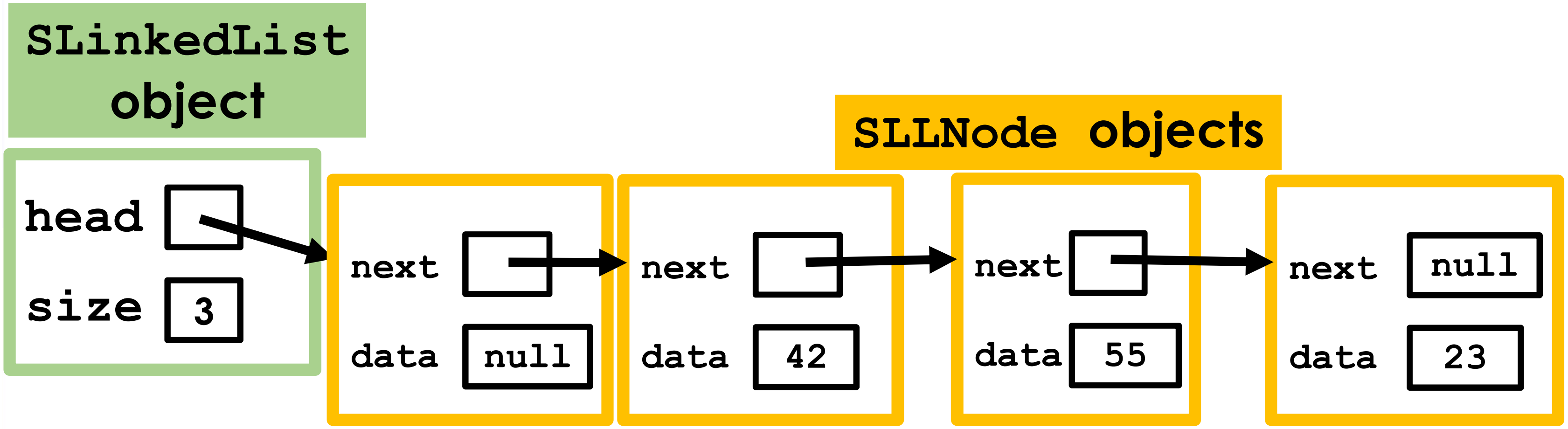
Linked List Implementation: Concept Challenge

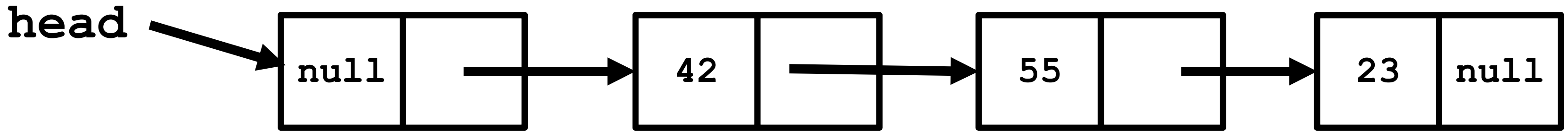
Concept Challenge: Procedure

- **Pause** Try to solve the problem yourself
- **Discuss** with other learners (if you can)
- **Watch** the UC San Diego learners video
- **Answer** the question again
- **Confirm** your understanding with our explanation

Implementing a Singly Linked List in Java



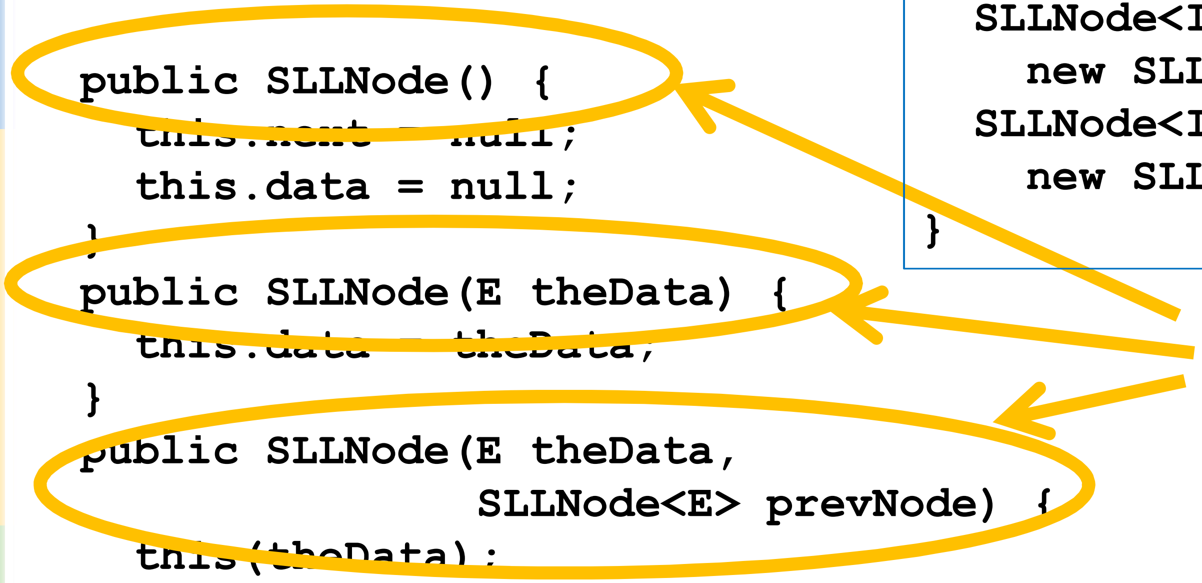
Implementing a Singly Linked List in Java



```
class SLLNode<E> {
  SLLNode<E> next;
  E data;

  public SLLNode(E theData) {
    this.data = theData;
  }
  public SLLNode(E theData,
                SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
}
```

```
public static void main(String[] args)
{
  SLLNode<Integer> n0 =
    new SLLNode<Integer>();
  SLLNode<Integer> n1=
    new SLLNode(1,n0);
  SLLNode<Integer> n2 =
    new SLLNode(2,n0);
}
```



```
class SLLNode<E> {
  SLLNode<E> next;
  E data;

  public SLLNode() {
    this.next = null;
    this.data = null;
  }
  public SLLNode(E theData) {
    this.data = theData;
  }
  public SLLNode(E theData,
                 SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
}
```

```
public static void main(String[] args)
{
  SLLNode<Integer> n0 =
    new SLLNode<Integer>();
  SLLNode<Integer> n1=
    new SLLNode(1,n0);
  SLLNode<Integer> n2 =
    new SLLNode(2,n0);
}
```

