HOW TO VISIT EVERY ELEMENTS IN ARRAY ?

```
for(int i=0; i<array.length; i++){   }
```

A COMMON SCHEME FOR STEPPING THROUGH ALL ELEMENTS IN ANY COLLECTION

ITERATOR

WHAT'S ITERATOR ?

An iterator is a software design pattern that abstracts the process of scanning through a sequence of elements, one element at a time.

Every data structure in java can convert to iterator by method iterator()

MAIN ITERATOR FUNCTIONS

- *hasNext( ): Returns true if there is at least one additional element in the sequence, and false otherwise.*
- *next( ): Returns the next element in the sequence.*
- remove( ): *Removes from the collection the element returned by the most recent call to next(*). Throws an IllegalStateException if next has not yet been called, or if remove was already called since the most recent call to next.

HOW TO WORK WITH ELEMENT IN ITERATOR

```
Iterator<ElementType> iter = collection.iterator( );
while (iter.hasNext( )) {
ElementType variable = iter.next( );
 loopBody // may refer to ”variable” }
```

FAMILIAR WITH ITERATOR IN JAVA COLLECTION FRAME WORK

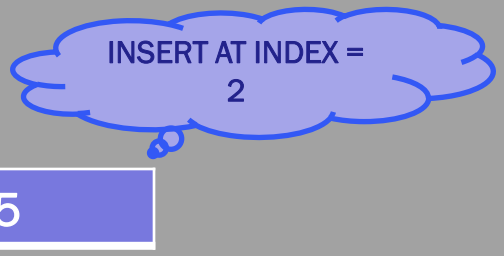
- *CREATE CLASS TO TEST ITERATOR*
- CREATE ARRAY 1D String[] students = new String[] {“Vo Tac Thien”,”Duong Minh Hoang”,”Luong Son Ba”};
- CONVERT ARRAY 1D TO ITERATOR
- TEST FUNCTIONS OF ITERATOR

Question

HOW TO INSERT NEW ELEMENT IN AN ARRAY ?

HOW TO INSERT NEW ELEMENT IN AN ARRAY ?

- ***FUNCTION*** *: INSERT NEW ELEMENT IN AN ARRAY*
- ***PARAMETER****: ARRAY, NEW ELEMENT, INDEX OF NEW ELEMENT IN ARRAY*
- ***RETURN****: AN ARRAY HAS LENGTH EQUAL OLD LENGTH PLUS 1 AND HAS NEW ELEMENT.*



Disscuss

ADVANCE AND DISADVANCE OF ARRAY ?

LIST

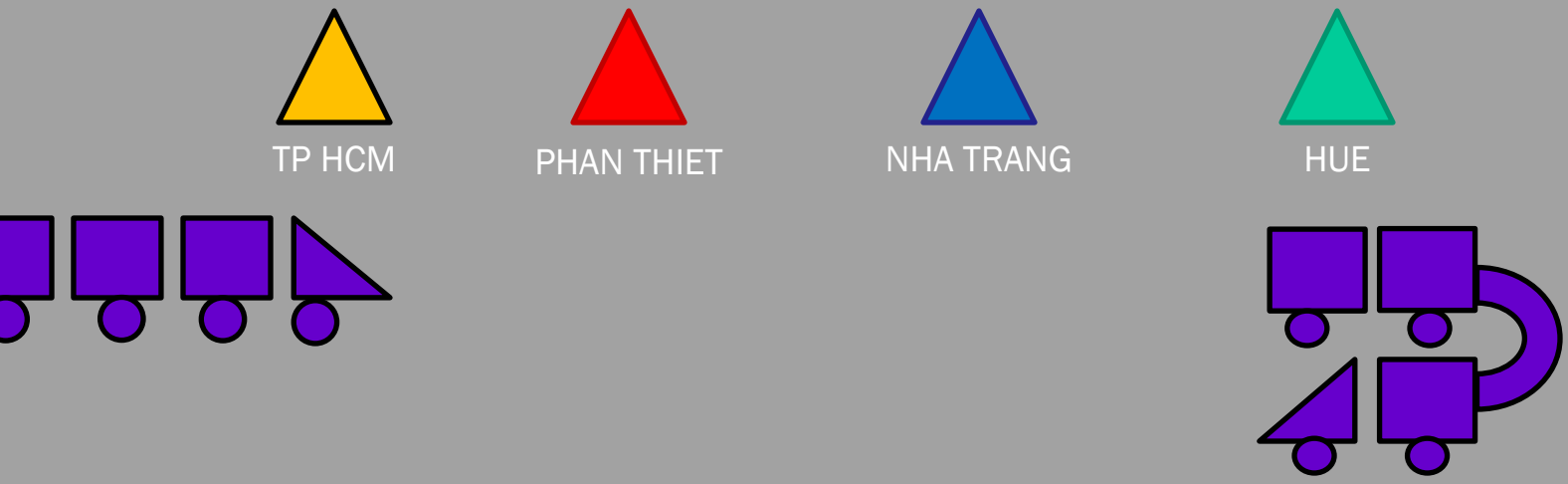
WHAT'S IS LIST?

