2110211 โครงสร้างข้อมูลเบื้องต้น

ArrayList & LinkedList

สมชาย ประสิทธิ์จูตระกูล

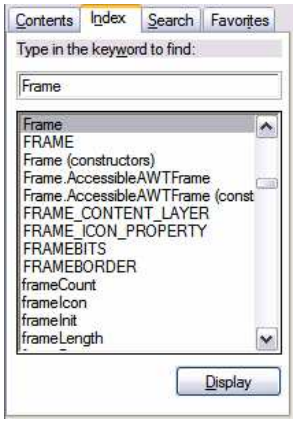
List

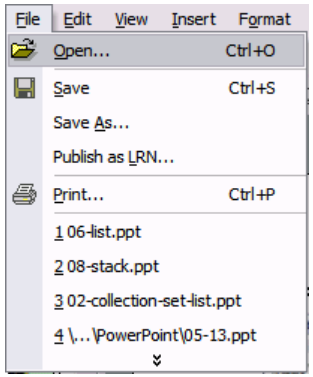
- List ก็เป็น Collection อย่างหนึ่ง
- เก็บแบบมีอันดับ ข้อมูลแต่ละตัวมีหมายเลขกำกับ

$$< a_0, a_1, a_2, ..., a_{n-1} >$$

```
public interface List extends Collection {
  public void add(int index, Object element);
  public void remove(int index);
  public Object get(int index);
  public void set(int index, Object element);
  public int indexOf(Object element);
}
```


List






```
<"SU", "MO", "TU", "WE", "TH", "FR", "SA">
```


สร้าง list ด้วยอาเรย์

- extends จาก ArrayCollection

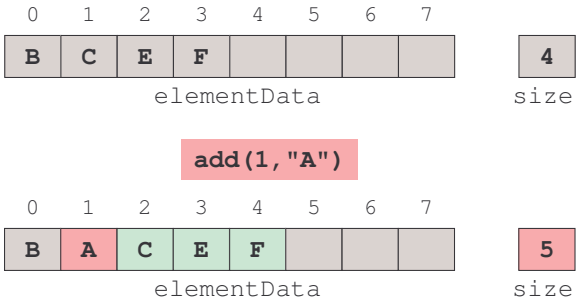
```
public class ArrayList extends ArrayCollection
                       implements List {
  public ArrayList() {}

  public Object get(int index) {
    return elementData[index];
  }                                    O(1)

  public void set(int index, Object element) {
    elementData[index] = element;
  }                                    O(1)

  public int indexOf(Object element) {
    for(int i=0; i<size; i++)
      if (elementData[i].equals(element)) return i;
    return -1;
  }                                    O(n)
  ...
```

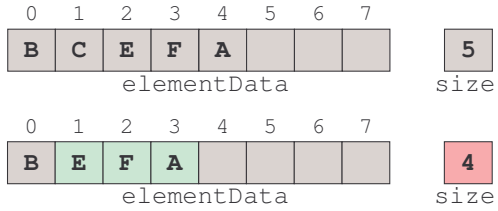
ArrayList : add(index, element)




```
public void add(int index, Object element) {
  ensureCapacity(size+1);
  for(int i=size; i>index; i--) {
    elementData[i] = elementData[i-1];
  }
  elementData[index] = element;
  size++;
}
```

add(0,e) ช้า add(size,e) เร็ว

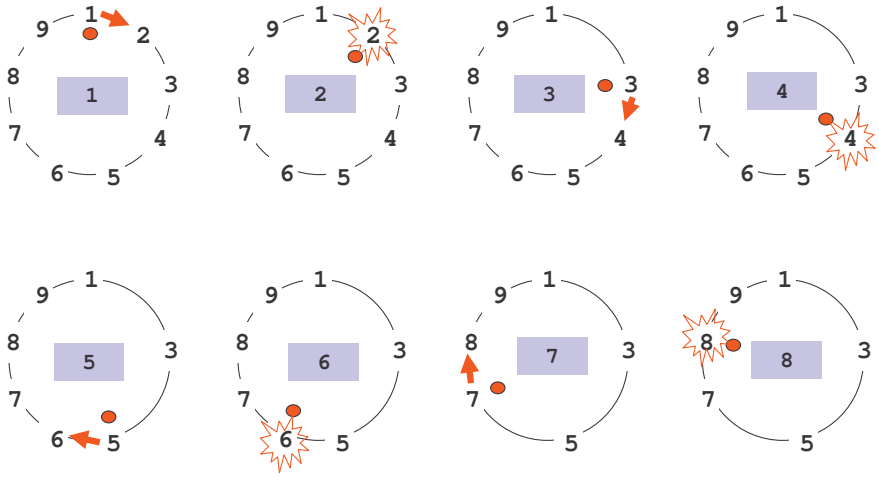
ArrayList : remove(index)