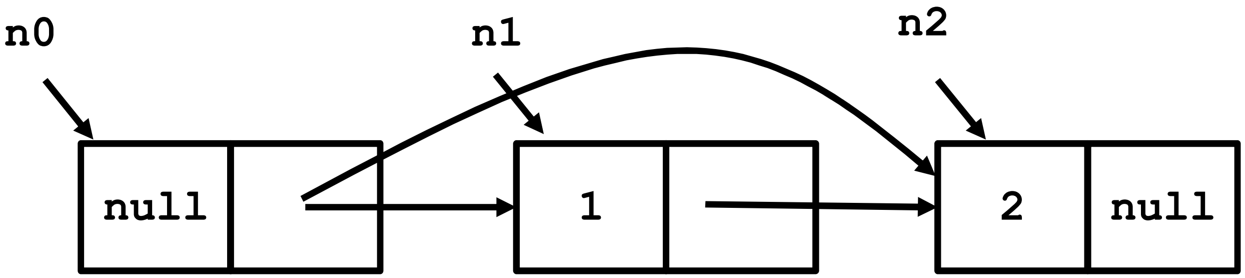
```
class SLLNode<E> {
  SLLNode<E> next;
  E data;

  public SLLNode() {
    this.next = null;
    this.data = null;
  }
  public SLLNode(E theData) {
    this.data = theData;
  }
  public SLLNode(E theData,
               SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
}
```

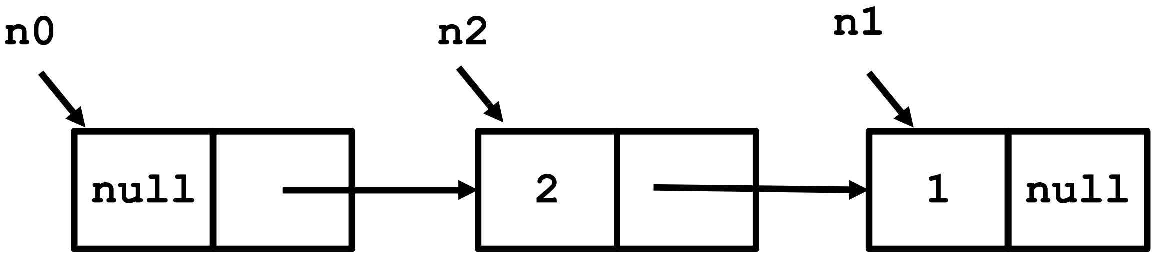
```
public static void main(String[] args)
{
  SLLNode<Integer> n0 =
    new SLLNode<Integer>();
  SLLNode<Integer> n1=
    new SLLNode(1,n0);
  SLLNode<Integer> n2 =
    new SLLNode(2,n0);
}
```

What does the list of nodes look like at the end of main?

