การเก็บข้อมูลด้วยการโยง

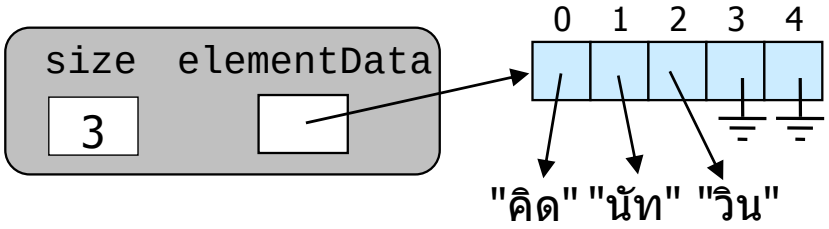
(LinkedCollection)

หัวข้อ

- การสร้างคอลเล็กชันด้วยการโยงข้อมูล
- การโยงแบบไม่มีและมีปมหัว
- การสร้างเซตด้วยคอลเล็กชัน

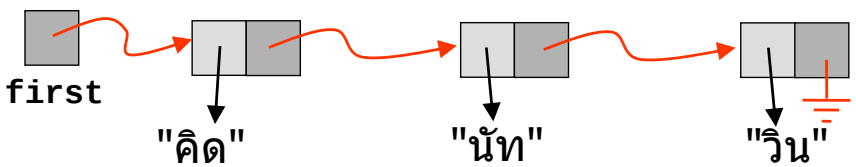
การเก็บข้อมูลด้วยอาเรย์

- แต่ละช่องเก็บ reference ไปยังออปเจกต์
- ช่องใดไม่เก็บข้อมูล ก็เก็บ null
- ข้อดี : เข้าถึง elementData[k] ใดๆ ได้ง่ายและรวดเร็ว
- ข้อเสีย : ต้องจองอาเรย์ไว้ก่อน ไม่พอต้องขยาย



การเก็บข้อมูลด้วยการโยง

- เก็บข้อมูลไว้ตาม "ปม" ข้อมูล
- เชื่อมโยงปมต่าง ๆ ให้ถึงกัน
- แต่ละปมเก็บ
  - reference ไปยังข้อมูล
  - reference ไปยังปมถัดไป (ไม่มีปมถัดไปเก็บ null)
- มีตัวแปรเก็บปมแรก
- ข้อดี : จองปมข้อมูลเท่าที่จำเป็น
- ข้อเสีย : เปลืองเนื้อที่สำหรับเก็บการโยง



การสร้างคอลเลกชันด้วยการโยง

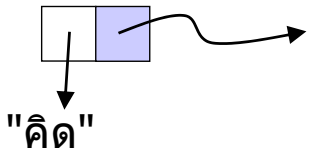
```
public interface Collection {
  public void add(Object element);
  public void remove(Object element);
  public boolean isEmpty();
  public boolean contains(Object element);
  public int size();
}
```

```
public class LinkedCollection implements Collection {
  public LinkedCollection() { ... }
  public void add(Object element) { ... }
  public void remove(Object element) { ... }
  public boolean isEmpty() { ... }
  public boolean contains(Object element) { ... }
  public int size() { ... }
}
```


LinkedNode : โครงสร้างของปม

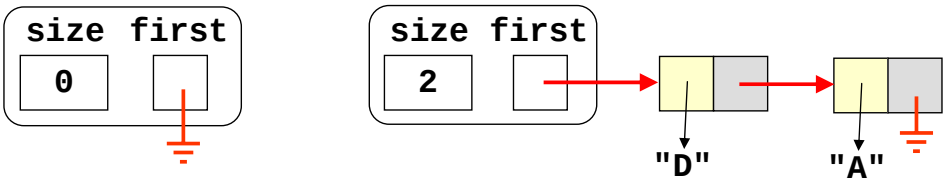
```
public class LinkedCollection implements Collection {
  private static class LinkedNode {
    Object element;
    LinkedNode next;
    LinkedNode(Object e, LinkedNode next) {
      this.element = e;
      this.next = next;
    }
  }
  ...
```

static inner class



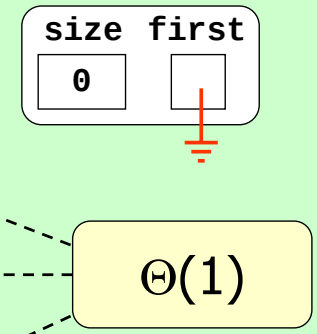
LinkedCollection : โครงสร้างการเก็บข้อมูล

```
public class LinkedCollection implements Collection {
  private static class LinkedNode { ... }

  private LinkedNode first;
  private int size;

  public LinkedCollection() {
    first = null;
    size = 0;
  }
  ...
```



