5. Abstract Data Types: SINGLE LINKED LIST

Lecture 5

Mark Cudden

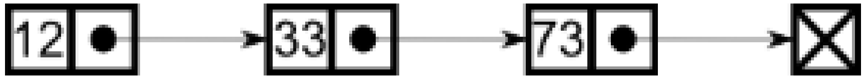
Mark.Cudden@ncirl.ie

5.1 Anatomy of a Linked List

- A LIST is a sequence of connected nodes:
    - A node’s successor is the next node in the sequence
    - A node’s predecessor is the previous node in the sequence
- Each node contains:
    - an object and
    - link(s) (pointer or reference) to its successor (and predecessor)
- The first node in the list is named header
- The last node contains a null link

5.2 Single-Linked Lists

- Here is a single-linked list (SLL):



- Each node contains an object and ONE link to its successor
- The header is a reference to the first node in the list
- Some methods for linked lists
  - isEmpty()
  - size()
  - get (index)
  - remove (index)
  - add (theElement, index)

5.3 Linked List Interface in Java

```
public interface LinearList
{
        public boolean isEmpty();
        public int size();
        public Object get(int index);
        public void remove(int index);
        public void add(Object theElement, int index);
        public void printList();
}
```

5.4. Single Linked List in Java
Node class

```
class Node {
        private Object element;
        private Node next;

        public Node(Object e, Node n){
                element = e;
                next = n;
                }
        public Node getNext() {
                return next;
        }
        // other set and get methods
}  // end of class Node
```

5.4. Single Linked List in Java SLList class

```
public class SLList implements LinearList {
      private Node head;
      private Node curr;
      private Node prev;
      private int size;

      SLList()
      {
            head = null; size = 0;
            curr = head; prev = null
      }

      public boolean isEmpty()
      {
             if (size = = 0)
                        return true
            else
                        return false
      }

      public int size()
      {
            return size;
      }
} // end class
```

5.4. Single Linked List in Java
Inserting a node

- There are many ways you might want to insert a new node into a list:
  - As the new first element
  - As the new last element
  - Before a given node (specified by a reference)
  - After a given node
  - Before a given value
  - After a given value
  - At a given position – index

- All are possible, but differ in difficulty
  - At a given position - index

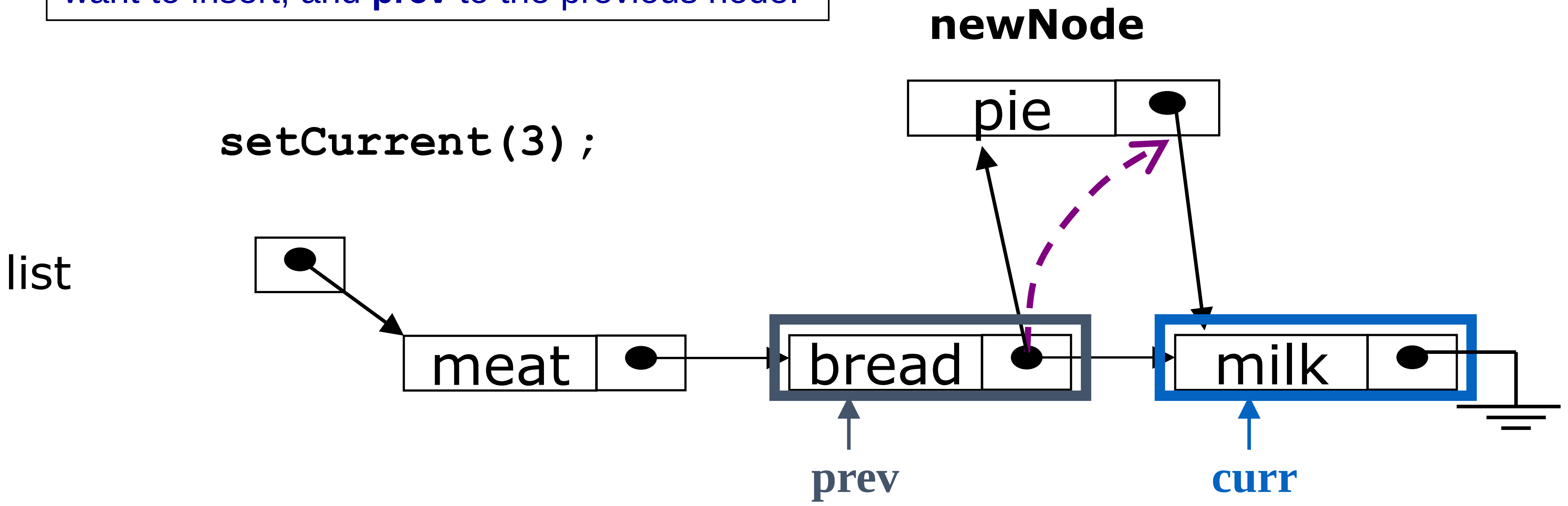
5.4. Single Linked List in Java
Inserting a node on a position (animation)

```
add(3,"pie");
```