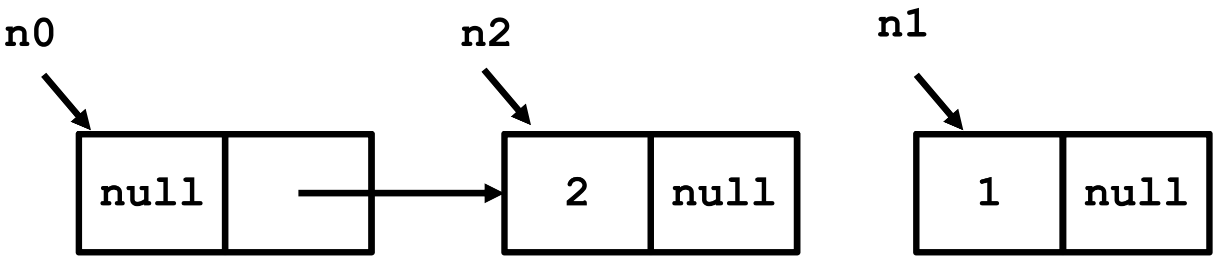
A



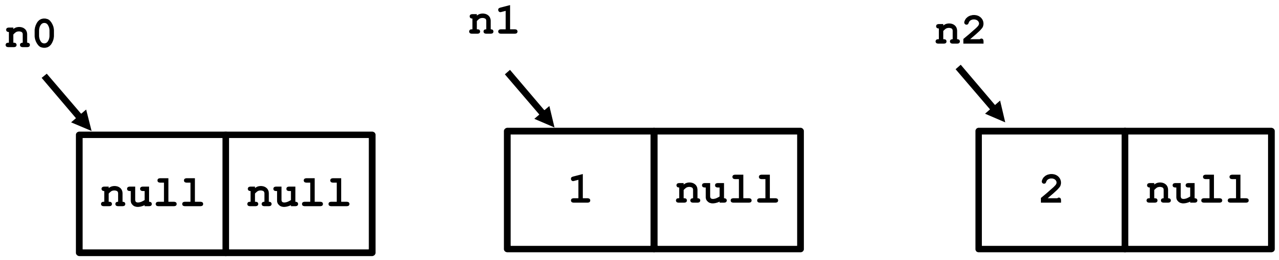
B



C



D



E Null pointer exception

Learner video here

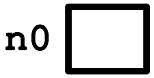
```
class SLLNode<E> {
  SLLNode<E> next;
  E data;

  public SLLNode() {
    this.next = null;
    this.data = nul
  }
  public SLLNo
    this.data
  }
  public SLLNode(E theData,
                 SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
}
```

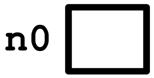
```
public static void main(String[] args)
{
  SLLNode<Integer> n0 =
    new SLLNode<Integer>();
  SLLNo          ger> n1=
                  (1,n0);
                 er> n2 =
              Node(2,n0);
```

Let's draw the diagram one line at a time

```
SLLNode<Integer> n0 = new SLLNode<Integer>();
```



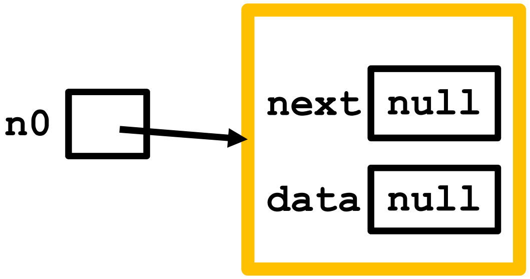
```
SLLNode<Integer> n0 = new SLLNode<Integer>();
```



No-arg constructor

```
public SLLNode() {
   this.next = null;
   this.data = null;
}
```

```
SLLNode<Integer> n0 = new SLLNode<Integer>();
```

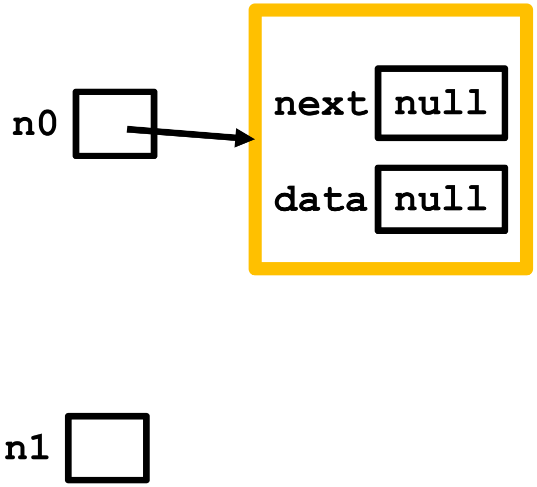


No-arg constructor

```
public SLLNode() {
   this.next = null;
   this.data = null;
}
```

```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
                         SLLNode<E> prevNode) {
      this(theData);
      this.next = prevNode.next;
      prevNode.next = this;
   }
```



