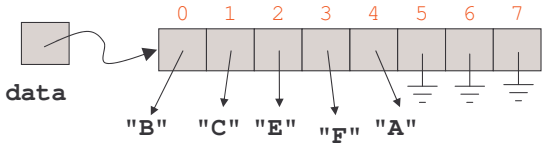
2110211 โครงสร้างข้อมูลเบื้องต้น

LinkedCollection

สมชาย ประสิทธิ์จูตระกูล

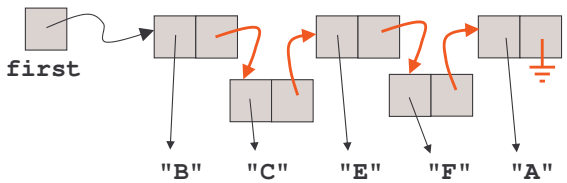
Array

- การอ้างอิงออปเจกต์อาศัย reference
- ใช้อาเรย์เก็บ references
- ช่องใดไม่อ้างอิงออปเจกต์ก็ใส่ null
- ใช้ index ของอาเรย์เป็นตัวบอกลำดับของข้อมูล
  - ตัวถัดจาก data[k] คือ data[k+1]
- ข้อดี : ง่าย หยิบ data[k] ใดๆ ได้ในเวลาเท่าๆ กัน
- ข้อเสีย : ต้องจองอาเรย์ไว้ก่อน ไม่พอต้องขยาย



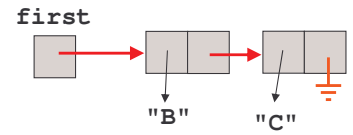
Linked List

- สร้าง "ก้อน" ออปเจกต์ที่ประกอบด้วย 2 fields
  - field หนึ่งเก็บ reference ของข้อมูล
  - field หนึ่งเก็บ refencence ของ "ก้อน" ถัดไป
    - มีตัวแปรเก็บ "ก้อน" แรก ถ้าไม่มีก้อนถัดไป ให้ใส่ null
- array รู้ตัวถัดไปด้วยการคำนวณ (k+1)
- linked รู้ตัวถัดไปด้วยการจำ
  - ข้อดี : จองก้อนข้อมูลเท่าที่จำเป็น
    เพิ่มลบข้อมูลไม่ต้องย้ายตำแหน่ง
  - ข้อเสีย : เปลือง references, ช้าเมื่อหยิบข้อมูลตัวที่ k



LinkedNode

```
class LinkedNode {
  Object element;
  LinkedNode next;

  LinkedNode(Object e, LinkedNode next) {
    this.element = e;
    this.next = next;
  }
}
```



LinkedCollection

- ย้าย LinkedNode เข้ามาเป็น private nested class
- มี field ชื่อ first เอาไว้เก็บ node แรก
- มี field ชื่อ size คอยจำจำนวนข้อมูล

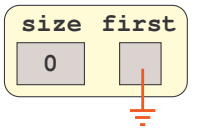
```
public class LinkedCollection implements Collection {

  static class LinkedNode {
    Object element;
    LinkedNode next;

    LinkedNode(Object e, LinkedNode next) {
      this.element = e;
      this.next = next;
    }
  }

  LinkedNode first;
  int size;

  ...
```


LinkedCollection : constructor, isEmpty, size

- collection ว่าง ๆ จะมี first เป็น null, size เป็น 0

```
public class LinkedCollection implements Collection {

  static class LinkedNode { ... }

  LinkedNode first;
  int size;

  public LinkedCollection() {}

  public boolean isEmpty() { return size == 0; }

  public int size() { return size; }

  ...
```

O(1)

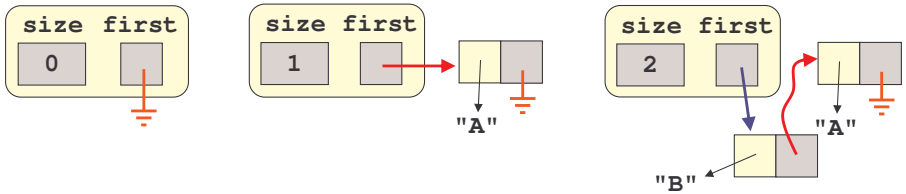


LinkedCollection : add

- สร้างก้อนข้อมูลใหม่ แล้วแทรกไว้ด้านหน้า

```
public class LinkedCollection implements Collection {

  static class LinkedNode { ... }

  LinkedNode first;
  int size;

  public void add(Object element) {
    first = new LinkedNode(element, first);
    ++size;
  }
  ...
```

O(1)

