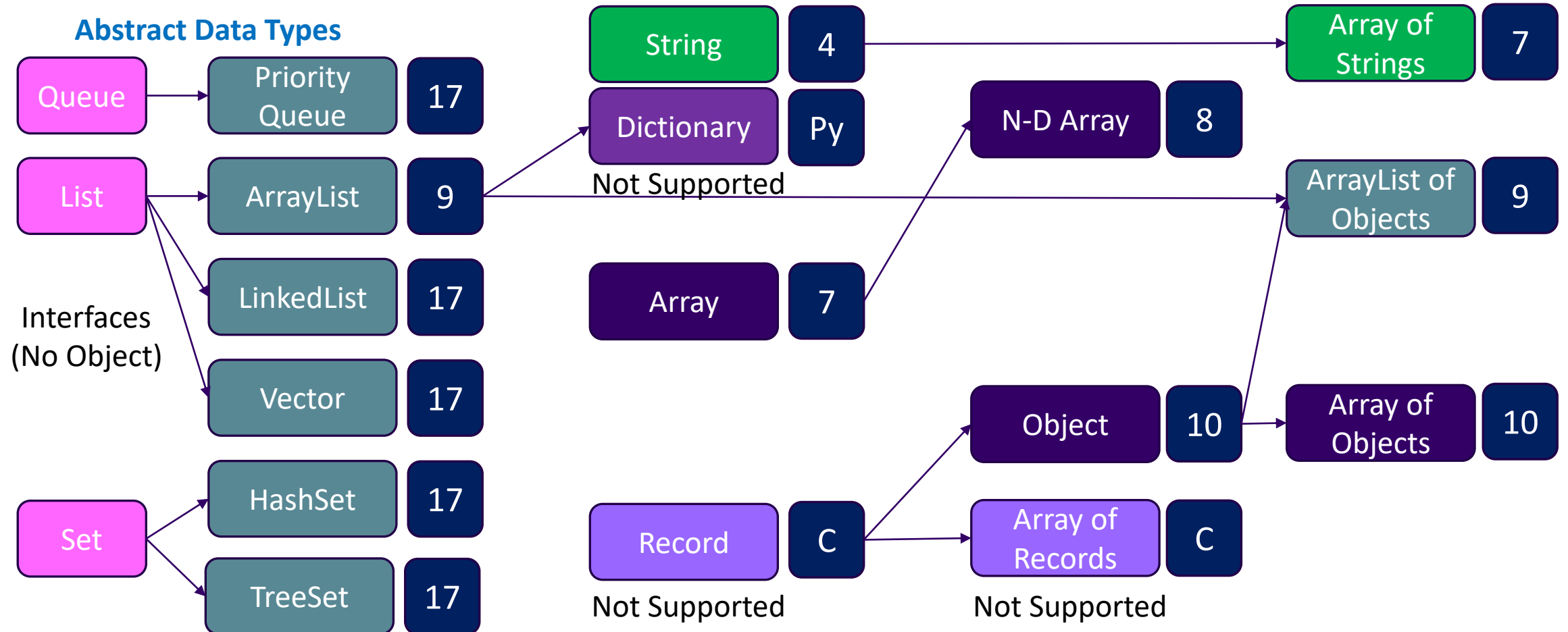- Priority queue
- Tree
- Graph

Java Supports for ADT:
Object-Oriented Programming Paradigm.
Classes and Objects (Ch. 9-13)
More Data Structures (Ch. 16)

Data Structure Supported by Java

Abstract Data Types

Queue → Priority Queue | 17

List → ArrayList | 9

Interfaces (No Object)

List → LinkedList | 17

List → Vector | 17

Set → HashSet | 17

Set → TreeSet | 17

String | 4 → Array of Strings | 7

Dictionary | Py
Not Supported

N-D Array | 8

ArrayList of Objects | 9

Array | 7

Object | 10 → Array of Objects | 10

Record | C
Not Supported
→ Object | 10
→ Array of Records | C
Not Supported

Comparison between Array and String
(Using Array of Char as example)