isEmpty, size

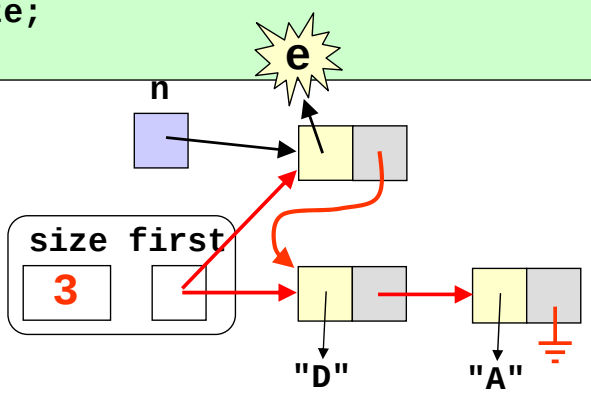
```
public class LinkedCollection implements Collection {
  private static class LinkedNode { ... }

  private LinkedNode first;
  private int size;

  public LinkedCollection() {}

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



add : การเพิ่มข้อมูล

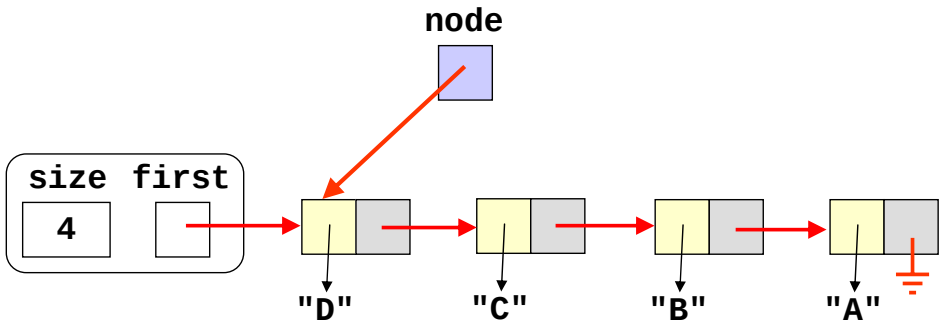
- สร้างปมข้อมูลใหม่ แล้วแทรกไว้ด้านหน้า

```
public class LinkedCollection implements Collection {
  private LinkedNode first;
  private int size;
  ...
  public void add(Object e) {

    first = new LinkedNode(e, first);

    ++size;
  }
```



Θ(1)

การเลื่อนไปทีละปมตามการโยง




```
LinkedNode node = first;
node = first.next;
node = node.next;
node = node.next;
node = node.next;
```

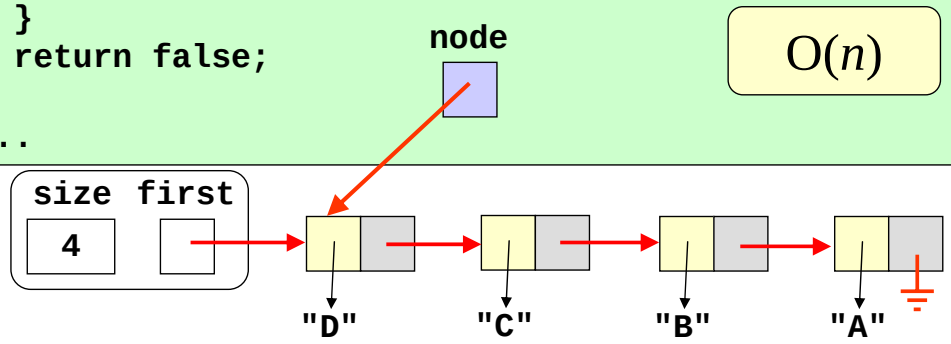
การเลื่อนไปยังปมถัดไป

contains : การค้นข้อมูล

- ค่อย ๆ วิ่งไล่เปรียบเทียบจาก first