- *List is sequential data structure*
- *Lists typically allow duplicate elements*
- *It easily add or remove element by position*

LIST

- *ITERATOR LIST*
- *ARRAY LIST*
- *LINKED LIST*

LIST ITERATOR



MAIN FUNCTIONS

- Boolean **hasNext()**
- Boolean **hasPrevious()**
- Object **next()**
- Int nextIndex()
- Object **previous()**
- Int previousIndex()
- Void set(Object o)

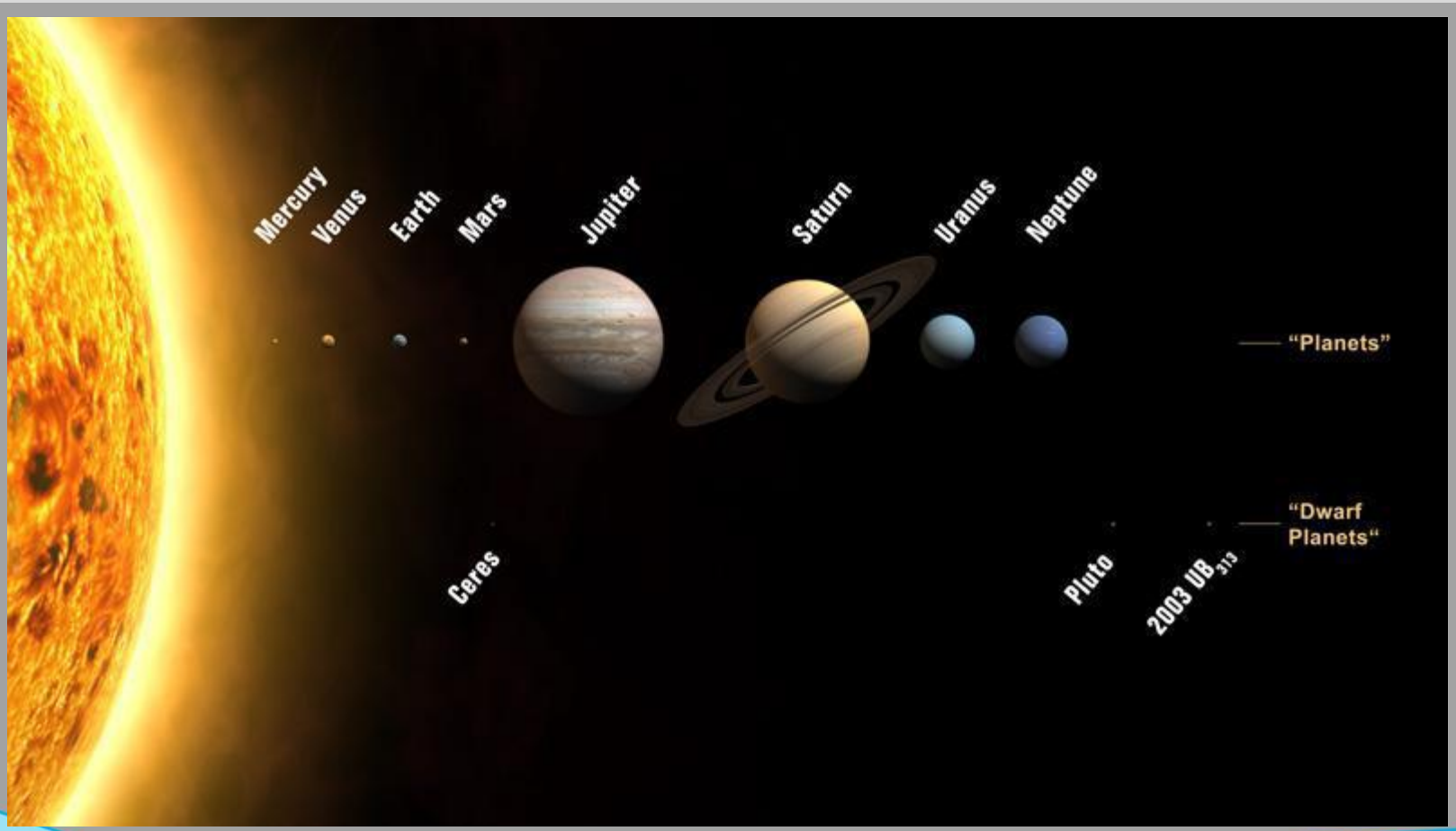
EXAMPLE OF LIST ITERATOR

```
int[ ] array = {1,3,5,7,9};
ListIterator li = Arrays.asList(array).listIterator();
```