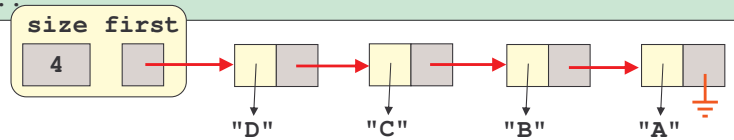


LinkedCollection : contains

- ค่อย ๆ วิ่งไล่เปรียบเทียบจาก first

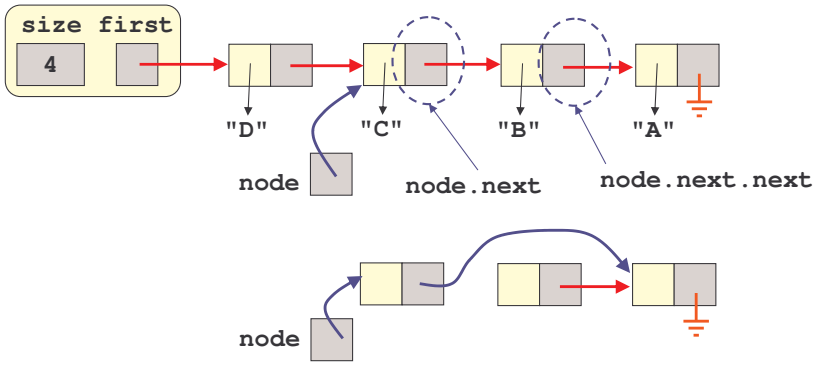
```
public class LinkedCollection implements Collection {
  ...
  LinkedNode first;
  int size;

  public boolean contains(Object e) {
    LinkedNode node = first;
    while( node != null ) {
      if (node.element.equals(e)) return true;
      node = node.next;
    }
    return false;
  }
  ...
```

O(n)



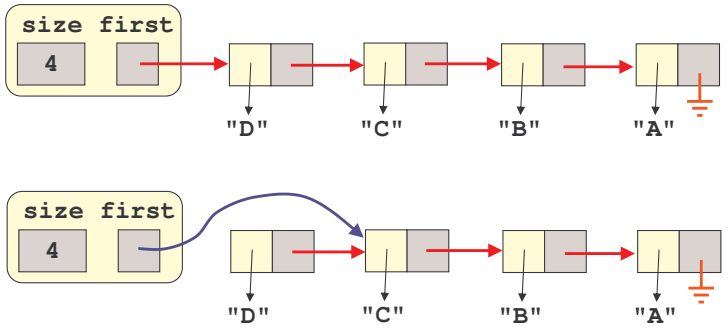
LinkedCollection : remove

- ต้องหาตัวก่อนหน้าตัวที่จะถูกลบ แล้วเปลี่ยน next
- ตัวอย่าง : remove("B")
  - ต้องหา node ที่ node.next.element มีค่าเป็น "B"
  - แล้วทำ node.next = node.next.next;



LinkedCollection : remove

- การลบ node แรก จะเป็นกรณีพิเศษ
- ตัวอย่าง : remove("D")
  - ต้องทำ first = first.next;



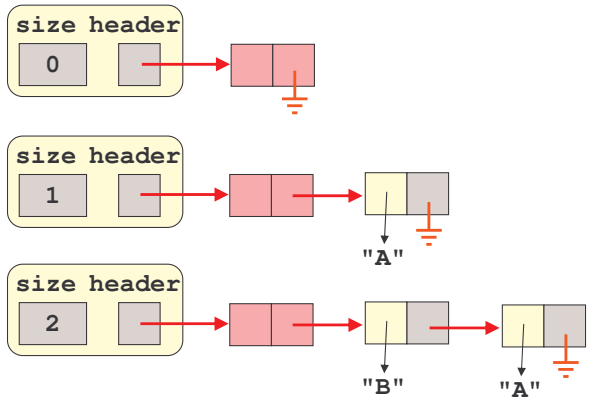
LinkedCollection : remove

```
public class LinkedCollection implements Collection {
  ...
  LinkedNode first;
  int size;

  public void remove(Object e) {
    if (first != null && first.element.equals(e)) {
      first = first.next;
    } else {
      LinkedNode node = first;
      while( node.next != null ) {
        if (node.next.element.equals(e)) {
           node.next = node.next.next;
           --size;
           break;
        }
        node = node.next;
      }
    }
  }
  ...
```

O(n)

Linked with Header

- remove น่ารำคาญที่ต้องมีกรณีพิเศษ
- ห้ามให้ลบ node แรก ก็ไม่เกิดกรณีพิเศษ
- ยอมให้ node แรกไม่เก็บข้อมูล เรียกว่า header