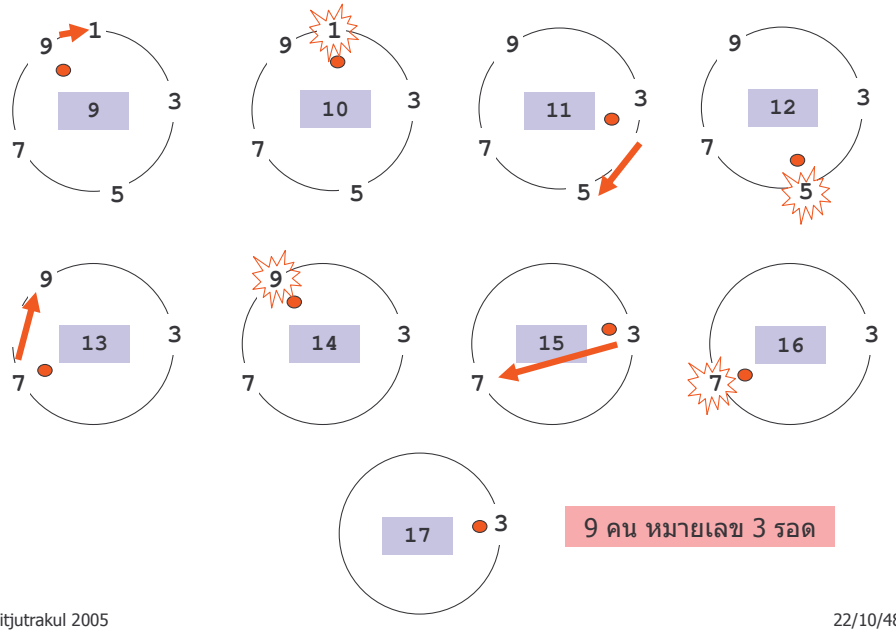

```
public void remove(int index) {
  for(int i=index+1; i<size; i++) {
    elementData[i-1] = elementData[i];
  }
  size--;
}
```

remove(0) ช้า remove(size-1) เร็ว

Example : Josephus Problem

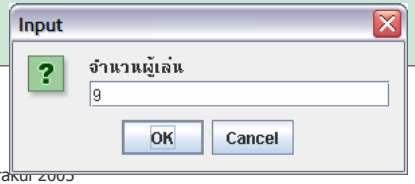


Example : Josephus Problem



Josephus Problem : n คน ใครรอด ?

```
public class Josephus {
  public static void main(String[] args) {
    String s = JOptionPane.showInputDialog(null, "จำนวนผู้เล่น");
    int n = Integer.parseInt(s);
    List list = new ArrayList();
    for(int i=1; i<=n; i++) list.add( i ); // autobox
    n = 1;
    while(list.size() > 1) {
      System.out.println(list.get(n));
      list.remove(n);
      n = (n + 1) % list.size();
    }
    JOptionPane.showMessageDialog(
       null, "ผู้ชนะคือ " + list.get(0));
  }
}
```



นอกเรื่อง : Autoboxing

- ภายใน ArrayList เราจองอาเรย์ของ Object
  - Object[] elementData;
- มีเมท็อด void add(Object e)
- แล้วทำไม Josephus เขียนแบบนี้ได้ ?
  - for(int i=1; i<=n; i++) list.add(i);
  - i เป็น int เพิ่มเข้าไปใน list ได้อย่างไร เขารับแต่ Object
- คำตอบ : Java 5 ทำ เปลี่ยน primitive data ให้เป็นออปเจคก์แบบ wrapper classes ให้อัตโนมัติ
  - int เปลี่ยนเป็นออปเจกต์ของ Integer
  - double เปลี่ยนเป็นออปเจกต์ของ Double
  - char เปลี่ยนเป็นออปเจกต์ของ Character
  - และอื่น ๆ

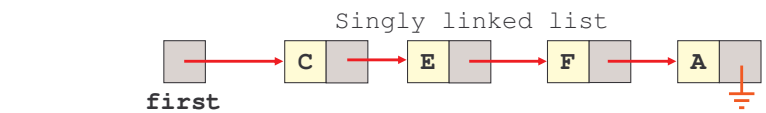
เรียกว่า autoboxing : จับ primitive data ห่อใส่ "กล่อง" ให้เป็นออปเจกต์โดยอัตโนมัติ