1	3	5	7	9	11

```
li.hasNext() ; // return true
list.next() ; // return 1
list.next(); // return 3
list.previous(); //return 3
list.previous(); //return 1
```

Exercise: Compare planets



Exercise: Compare planets

REQUIREMENTS

1. *Tìm ngôi sao có diện tích lớn nhất*
2. *Tìm ngôi sao có khối lương lớn nhất*
3. *Tìm ngôi sao có chu kỳ lớn nhất*
4. *Tìm ra thông tin về ngôi sao nếu như biết tên tiếng Việt hoặc tiếng Anh của ngôi sao.*
5. *Cho ra thông tin ngẫu nhiên của 1 hành tinh(ngôi sao)*
6. *So sánh chu kỳ quay của Trái Đất và các hành tinh khác*

Exercise: Compare planets

TEST WITH DATA

Tên	Bán trục lớn	Bán kính	DT bề mặt	Thể tích	Khối lượng	KL riêng	Gia tốc	TĐ VT2	CK TQ	CK QĐ	Tốc độ	Tâm sai	ĐN QĐ [2]	ĐN trục	T_{bm}	Số vệ tinh	Vành đa
Đơn vị	10^9 km	10^3 km	10^9 km^2	10^{12} km^3	10^{24} kg	g/cm^3	m/s^2	km/s	ngày	năm	km/s		độ	độ	Độ K		
Sao Thủy[3][4]	0,058	2,440	0,075	0,061	0,330	5,427	3,70	4,25	58,646	0,241	47,87	0,206	7,0	0,01	440	0	không
Sao Kim[5][6]	0,108	6,052	0,46	0,928	4,869	5,243	8,87	10,36	243,686	0,615	35,02	0,007	3,39	2,64	730	0	không
Trái Đất[7][8]	0,150	6,378	0,51	1,083	5,974	5,515	9,78	11,19	0,997	1	29,78	0,016	1,58	23,44	287	1	không
Sao Hỏa[9][10]	0,228	3,402	0,145	0,164	0,642	3,934	3,69	5,03	1,026	1,881	24,08	0,093	1,85	25,19	210	2	không
Sao Mộc[11][12]	0,778	71,492	61,4	1,338	1899	1,326	23,12	59,54	0,414	11,87	13,05	0,048	1,30	3,13	152	67[13]	có
Sao Thổ[14][15]	1,427	60,268	42,7	746	568,46	0,687	8,96	35,49	0,444	29,45	9,64	0,054	2,49	26,73	134	62	có
Thiên Vương[16][17]	2,871	25,559	8,084	68,34	86,832	1,318	8,69	21,29	0,718	84,02	6,795	0,047	0,77	97,77	68	27	có
Hải Vương[18][19]	4,498	24,764	7,619	62,526	102,43	1,638	11	23,5	0,671	164,89	5,432	0,009	1,77	28,32	53	13	có