1. Set **curr** to the node at the position you want to insert, and **prev** to the previous node.

2. Make new node

```
setCurrent(3);
```



3. Copy the link from **prev** node that's already in the list in the next of the newNode

4. Change the link in **prev** node that's already in the list to points towards newNode

5.4. Single Linked List in Java add (int index, Object item)

```
// Insert element at a given position

// assume the index is in the correct range
public void add(int index, Object item){
    // special case of adding at the head of the list
    if (index == 1){
        Node newNode = new Node(item,head);
        head = newNode;
    }
    else{
        setCurrent(index);
        Node newNode = new Node(item, curr);
        prev.setNext(newNode);
    }
    size=size+1;
}
```

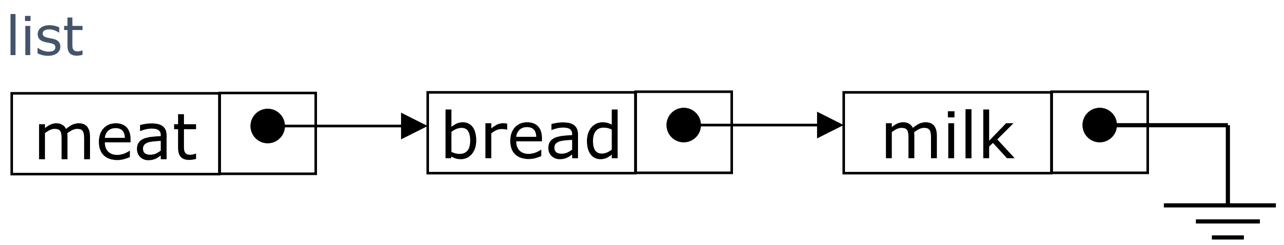
5.4. Single Linked List in Java setCurrent(int index)

```
private void setCurrent(int index){
   int k;
           prev = null;
           curr = head;
           for (k  = 1; k < index; k++){
                         prev = curr;
                         curr = curr.getNext();
       }
}
```

- **As a result of this method:**
  - *curr* is set to point to the index-th element in the list
  - *prev* is set to point to the predecessor of the index-th element in the list

5.4. Single Linked List in Java
Creating a simple list

- To create the list ("meat", "bread", “milk"):
  - SLList list = new SLList();
  - list.add(1,"meat");
  - list.add(2,"bread");
  - list.add(3,"milk");
- This code may be part of the main() method

