```java
/**********************************************************************
 * Class for LineList, providing overall list of Linked List structure
 * @author Michael Hartung, Matthew Armand
 */
public class LineList {
    //Current and head object references for line objects
    private Line head, current;
    private LineList next, prev;
    private int clipNum;
    /**********************************************************************
     * Constructs a new, empty LineList object
     */
    public LineList() {
        head = null;
        current = null;
        next = null;
        prev = null;
    }

    /**********************************************************************
     * Inserts a new line before the current line
     * @param newLine content to be set as the text of new line
     */
    public void insertBefore(String newLine) {
        Line addLine = new Line(newLine);
        if (current == null) {
            current = head = addLine;
        }
        else if (current.getPrev() != null) {
            addLine.setNext(current);
            addLine.setPrev(current.getPrev());
            current.getPrev().setNext(addLine);
            current.setPrev(addLine);
            current = addLine;
        }
        else {
            addLine.setNext(head);
            head.setPrev(addLine);
            current = head = addLine;
        }
    }

    /**********************************************************************
     * Inserts a new line at the end of the list
     * @param newLine content to be set as the text of new line
     */
    public void insertLast(String newLine) {
        while(current.getNext() != null) {
```

```
            down();
        }
        insertAfter(newLine);
    }

    /**************************************************************************
     * Inserts a new line after the current line
     * @param newLine content to be set as the text of new line
     */
    public void insertAfter(String newLine) {
        Line addLine = new Line(newLine);
        if (current == null) {
            current = head = addLine;
        }
        else if (current.getNext() != null) {
            addLine.setNext(current.getNext());
            addLine.setPrev(current);
            current.getNext().setPrev(addLine);
            current.setNext(addLine);
            current = addLine;
        }
        else {
            addLine.setPrev(current);
            current.setNext(addLine);
            current = addLine;
        }
    }

    /**************************************************************************
     * Moves current line indicator down one position
     */
    public void down() {
        if(current.getNext() != null) {
            current = current.getNext();
        }
        else {
            System.out.println("Bottom Line Reached!");
        }
    }

    /**************************************************************************
     * Moves current line indicator up one position
     */
    public void up() {
        if(current.getPrev() != null) {
            current = current.getPrev();
        }
        else {
```

```
        System.out.println("Top Line Reached!");
    }
}

/**************************************************************************
 * Removes current line from the list
 */
public void remove() {
    if (current == null) {
        System.out.println("Nothing to remove!");
    }
    else if(current.getNext() != null) {
        if(current.getPrev() != null) {
            current.getPrev().setNext(current.getNext());
            current.getNext().setPrev(current.getPrev());
            current = current.getNext();
        }
        else {
            current.getNext().setPrev(null);
            head = current.getNext();
            current = current.getNext();
        }
    }
    else if(current.getPrev() != null) {
        current.getPrev().setNext(null);
        current = current.getPrev();
    }
    else {
        head = current = null;
    }
}

/**************************************************************************
 * Sets the input line as the current line indicator in the list
 * @param l Line to be set as current
 */
public void setCurrent (Line l) {
    current = l;
}

/**************************************************************************
 * Gets the line currently set as current within the list
 * @return current Line in list
 */
public Line getCurrent () {
    return current;
}
```

```java
/**************************************************************************
 * Gets the line at the head of the list
 * @return object reference to head of the list
 */
public Line getHead () {
    return head;
}

/**************************************************************************
 * Displays certain lines from x to y in the list
 * @param x starting point to display
 * @param y ending point to display
 * @return string representation of desired lines in list
 */
public String display (int x, int y) {
    String result = "";
    Line pass = head;
    int lineNumber = 1;
    boolean done = false;
    while (pass != null && !done) {
        if (lineNumber == x) {
            while (pass != null && !done) {
                if (pass == current) {
                    result += "--> ";
                }
                else {
                    result += "    ";
                }
                result += lineNumber + ": ";
                result += pass.toString();
                pass = pass.getNext();
                lineNumber++;
                if (lineNumber == y+1) {
                    done = true;
                }
            }
        }
        lineNumber++;
        try {
            pass = pass.getNext();
        } catch (NullPointerException e) {
            done = true;
        }
    }
    return result;
}

/**************************************************************************
```

```
     * Returns a string representation of the list object
     */
    public String toString() {
        String result = "";
        Line pass = head;
        int lineNumber = 1;
        while(pass != null) {
            if(pass == current) {
                result += "--> ";
            }
            else {
                result += "    ";
            }
            result += lineNumber + ": ";
            result += pass.toString();
            pass = pass.getNext();
            lineNumber++;
        }
        return result;
    }

    public void setPrev(LineList pPrev) {
        prev = pPrev;
    }

    public void setNext(LineList pNext) {
        next = pNext;
    }

    public LineList getPrev() {
        return prev;
    }

    public LineList getNext() {
        return next;
    }

    public int getClipNum() {
        return clipNum;
    }

    public void setClipNum(int pClipNum) {
        clipNum = pClipNum;
    }
}
```