n0

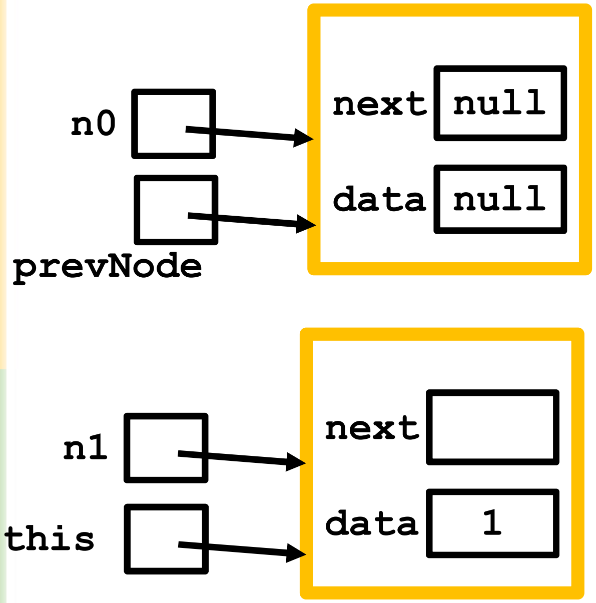
next null

data null

n1

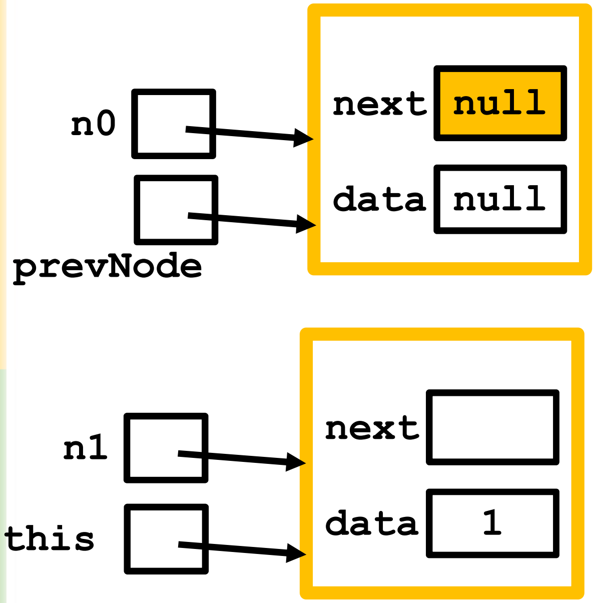
```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
                  SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```



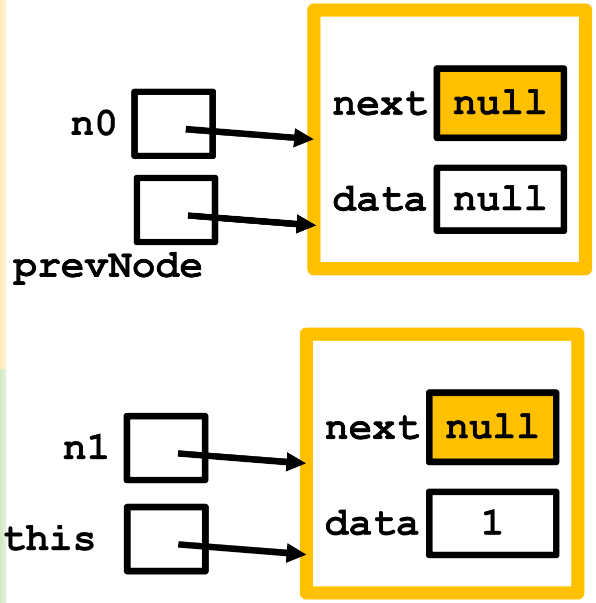
```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
                         SLLNode<E> prevNode) {
     this(theData);
     this.next = prevNode.next;
     prevNode.next = this;
   }
```



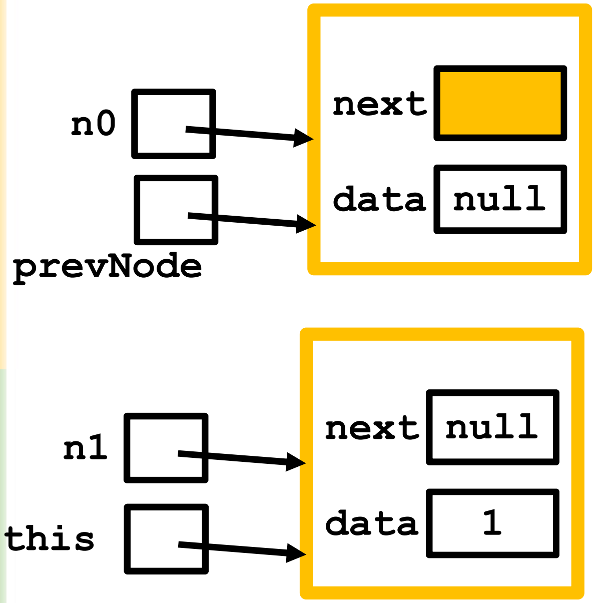
```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
               SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```



