## Solutions to Laboratory Exercise 4

# Implementation of ADTs - linked lists

1. A constructor can be a class that contains a method for defining a new type. The name of a Java constructor method must be identical to (the name of) the class containing it. A constructor may contains methods for polymorphic purpose.

2. It defines a new type Node which contains a field of String type and of field of Node type.

   The new type can be called with *one* arguments or *two*. The first one is for the node where the next field is null in value.

3. ```
   public class Node {
     private Object item;
     private Node next;

     public Node(Object newItem) {
       item = newItem;
       next = null;
     } // Constructor

     public Node(Object newItem, Node nextNode) {
       item = newItem;
       next = nextNode;
     } // Constructor
   } // end class Node
   ```

4. (a) created a node:

   ```
   "pen" -> null
   ```

   (b) created a node:

   ```
   "pencil" -> null
   ```

   (c) created a linked list containing 2 nodes:

   ```
   "pencil" -> "pen" -> null
   ```

   (d) created a linked list containing 3 nodes:

   ```
   "cat" -> "ate" -> "an apple?" -> null
   ```

5. ```
   import java.io.*;

   public class Node {
     private Object item;
     private Node next;

     public Node(Object newItem) {
       item = newItem;
       next = null;
     } // Constructor

     public Node(Object newItem, Node nextNode) {
       item = newItem;
       next = nextNode;
   ```

```
  } // Constructor

public void setItem(Object newItem) {
    item = newItem;
  } // end setItem

  public Object getItem() {
    return item;
  } // end getItem

  public void setNext(Node nextNode) {
    next = nextNode;
  } // end nextNode

  public Node getNext() {
    return next;
  } // end getNext()

} // end class Node

// This is a program for testing (not required by question):
public class testList {
  public static void main(String [] args) {
    String s="an-apple?";
    Node n = new Node(s);
    System.out.println(n.getItem());
    n = new Node("ate", n);
    System.out.println(n.getItem());
    n = new Node("cat", n);
    System.out.println(n.getItem());
  } // end main
} // end testList
```

6.
```
  public static void display(Node L) {
    Node current = L;
    while (current!=null) {
      System.out.println(current.getItem());
      current = current.getNext();
    }  // end while
  } // end display
```

7.
```
public static Node newEmptyList() {
    return null;
  } // end newEmptyList
```

8.
```
public static boolean isEmpty(Node L) {
    return (L==null);
  } // end isEmpty
```

9.
```
// The recursive version
  public static int size(Node L) {
    if (L==null) {
       return 0;
    }
    else return (1+size(L.getNext()));
  } // end size
```

10. (Optional) [9]

[9] *No solution available, open for discussion.*