Exercise: Compare planets

HINT TO SOLUTION

```
public class HanhTinh {
        private String tenTV;
        private String tenTA;
        private double chuKy;
        private double dienTich;
        private double khoiLuong;

public HanhTinh(String tenTV, String tenTA, double dienTich, double khoiLuong,
double chuKy) {
        this.tenTV = tenTV;
        this.tenTA = tenTA;
        this.chuKy = chuKy;
        this.dienTich = dienTich;
        this.khoiLuong = khoiLuong;
}
```

Exercise: Compare planets

HINT TO SOLUTION

```
public class ListIterHanhTinh {
private ListIterator<HanhTinh> dsHT = null;
public ListIterHanhTinh(HanhTinh[] arrayHT){
dsHT = Arrays.asList(arrayHT).listIterator();
}
```

Exercise: Compare planets

PAY ATTENDTION

- *Lưu ý listIterator giống như con trỏ sau khi chạy 1 hàm sẽ đứng ở vị trí nào thì hàm thứ 2 sẽ chạy bắt đầu từ vị trí đó. Do đó cần đưa con trỏ về đầu trước khi thực hiện phương thức khác.*

WHAT'S ARRAYLIST

- *Array lists are created with an initial size. When this size is exceeded, the collection is automatically enlarged. When objects are removed, the array may be shrunk.*