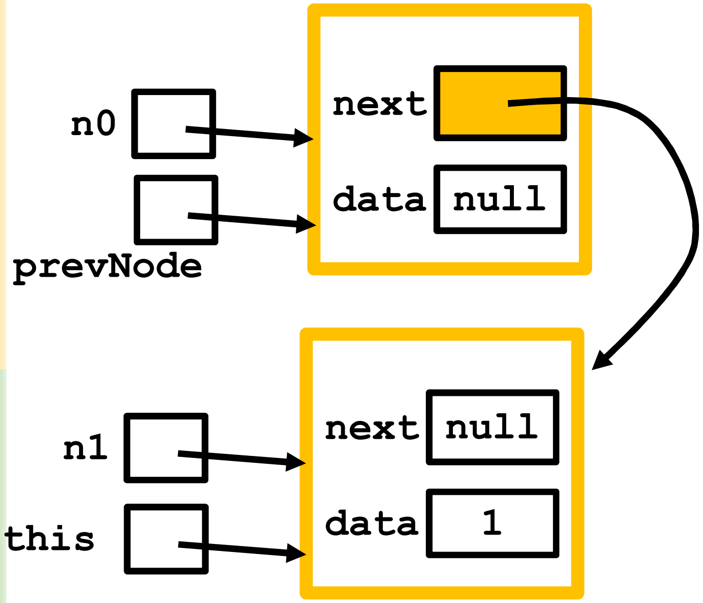
```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
                 SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```

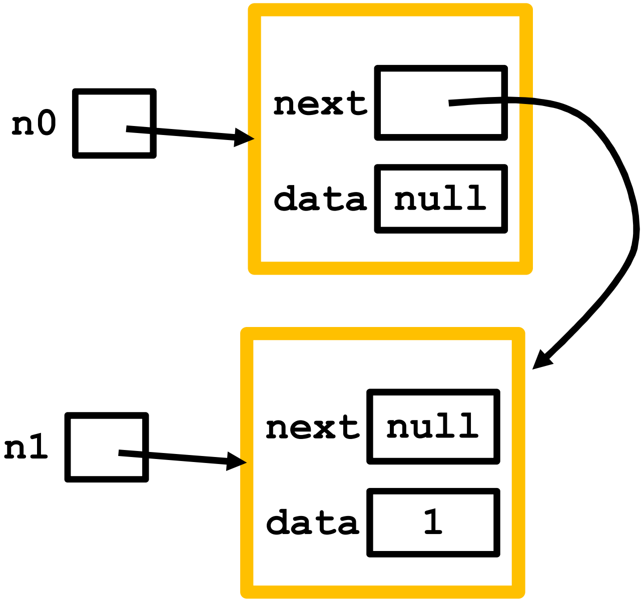


```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```

```
public SLLNode(E theData,
                  SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
 }
```

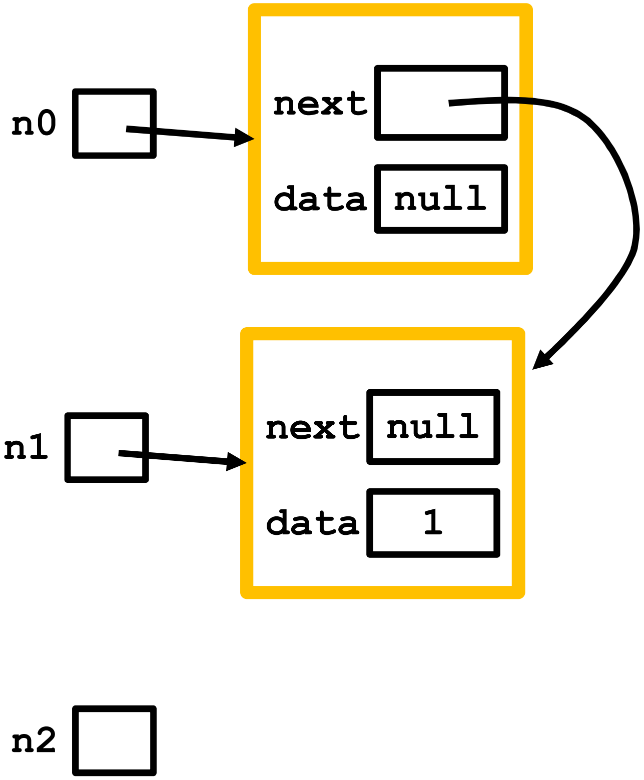


```
SLLNode<Integer> n1 = new SLLNode(1,n0);
```