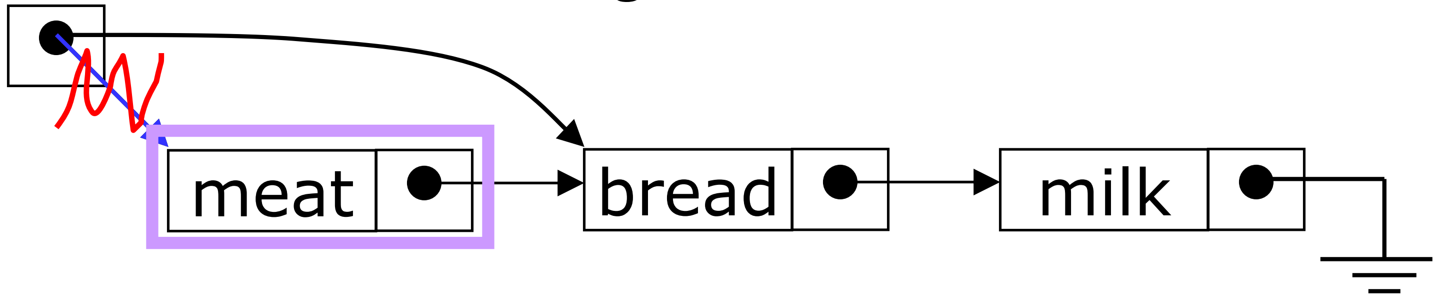


5.4. Single Linked List in Java
Deleting a node

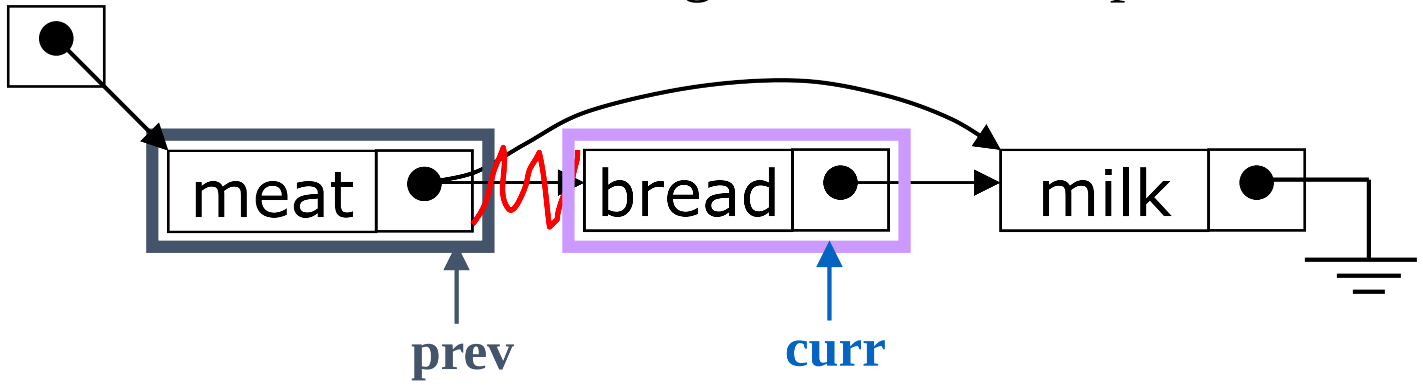
- In order to delete a node from a SLL, you have to change the link in its predecessor
- This is slightly tricky, because you can’t follow a pointer backwards
- Deleting the first node in a list is a special case, because the node’s predecessor is the list header