LinkedCollection with Header

```
public class LinkedCollection implements Collection {
  LinkedNode first;
  int size;

  public LinkedCollection() { }

  public void add(Object e) {
    first = new LinkedNode(e, first);
    ++size;
  }
```

w/o header

```
public class LinkedCollection implements Collection {
  LinkedNode header;
  int size;

  public LinkedCollection() {
    header = new LinkedNode(null, null);
  }

  public void add(Object e) {
    header.next = new LinkedNode(e, header.next);
    ++size;
  }
```

w/ header

LinkedCollection with Header : contains

```
public class LinkedCollection implements Collection {
  ...
  public boolean contains(Object e) {
    LinkedNode node = first;
    while( node != null ) {
      if (node.element.equals(e)) return true;
      node = node.next;
    }
    return false;
  }
```

w/o header

```
public class LinkedCollection implements Collection {
  ...
  public boolean contains(Object e) {
    LinkedNode node = header.next;
    while( node != null ) {
      if (node.element.equals(e)) return true;
      node = node.next;
    }
    return false;
  }
```

w/ header

LinkedCollection with Header : remove

```
public class LinkedCollection implements Collection {
  ...
  public void remove(Object e) {
    if (first != null && first.element.equals(e)) {
      first = first.next;
    } else {
      LinkedNode node = header;  // <---
      while( node.next != null ) {
        if (node.next.element.equals(e)) {
           node.next = node.next.next;
           --size;
           break;
        }
        node = node.next;
      }
    }
  }
  ...
```

w/ header

การบ้าน

- ถ้าเปลี่ยน node = node.next เป็น node.next = node.next.next; อะไรจะเกิดขึ้น

```
public class LinkedCollection implements Collection {
  ...
  LinkedNode first;
  int size;

  public boolean contains(Object e) {
    LinkedNode node = first;
    while( node != null ) {
      if (node.element.equals(e)) return true;
      node.next = node.next.next;  // <----
    }
    return false;
  }
  ...
```

การบ้าน

- เช่นเดียวกับ ArrayCollection เราก็จะไม่ให้ LinkedCollection เก็บข้อมูลที่มีค่า null
- จงปรับปรุงให้สร้างความมั่นใจว่า LinkedCollection จะไม่เก็บ null

การบ้าน

- ในคลาส LinkedCollection
  - เขียน toArray(), toString() และ equals(Object o) เหมือนกับการบ้านใน ArrayCollection

```
public class LinkedCollection implements Collection {
  ...
  public Object[] toArray() {
    ...
  }
  public String toString() {
    ...
  }
  public boolean equals(Object e) {
    ...
  }
}
```


การบ้าน

- ลองปิด sheet ให้หมด แล้วเขียนคลาส LinkedCollection ด้วยตนเองให้ครบ

```
public interface Collection {
  public void add(Object element);
  public void remove(Object element);
  public boolean isEmpty();
  public boolean contains(Object element);
  public int size();
}
```


การบ้าน : การสร้าง Set

- Set ก็เป็น collection อย่างหนึ่ง ต่างกันแค่ห้ามมีตัวซ้ำ

```
public interface Set extends Collection {
  /**
   * add a new element without duplication.
   *
   * @param element a new element
   */
  public void add(Object element);
}
```

การบ้าน : สร้าง ArraySet จาก ArrayCollection

```
public class ArraySet extends ArrayCollection
                      implements Set {
  /**
   * add a new element without duplication
   * @param element a new element
   * @throws IllegalArgumentException if element is null
   **/
  public void add(Object element) {


   // add your code here


  }
}
```


การบ้าน : สร้าง LinkedSet จาก LinkedCollection

```
public class LinkedSet extends LinkedCollection
                       implements Set {
  /**
   * add a new element without duplication
   * @param element a new element
   * @throws IllegalArgumentException if element is null
   **/
  public void add(Object element) {


   // add your code here


  }
}
```



```
public class UniqueWords {
  public static void main(String[] args) throws Exception {
    String fn = "thai.txt";
    BufferedReader fi = new BufferedReader(new FileReader(fn));
    String line;
    Set words = new ArraySet();
    while ((line = fi.readLine()) != null) {
      BreakIterator boundary =
        BreakIterator.getWordInstance(new Locale("th"));
      boundary.setText(line);
      int start = boundary.first();
      int end = boundary.next();

      while (end != BreakIterator.DONE) {
        String word = line.substring(start, end).trim();
        if (!word.equals("")) {
          words.add(line.substring(start, end));
        }
        start = end;
        end = boundary.next();
      }
    }
    fi.close();
    System.out.println(words);
  }
}
```