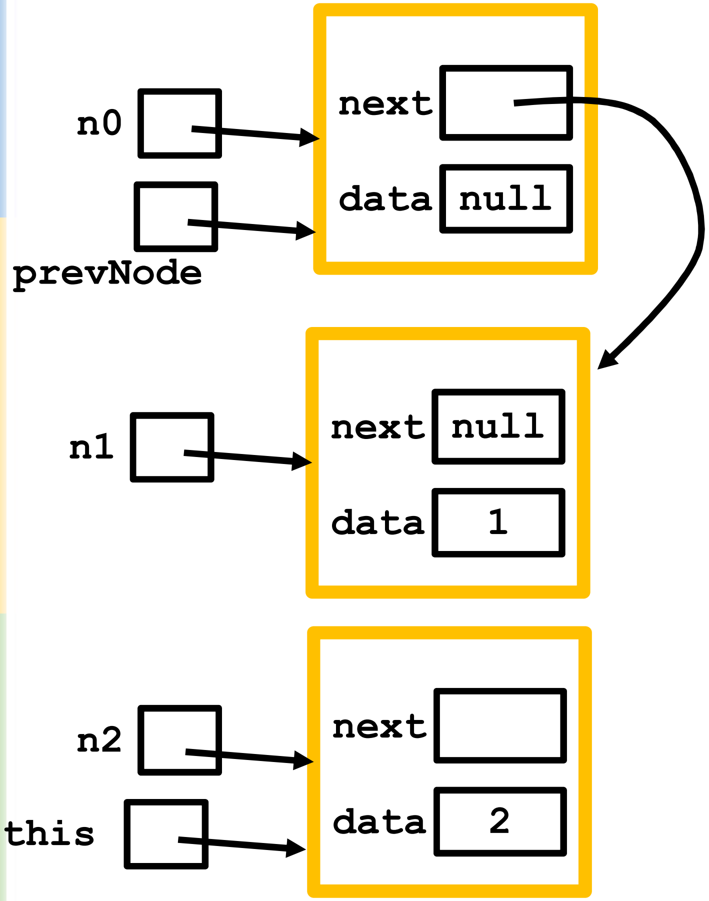
```
public SLLNode(E theData,
                           SLLNode<E> prevNode) {
      this(theData);
      this.next = prevNode.next;
      prevNode.next = this;
   }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```



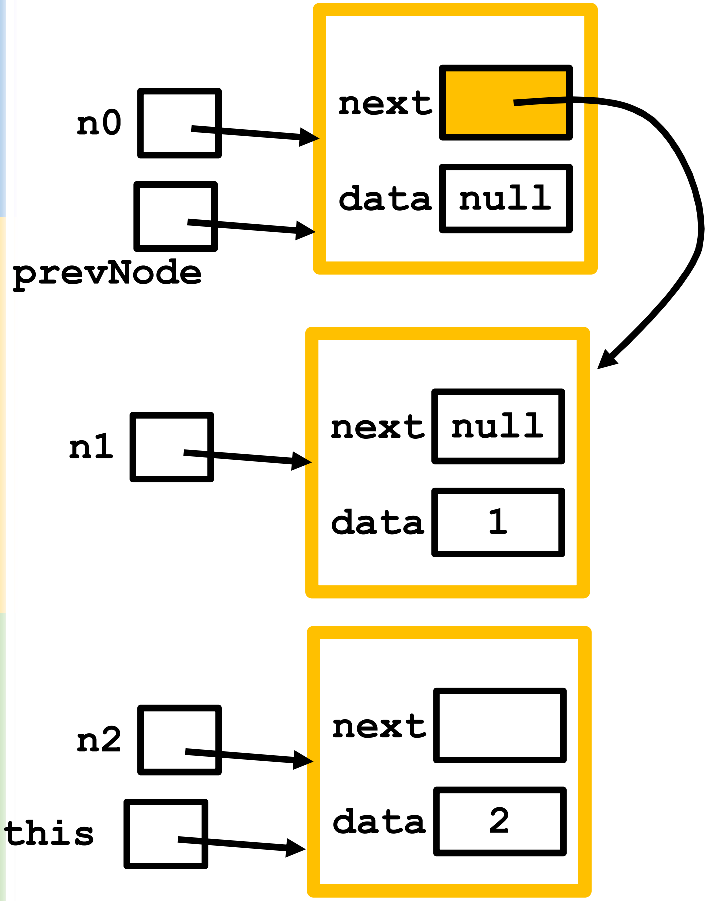
```
public SLLNode(E theData,
                         SLLNode<E> prevNode) {
      this(theData);
      this.next = prevNode.next;
      prevNode.next = this;
   }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```