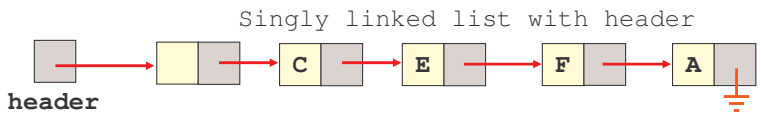
นอกเรื่อง : Autoboxing & Wrapper Classes

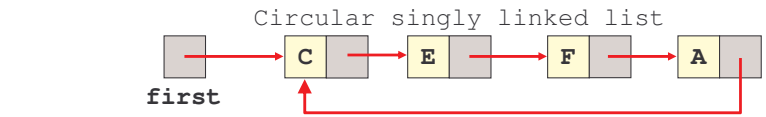
- เขียน list.add(2);
- javac เปลี่ยนให้เป็น list.add(new Integer(2)); syntactic sugar
- Integer เป็นคลาสที่ตัวออปเจกต์เก็บหนึ่ง int
  - เป็นวิธี "ห่อ" ให้ primitive int เป็นออปเจกต์
  - ให้บริการต่าง ๆ เกี่ยวกับเลขจำนวนเต็ม
- สร้าง Integer จาก int : ใช้ new Integer(i)
- ดึง int ออกจาก Integer : เรียก intValue()
- แต่ตอนเรียก list.get(n) จะได้ออปเจกต์ออกมา ไม่ใช่ int (ต้องการให้ได้ primitive ต้องใช้ Generics ของ Java 5 แล้วเขาจะทำ auto-unboxing ให้)

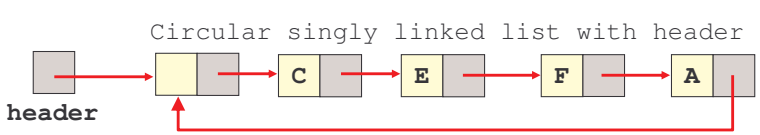
```
int i = 23;
Integer a = new Integer(i);
int j = a.intValue() + 1;
```


Singly Linked Lists

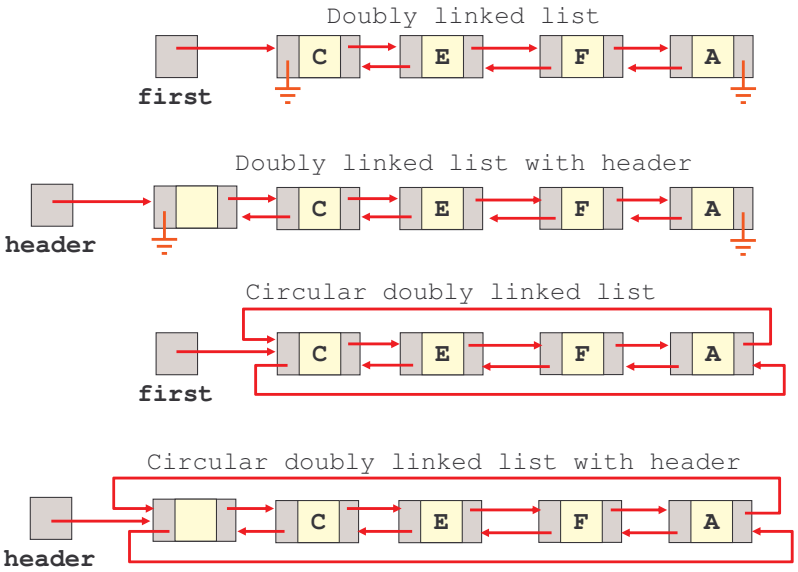








Doubly Linked Lists



Circular Doubly Linked List w/ Header

```
public class LinkedList implements List {

  static class DLinkedNode {
    Object element;
    DLinkedNode prev;
    DLinkedNode next;

    DLinkedNode(Object e, LinkedNode p, LinkedNode n) {
      this.element = e;
      this.prev = p
      this.next = n;
    }
  }

  DLinkedNode header;
  int size;

  ...
```


LinkedList

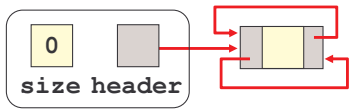
```
public class LinkedList implements List {
  ...

  DLinkedNode header;
  int size;

  public LinkedList() {
    header = new DLinkedNode(null, null, null);
    header.prev = header.next = header;
  }

  public boolean isEmpty() {
    return header.next == header;  // size == 0
  }

  ...
```

