Lecture
Chapter 5
Linked Lists

determine node structure

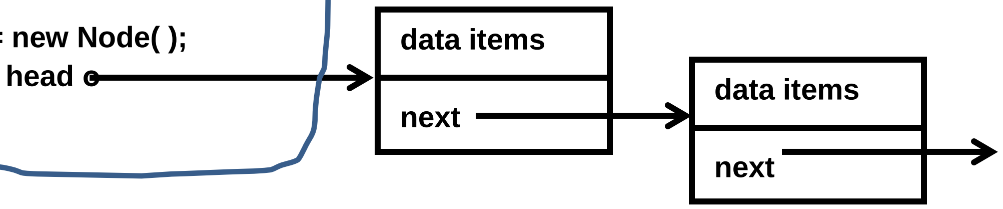
- **Java – object**
- **C – structure**
- **C++ -- structure or object**

Main Class

```
class Node
{
   determine data items
   Node next;
}

Node head, curr, prev;

head = new Node( );
```



Abstract Classes -- extend

- **Define data types ➔ -- filled: Boolean**
- **Define methods ➔ double area (double len, double wid) { return len * wid}**
- **Specify abstract methods ➔ double area (double length, double width) = 0;**

<u>ADT List – Reference-Based Implementation</u> **pages 265 -- 268**

```
package List:
public interface ListInterface
{
  public boolean isEmpty( );
  public int size( );
  public void add( int index, Object item ) throws ListIndexOutOfBoundsException;
  public void remove( int index ) throws ListIndexOutOfBoundsException;
  public Object  get( int index ) throws ListIndexOutOfBoundsException;
  public removeAll( );
}
```

```
package List:
public class ListReferenceBased implements ListInterface
{
  private Node head;
  private int numItems;



}
```

- **Define data types ➔ -- filled: Boolean**
- **Specify abstract methods ➔**
    - **double area (double length, double width) = 0;**
    - **public Boolean isFilled();**

private data items **private methods** **public methods**

Recursive Processing of Linked Lists

- **recursive traversal**
    - **writeList( )**
    - **writeBackward2( )**
    - **writeBackward( ) -- arrays**

tail references page 277-278
circular linked lists page 278
doubly linked lists – bidirectional linked lists page 282

Inventory Project pages 284—290

Java Collections Framework (JCF)

Generics

```
public class MyClass<E>
{
  private   E   theData;
  private int n;
  etc.
}

static public void main(String [ ] args)
{
  MyClass<String> a = new MyClass<String>( );
  etc.
}
```

java.util.Iterator

```
public interface Iterable<E>
{
   …
}
```

java.util.ListIterator

```
public interface ListIterable<E> extends Iterable<E>
{
  …
}
```

List Interface

```
public interface List<E> extends Collection<E>
```