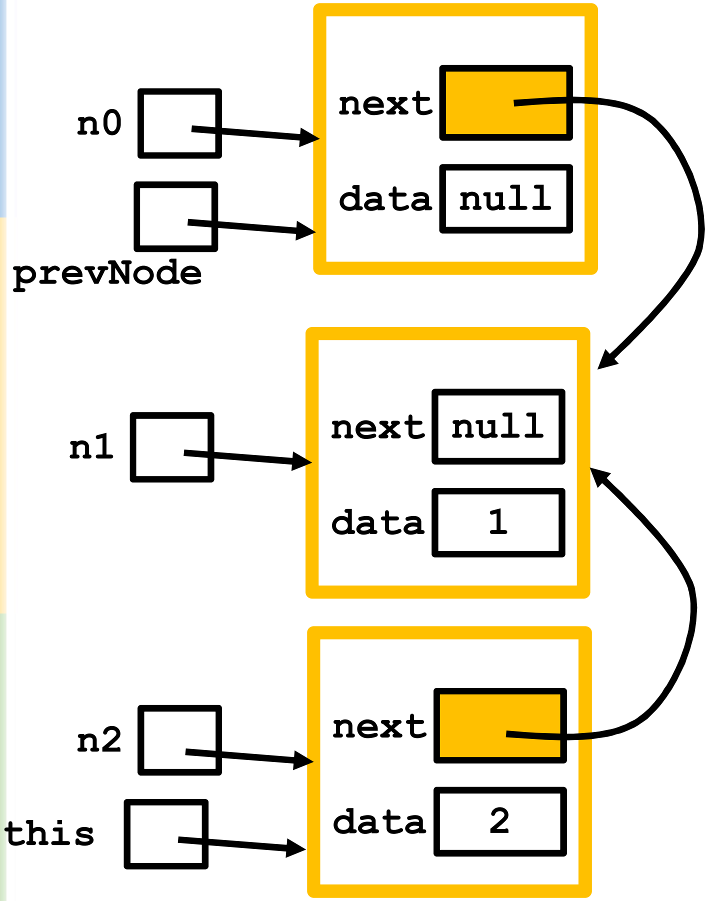
```
public SLLNode(E theData,
                        SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```



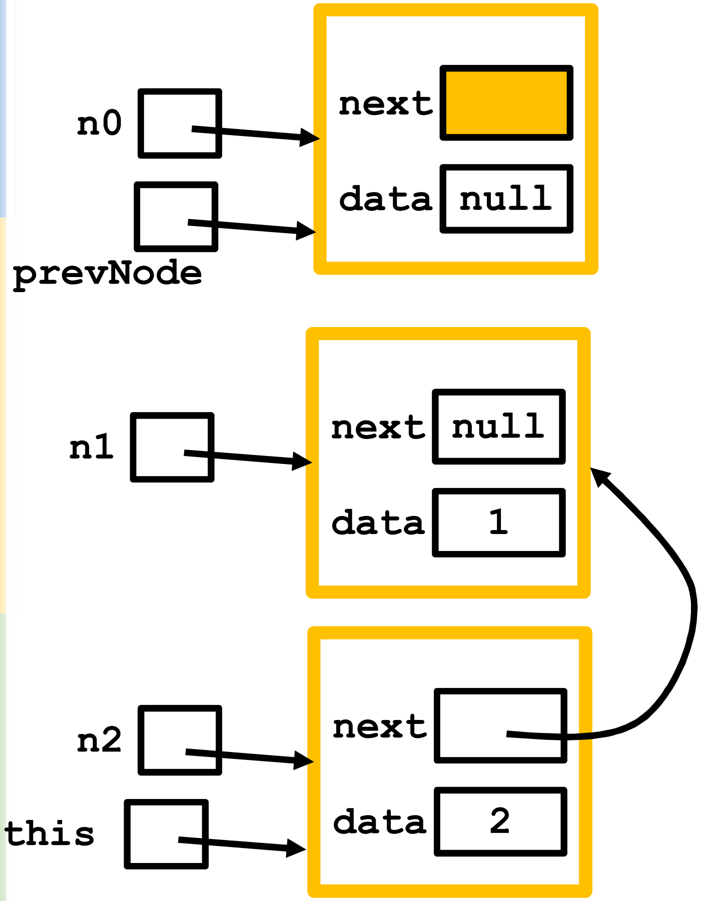
```
public SLLNode(E theData,
                 SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```