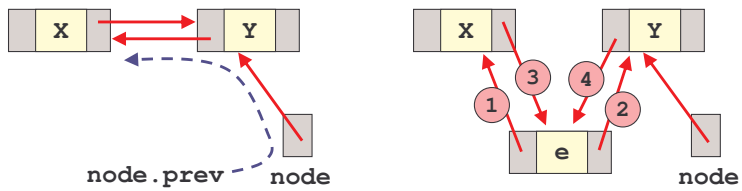


LinkedList : add(node, e)

```
public class LinkedList implements List {
  ...

  private void add(DLinkedNode node, Object e) {
    DLinkedNode newNode = new DLinkedNode(e,
                                          node.prev, node);
                                          (1)        (2)

    newNode.next.prev = newNode.prev.next = newNode;
    ++size;          (3)                  (4)
  }
  ...
```

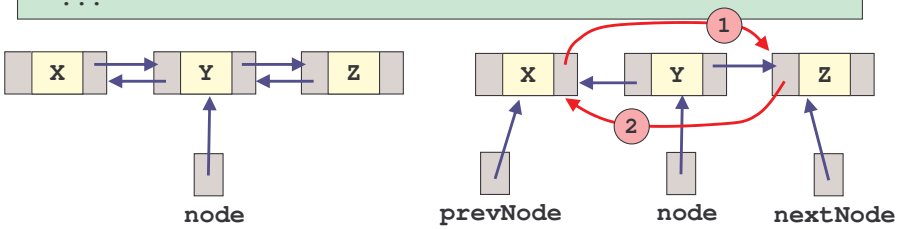


LinkedList : addFirst, addLast

```
public class LinkedList implements List {
  ...
  public void addFirst(Object e) {
    add(header.next, e); // before the 1st node
  }
  public void addLast(Object e) {
    add(header, e);   // before the header node
  }
  private void add(DLinkedNode node, Object e) {
    DLinkedNode newNode = new DLinkedNode(e,
                              node.prev, node);
    newNode.next.prev = newNode.prev.next = newNode;
    ++size;
  }
  ...
```


LinkedList : remove(node)

```
public class LinkedList implements List {
  ...
  private void remove(DLinkedNode node) {
    DLinkedNode prevNode = node.prev;
    DLinkedNode nextNode = node.next;
    prevNode.next = nextNode; (1)
    nextNode.prev = prevNode; (2)
    --size;
  }
  ...
```



LinkedList : removeFirst, removeLast

```
public class LinkedList implements List {
  ...
  public void removeFirst() {
    remove(header.next);
  }
  public void removeLast() {
    remove(header.prev);
  }
  private void remove(DLinkedNode node) {
    if (node != header) { // why ?
      DLinkedNode prevNode = node.prev;
      DLinkedNode nextNode = node.next;
      prevNode.next = nextNode;
      nextNode.prev = prevNode;
      --size;
    }
  }
  ...
```


LinkedList : add, remove, contains

```
public class LinkedList implements List {
  ...
  public void add(Object e) { addLast(e); }
  public void remove(Object e) { remove(nodeOf(e)); }
  public boolean contains(Object e) {
    return nodeOf(e) != header;
  }
  private DLinkedNode nodeOf(Object e) {
    DLinkedNode node = header.next;
    while(node != header && !node.element.equals(e)) {
      node = node.next;
    }
    return node;
  }
  ...
```

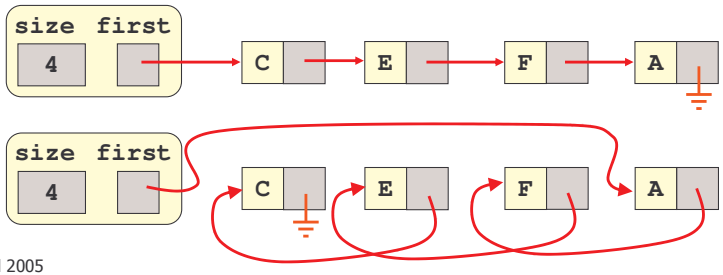
LinkedList : get(i), set(i, e)

```
public class LinkedList implements List {
  ...
  public Object get(int index) {
    return nodeAt(index).element;
  }
  public void set(int index, Object e) {
    nodeAt(index).element = e;
  }
  private DLinkedNode nodeAt(int index) {
    DLinkedNode node = header.next;
    for(int i=0; i<=index && node != header; i++) {
      node = node.next;
    }
    return node;
  }
  ...
```


การบ้าน : เขียนต่อที่เหลือ

```
public class LinkedList implements List {
  ...
  public int size() { ... }
  public void add(int index, Object e) { ... }
  public void remove(int index) { ... }
  public int indexOf(Object e) { ... }
  public void reverse() { ... }
}
```

reverse ย้ายเฉพาะ links ต่าง ๆ ภายใน list เพื่อให้ list มีรายการกลับลำดับกับที่เคยเป็น



การบ้าน

- เขียน linked list แต่คราวนี้ให้ extends จาก LinkedCollection (ที่ใช้ singly linked list with header)