Chapter 17 Programming

- No changes were made to Node.java

Method: insertEnd

```java
/**
 * Insert at the end
 * @param x the item to insert.
 */
public void insertEnd(AnyType _x)
{
    // Use previous to keep the last node
    Node<AnyType> previous = this.list;
    // This is the start of the chain
    Node<AnyType> current = this.list.getNextNode();

    // Iterate though the nodes
    while (current != null)
    {
        previous = current;
        current = current.getNextNode();
    }

    // Attach the new node to the last node's pointer
    Node<AnyType> last = new Node<AnyType>(_x);
    previous.setNextNode(last);
}
```

Method: removeLast

```java
/**
 * Remove the last occurrence of an item.
 * @param x the item to remove.
 */
public void removeLast(AnyType _x)
{
    Node<AnyType> matchBefore = null;
    Node<AnyType> matchAfter = null;

    // Use previous to keep the last node
    Node<AnyType> previous = this.list;
    // This is the start of the chain
    Node<AnyType> current = this.list.getNextNode();

    // Iterate though the all the nodes
    while (current != null)
    {
        // Check the node for the value
        if (current.getItem() == _x)
        {
            // Retain the previous and next nodes
            matchBefore = previous;
            matchAfter = current;
        }
        previous = current;
        current = current.getNextNode();
    }

    // Remove the node between the last prevous and next nodes
    if (matchBefore != null && matchAfter != null)
    {
        matchBefore.setNextNode(matchAfter.getNextNode());
    }
}
```

Chapter 17 Programming

Method: removeAll

```java
/**
 * Remove all occurrences of an item.
 * @param x the item to remove.
 */
public void removeAll(AnyType _x)
{
    // Use previous to keep the last node
    Node<AnyType> previous = this.list;
    // This is the start of the chain
    Node<AnyType> current = this.list.getNextNode();

    // Iterate though the all the nodes
    while (current != null)
    {
        // Remove every node that matches the value
        if (current.getItem() == _x)
        {
            previous.setNextNode(current.getNextNode());
        }
        previous = current;
        current = current.getNextNode();
    }
}
```

Method: previous

```java
/**
 * Return Node just before the first node containing an item.
 * @param x the item to search for.
 * @return a Node; node is not valid if item is not found, or is the
 * first item in the list.
 *
 */
public Node<AnyType> previous(AnyType _x)
{
    // Use previous to keep the last node
    Node<AnyType> previous = null;
    // This is the start of the chain
    Node<AnyType> current = this.list.getNextNode();

    // Iterate though the nodes until one matches the value
    while (current != null)
    {
        if (current.getItem() == _x)
        {
            break;
        }
        previous = current;
        current = current.getNextNode();
    }

    return previous;
}
```

Chapter 17 Programming

Method: findAndMoveToFront

```java
    /**
     * Return Node corresponding to the first node containing an item.
     * @param x the item to search for.
     * @return a Node; node is not valid if item is not found.
     *
     * As a by product, the found node is moved to the front of the list
     * according to the move to front heuristic of 17.17
     */
    public Node<AnyType> findAndMoveToFront(AnyType _x)
    {
        // First use the find method to get the node with the value
        Node<AnyType> found = this.find(_x);
        // If we found a match use the remove method, then the insertFront method
        if (found != null)
        {
            this.remove(_x);
            this.insertFront(_x);
        }

        return found;
    }
```

Testing Output

```
c:\Users\wes\github-repos\cs2420_summer2023\Chapter17\Programming - VS Code Console
Testing LinkedList methods
Finished LinkedList testing
Press any key to continue . . .
```