```
public class LinkedCollection implements Collection {
  private LinkedNode first;
  private 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;
  }
  ...
```

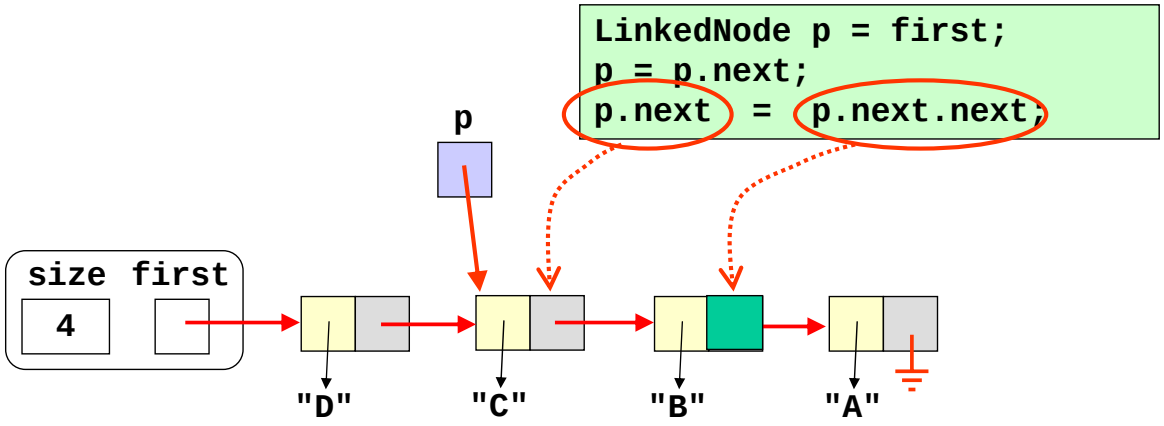
$O(n)$



remove : การลบข้อมูล

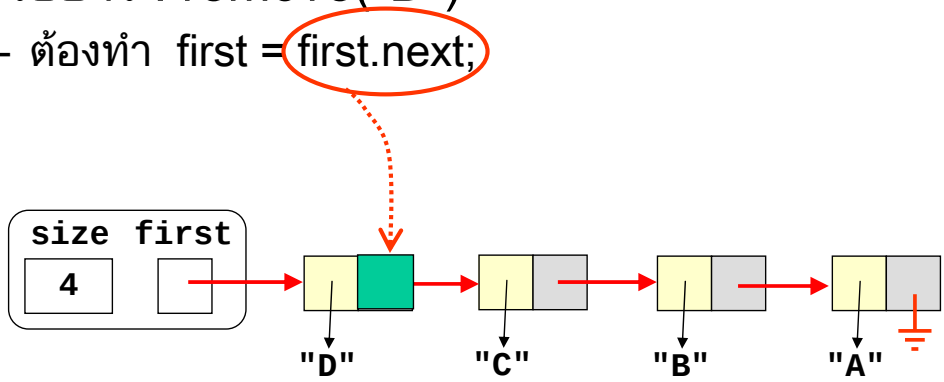
- ต้องการลบปมใดออก ต้องรู้ตำแหน่งของปมก่อนหน้า
  - ต้องการลบปม "B", หาปมที่ปมถัดไปเก็บ "B"
- เปลี่ยนตัวโยงให้ข้ามตัวที่ต้องการลบ

```
LinkedNode p = first;
p = p.next;
p.next = p.next.next;
```



remove : การลบข้อมูลที่ปมแรก

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



remove : การลบข้อมูล

```
public class LinkedCollection implements Collection {
  private LinkedNode first;
  private int size;
  ...
  public void remove(Object e) {
    if (first == null) return;
    if (first.element.equals(e)) {
      first = first.next; --size;
    } else {
      LinkedNode p = first;
      while( p.next != null &&
             ! p.next.element.equals(e)) {
        p = p.next;
      }
      if (p.next != null) {
        p.next = p.next.next; --size;
      }
    }
  }
```

คอลเลกชันว่าง

ลบปมแรก

ค้นให้พบ

ค้นพบ ก็ลบ

$O(n)$

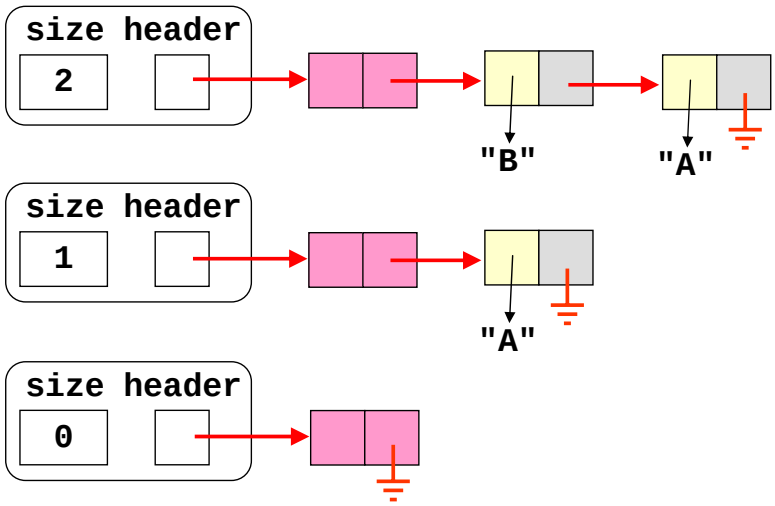
การลบมีหลายกรณี

- remove น่ารำคาญที่ต้องมีกรณีพิเศษ

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


การโยงแบบมีปมหัว (header)

- ต้องมี "ปมหัว" เป็นปมแรก
- ห้ามลบปมหัว
- ปมหัวนี้ไม่เก็บข้อมูล



การโยงแบบมีปมหัว

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

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

ไม่มีปมหัว