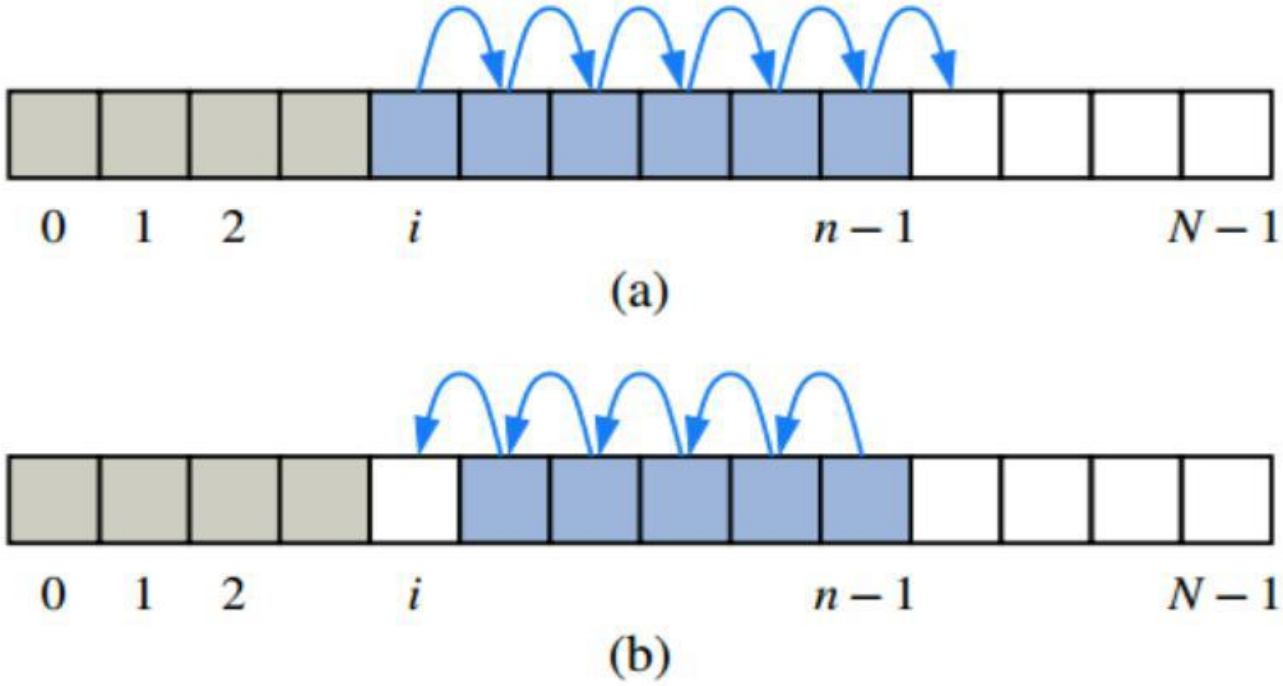


Figure 7.1: Array-based implementation of an array list that is storing n elements: (a) shifting up for an insertion at index i; (b) shifting down for a removal at index i.

MAIN FUNCTIONS AND PERFORMANCE OF ARRAYLIST STRUCTURE

Method	Running Time
size()	$O(1)$
isEmpty()	$O(1)$
get(i)	$O(1)$
set(i, e)	$O(1)$
add(i, e)	$O(n)$
remove(i)	$O(n)$

Table 7.1: Performance of an array list with n elements realized by a fixed-capacity array.

HOW TO BUILDING ARRAYLIST SIMPLE

READ, RUN AND UNDERSTAND

PAGE 260, 261

```
1  public class ArrayList<E> implements List<E> {
2    // instance variables
3    public static final int CAPACITY=16;     // default array capacity
4    private E[] data;                        // generic array used for storage
5    private int size = 0;                    // current number of elements
6    // constructors
7    public ArrayList( ) { this(CAPACITY); }  // constructs list with default capacity
8    public ArrayList(int capacity) {         // constructs list with given capacity
9      data = (E[ ]) new Object[capacity];    // safe cast; compiler may give warning
10   }
```

```
11   // public methods
12   /** Returns the number of elements in the array list. */
13   public int size( ) { return size; }
14   /** Returns whether the array list is empty. */
15   public boolean isEmpty( ) { return size == 0; }
16   /** Returns (but does not remove) the element at index i. */
17   public E get(int i) throws IndexOutOfBoundsException {
18     checkIndex(i, size);
19     return data[i];
20   }
21   /** Replaces the element at index i with e, and returns the replaced element. */
22   public E set(int i, E e) throws IndexOutOfBoundsException {
23     checkIndex(i, size);
24     E temp = data[i];
25     data[i] = e;
26     return temp;
27   }
28   /** Inserts element e to be at index i, shifting all subsequent elements later. */
29   public void add(int i, E e) throws IndexOutOfBoundsException,
30                                      IllegalStateException {
31     checkIndex(i, size + 1);
32     if (size == data.length)                  // not enough capacity
33       throw new IllegalStateException("Array is full");
34     for (int k=size-1; k >= i; k--)           // start by shifting rightmost
35       data[k+1] = data[k];
36     data[i] = e;                              // ready to place the new element
37     size++;
38   }
39   /** Removes/returns the element at index i, shifting subsequent elements earlier. */
40   public E remove(int i) throws IndexOutOfBoundsException {
41     checkIndex(i, size);
42     E temp = data[i];
43     for (int k=i; k < size-1; k++)            // shift elements to fill hole
44       data[k] = data[k+1];
45     data[size-1] = null;                      // help garbage collection
46     size--;
47     return temp;
48   }
49   // utility method
50   /** Checks whether the given index is in the range [0, n-1]. */
51   protected void checkIndex(int i, int n) throws IndexOutOfBoundsException {
52     if (i < 0 || i >= n)
53       throw new IndexOutOfBoundsException("Illegal index: " + i);
54   }
55 }
```

Code Fragment 7.3: An implementation of a simple ArrayList class with bounded capacity. (Continued from Code Fragment 7.2.)