5.4. Single Linked List in Java
Removing an element from a given position

- To remove the first element, change the link in the header list



- To remove some other element, change the link in its predecessor list

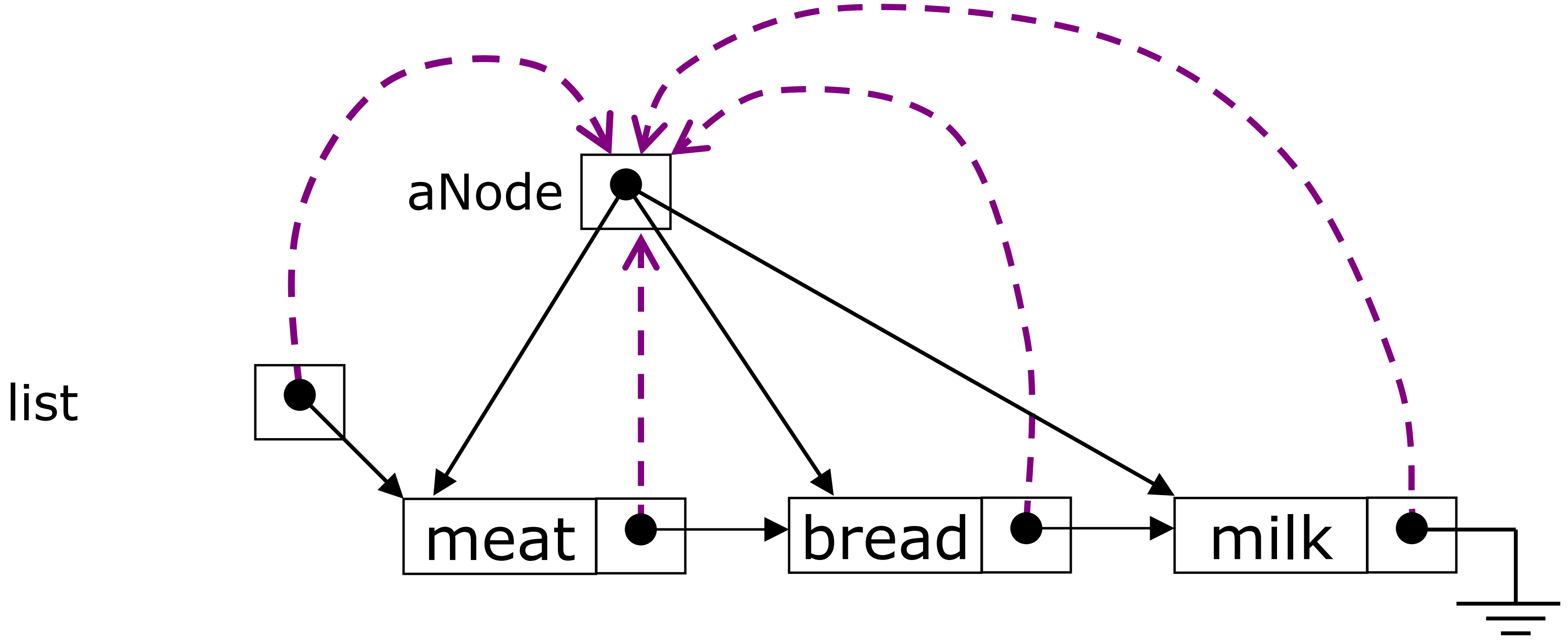


5.4. Single Linked List in Java remove(int index)

Remove an element from a given position

```
public void remove(int index){
      // special case of removing at the head of the list
            if (index == 1){
                        head = head.getNext();
            }
    else{
      // find the previous and current node
                        setCurrent(index);
                        prev.setNext(curr.getNext());
            }
    size=size-1;
}
```

5.4. Single Linked List in Java
Traverse a linked list – Animation



meat **bread** **milk**

5.4. Single Linked List in Java
Traverse a linked list – printList()

Write Java code to traverse a linked list and print out the content of each of its nodes.

```
public void printList(){
        Node aNode = head;
        while ( aNode != null ) {
         System.out.println(aNode.getElement().toString());
         aNode=aNode.getNext());
        }
}
```

5.4. Single Linked List in Java
Tasks for tutorial

- Create a NetBeans application that implements a Single Linked List
- Add the following java classes
  - LinearList interface
  - Node class
  - SLList class
  - Tester class – will have the main() method
  - The code is in the notes and on the Web
- Implement the main() method in Tester class that performs:
  - Create a SLList object and add a number of nodes.
  - The information from the node may be a string
  - Display the current size of the list
  - Print all the elements from the list
  - Remove an element from a given position
  - List again the content of the list

5.5 Revision Questions on Linked Lists

- Describe the Single Linked List ADT.
- Name and describe the methods/operations for the Single Linked List ADT.
- Illustrate the removal principle for SLL
- Advantages and Disadvantages of the SLL
- Write the Java code for the add/remove method
- Write the Java code for the Node class