```
public class LinkedCollection implements Collection {
  private LinkedNode header
                    = new LinkedNode(null, null);
  private int size;

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

มีปมหัว

การโยงแบบมีปมหัว : 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;
  }
```

ไม่มีปมหัว

```
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;
  }
```

มีปมหัว

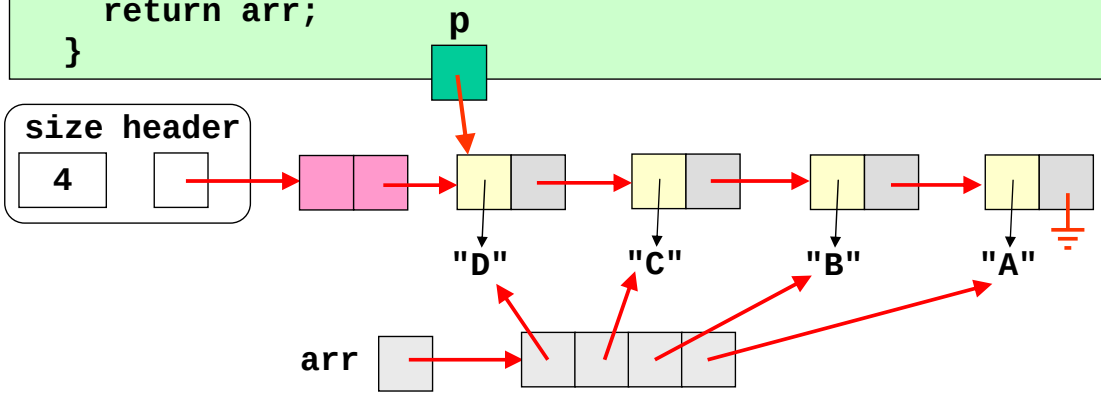
การโยงแบบมีปมหัว : remove

```
public class LinkedCollection implements Collection {
  private LinkedNode header = new LinkedNode(null,null);
  private int size;
  ...
  public void remove(Object e) {
    if (first == null) return;
    if (first.element.equals(e)) {
      first = first.next; --size;
    } else {
      LinkedNode p = header;
      while( p.next != null &&
             ! p.next.element.equals(e)) {
        p = p.next;
      }
      if (p.next != null) {
        p.next = p.next.next; --size;
      }
    }
  }
```


บริการเสริม : toArray

```
  public Object[] toArray() {
    Object[] arr = new Object[size];
    LinkedNode p = header.next;
    int k = 0;
    while (p != null) {
      arr[k++] = p.element;
      p = p.next;
    }
    return arr;
  }
```

$\Theta(n)$



Set คือ Collection ที่ไม่เก็บตัวซ้ำ

- Set ก็เป็น collection อย่างหนึ่ง ที่ห้ามเก็บตัวซ้ำ

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


การสร้างเซตด้วยคอลเลกชัน : inheritance

```
public class ArraySet extends ArrayCollection
                      implements Set {
  public void add(Object element) {
    if (!contains(element)) {
      add(element);
    }
  }
}
```

```
public class LinkedSet extends LinkedCollection
                       implements Set {
  public void add(Object element) {
    if (!contains(element)) {
      add(element);
    }
  }
}
```

การสร้างเซตด้วยคอลเลกชัน : composition

```
public class LinkedSet implements Set {
  private Collection c;
  public LinkedSet() {
    c = new LinkedCollection();
  }
  public boolean isEmpty() {
    return c.isEmpty();
  }
  public int size() {
    return c.size();
  }
  public boolean contains(Object e) {
    return c.contains(e);
  }
  public void remove(Object e) {
    return c.remove(e);
  }
  public void add(Object e) {
    if (!c.contains(e)) c.add(e);
  }
}
```


สรุป

- ข้อมูลเก็บในปมข้อมูล
- นำปมข้อมูลมาโยง
- โยงแบบไม่มีปมหัว มักมีกรณีพิเศษที่ต้องจัดการ
- โยงแบบมีปมหัว
  - เพิ่มความซับซ้อนในการจัดเก็บ
  - แต่ลดความซับซ้อนในการจัดการ