DISCUSS:

WHEN USING ARRAY LIST?

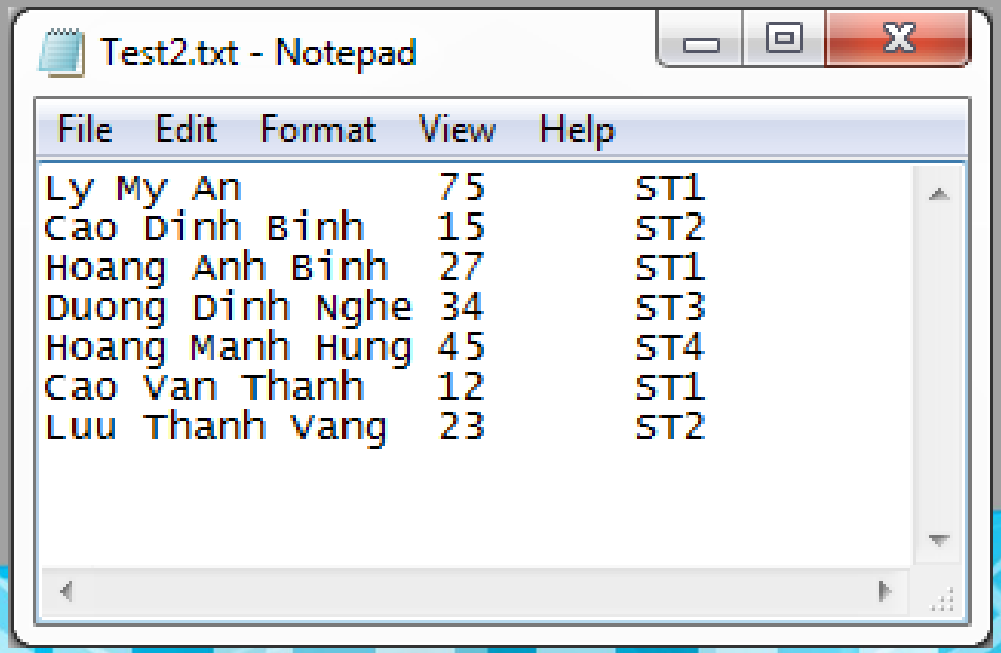
Exercise: Working with file
Rút thăm trúng thưởng

Create class FileTxtUtils : it has some functions which working with file.

1.Reading file .txt from url, result is array list of lines in .txt file

2.Checking that content of file contains string customer input.

3.Reading column in txt file with column index, separate symbol like "/ t".



Exercise: Working with file
Rút thăm trúng thưởng

1. *Tìm ra khách hàng có điểm thưởng cao nhất*
2. *Tìm ra N khách hàng ngẫu nhiên trúng giải thưởng.*
3. *In ra danh sách tên khách hàng tham gia rút thăm trúng thưởng, họ tên trùng nhau chỉ xét 1 lần.*
4. *In ra danh sách khách hàng trúng thưởng cho chương trình "Khuyến mãi lớn cuối năm" với 3 giải đặc biệt, 5 giải nhất, 10 giải nhì, 20 giải 3.*

Exercise: Working with file
Rút thăm trúng thưởng

1. *Tạo lớp FileUtil, chứa các phương thức đọc file, viết file*
2. *Tạo lớp PhieuDuThuong, có thông tin ten khach hàng, mã phiếu, mã siêu thị.*
3. *Tạo lớp QuanLyRutTham chứa danh sách các phiếu dự thưởng được load lên từ file .txt*

```
public static ArrayList load(String urlToFile){
ArrayList result = new ArrayList();
try{
File file = new File(urlToFile);
FileReader reader = new FileReader(file);
BufferedReader in = new BufferedReader(reader);
String name = in.readLine();
while(name != null){
result.add(name);
name= in.readLine();
}
}
catch(Exception e){

}

return result;
}
```