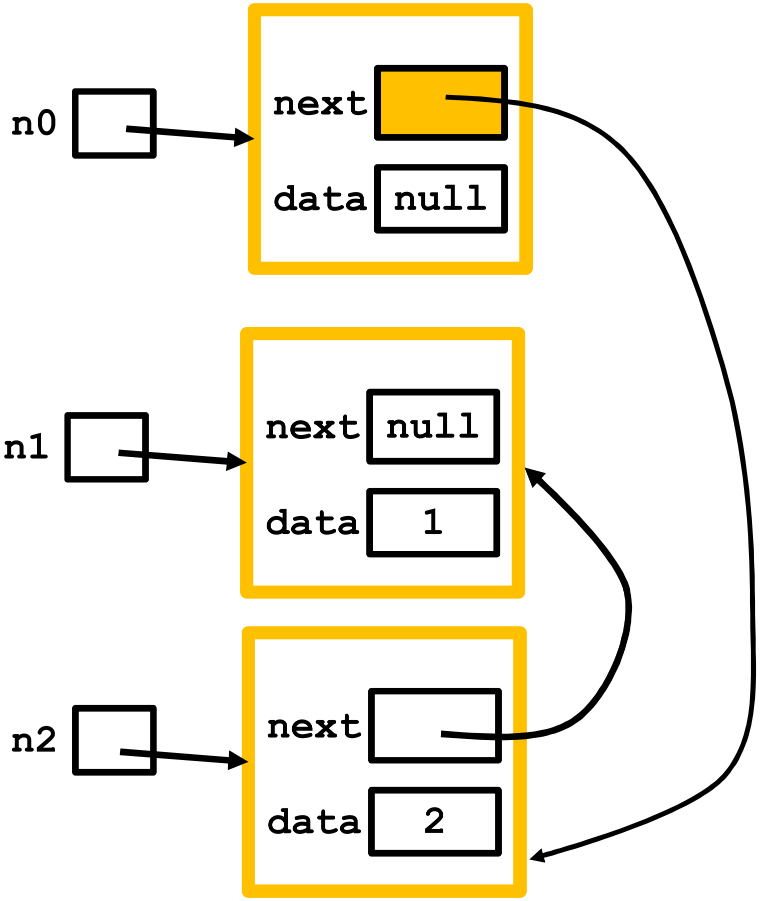
```
public SLLNode(E theData,
                SLLNode<E> prevNode) {
    this(theData);
    this.next = prevNode.next;
    prevNode.next = this;
  }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```

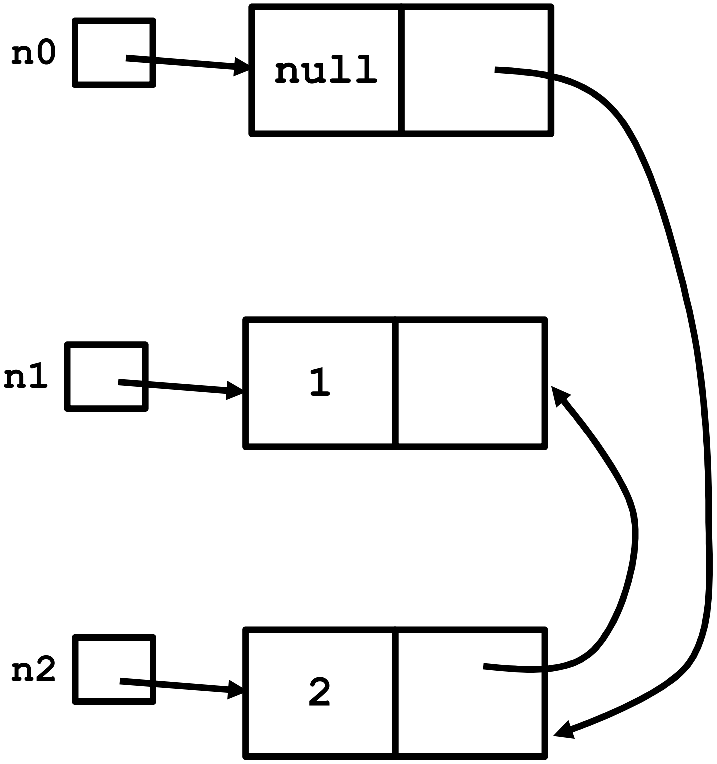


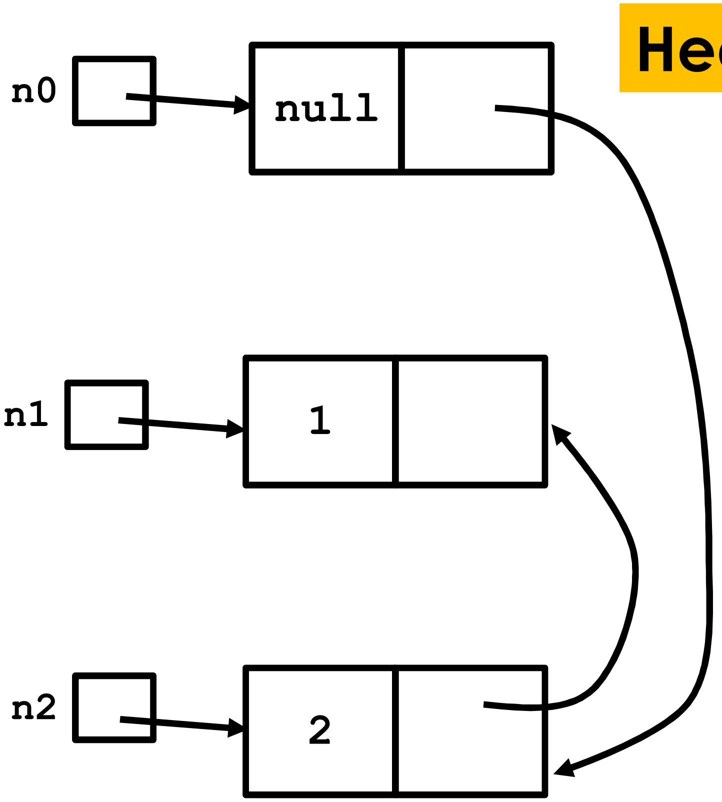
```
public SLLNode(E theData,
                           SLLNode<E> prevNode) {
     this(theData);
     this.next = prevNode.next;
     prevNode.next = this;
  }
```

```
SLLNode<Integer> n2 = new SLLNode(2,n0);
```



```
public SLLNode(E theData,
                 SLLNode<E> prevNode) {
     this(theData);
     this.next = prevNode.next;
     prevNode.next = this;
   }
```


n0
null
n1
1
n2
2



Head of list (Sentinel)