C:\Users\Rich\Documents\NetBeansProjects\Lab07\src\PositionalList.java

```java
/**
 * Data Structures & Algorithms 6th Edition
 * Goodrick, Tamassia, Goldwasser
 * Code Fragement 7.8
 */

/**
 * An interface for positional lists.
 */
public interface PositionalList<E> {

    /**
     * @return the number of elements in the list.
     */
    int size( );

    /**
     * @return true if the list is empty.
     */
    boolean isEmpty( );

    /**
     * @return the first Position in the list ( or null, if empty ).
     */
    Position<E> first( );

    /**
     * @return the last Position in the list ( or null, if empty ).
     */
    Position<E> last( );

    /**
     * @param p a position in the list,
     * @return position immediately before p ( or null if p is first ).
     * @throws IllegalArgumentException if p is not in list.
     */
    Position<E> before( Position<E> p ) throws IllegalArgumentException;

    /**
     * @param p a position in the list,
     * @return position immediately after p ( or null if p is last ).
     * @throws IllegalArgumentException if p is not in list.
     */
    Position<E> after( Position<E> p ) throws IllegalArgumentException;

    /**
     * @param e element to be inserted at front of list
     * @return position of inserted element
     */
    Position<E> addFirst( E e );

    /**
     * @param e element to be inserted at back of list
     * @return position of inserted element
     */
    Position<E> addLast( E e );

    /**
     * @param p position to be inserted before
     * @param e element to be inserted before position p
     * @return position of e
     * @throws IllegalArgumentException if p not in list
     */
    Position<E> addBefore( Position<E> p, E e ) throws IllegalArgumentException;

    /**
     * @param p position to be inserted after
     * @param e element to be inserted after position p
     * @return position of e
     * @throws IllegalArgumentException if p not in list
     */
    Position<E> addAfter( Position<E> p, E e ) throws IllegalArgumentException;

    /**
     * @param p position to store element at
     * @param e element to be stored at p
     * @return the element that is replaced
     * @throws IllegalArgumentException if p is not in list
     */
    E set( Position<E> p, E e ) throws IllegalArgumentException;

    /**
     * @param p position of element to be removed
     * @return removed element
     * @throws IllegalArgumentException if p not in list
     */
    E remove( Position<E> p ) throws IllegalArgumentException;

}

```