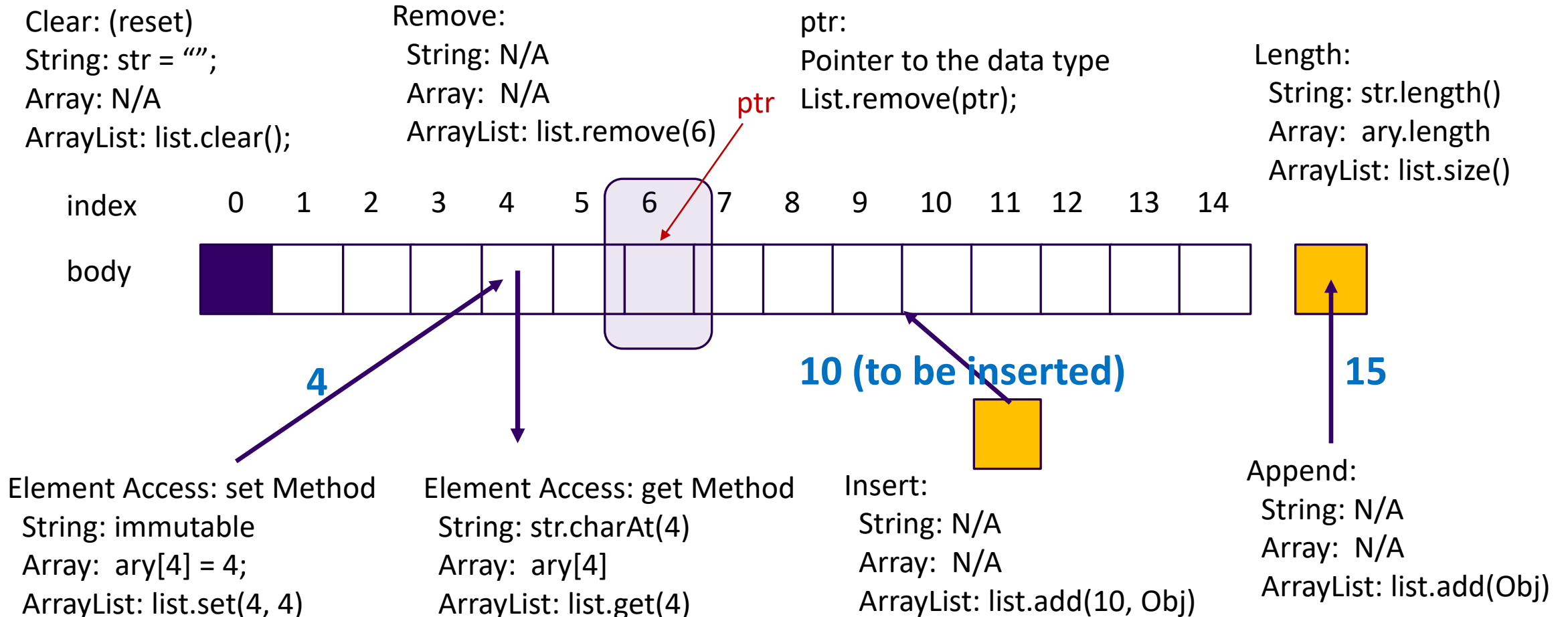|	Array of Character	String
Declaration	char[] chary = {'A', 'B', 'C'};	String str = "ABC";
New Object	char[] chary = new char[3];	String str = new String("ABC");
Access to Elements	chary[2]	str.charAt(2)
Change Content?	Yes	Immutable
Length	chary.length	str.length()
Partial elements	none	substring(1,3)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

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

Operation for String, Array and ArrayList

Clear: (reset)
String: str = "";
Array: N/A
ArrayList: list.clear();

Remove:
String: N/A
Array: N/A
ArrayList: list.remove(6)

ptr:
Pointer to the data type
List.remove(ptr);

ptr

Length:
  String: str.length()
  Array: ary.length
  ArrayList: list.size()

index   0   1   2   3   4   5   6   7   8   9   10   11   12   13   14

body

4

10 (to be inserted)

15

Element Access: set Method
String: immutable
Array: ary[4] = 4;
ArrayList: list.set(4, 4)

Element Access: get Method
  String: str.charAt(4)
  Array: ary[4]
  ArrayList: list.get(4)

Insert:
  String: N/A
  Array: N/A
  ArrayList: list.add(10, Obj)

Append:
  String: N/A
  Array: N/A
  ArrayList: list.add(Obj)

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

Demonstration Program

COMPARISON.JAVA

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

```java
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 unwarp() to convert between char[] and Character[]

```java
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;
}
```

```java
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

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