```
// $Id: edfile.java,v 1.9 2014-04-15 19:24:24-07 - - $

import java.util.Scanner;
import static java.lang.System.*;

class edfile{

   public static void main (String[] args) {
      boolean want_echo = true;
      dllist lines = new dllist ();
      auxlib.STUB ("Check for -e option");
      auxlib.STUB ("Load file from args filename, if any.");
      Scanner stdin = new Scanner (in);
      for (;;) {
         out.printf ("%s: ", auxlib.program_name());
         if (! stdin.hasNextLine()) break;
         String inputline = stdin.nextLine();
         if (want_echo) out.printf ("%s%n", inputline);
         if (inputline.matches ("^\\s*$")) continue;
         char command = inputline.charAt(0);
         switch (command) {
            case '#': break;
            case '$': auxlib.STUB ("Call $ command function."); break;
            case '*': auxlib.STUB ("Call * command function."); break;
            case '.': auxlib.STUB ("Call . command function."); break;
            case '0': auxlib.STUB ("Call 0 command function."); break;
            case '<': auxlib.STUB ("Call < command function."); break;
            case '>': auxlib.STUB ("Call > command function."); break;
            case 'a': auxlib.STUB ("Call a command function."); break;
            case 'd': auxlib.STUB ("Call d command function."); break;
            case 'i': auxlib.STUB ("Call i command function."); break;
            case 'r': auxlib.STUB ("Call r command function."); break;
            case 'w': auxlib.STUB ("Call w command function."); break;
            default : auxlib.STUB ("Print invalid command."); break;
         }
      }
      auxlib.STUB ("(eof)");
   }

}
```

```java
// $Id: dllist.java,v 1.1 2014-04-10 17:01:54-07 - - $

class dllist {

   public enum position {FIRST, PREVIOUS, FOLLOWING, LAST};

   private class node {
      String item;
      node prev;
      node next;
   }

   private node first = null;
   private node current = null;
   private node last = null;
   private int currentposition = 0;

   public void setposition (position pos) {
      throw new UnsupportedOperationException();
   }

   public boolean isempty () {
      throw new UnsupportedOperationException();
   }

   public String getitem () {
      throw new UnsupportedOperationException();
   }

   public int getposition () {
      throw new UnsupportedOperationException();
   }

   public void delete () {
      throw new UnsupportedOperationException();
   }

   public void insert (String item, position pos) {
      throw new UnsupportedOperationException();
   }

}

```

```java
// $Id: auxlib.java,v 1.2 2014-04-10 17:30:42-07 - - $

import static java.lang.System.*;

class auxlib {
   public static final int EXIT_SUCCESS = 0;
   public static final int EXIT_FAILURE = 1;
   public static int exit_status = EXIT_SUCCESS;

   //
   // program_name - Extract the basename of the jar file containing
   // the Java program, which appears as the class path.
   //
   public static String program_name() {
      String jarname = getProperty ("java.class.path");
      return jarname.substring (jarname.lastIndexOf ("/") + 1);
   }

   //
   // warn - Print a warning and set exit status to failure.
   //
   public static void warn (Object... args) {
      exit_status = EXIT_FAILURE;
      out.flush();
      err.printf ("%s", program_name());
      for (Object arg: args) err.printf (": %s", arg);
      err.printf ("%n");
      err.flush();
   }

   //
   // die - Print a warning and exit program.
   //
   public static void die (Object... args) {
      warn (args);
      exit (exit_status);
   }

   //
   // usage - Print a usage message and exit program.
   //
   public static void usage (Object... args) {
      exit_status = EXIT_FAILURE;
      out.flush();
      err.printf ("Usage: %s", program_name());
      for (Object arg: args) err.printf (" %s", arg);
      err.printf ("%n");
      err.flush();
      exit (exit_status);
   }

   public static void STUB (String... args) {
      out.printf ("%s:%n", Thread.currentThread().getStackTrace()[1]);
      for (Object arg: args) out.printf (": %s", arg);
      out.printf ("%n");
   }

}
```

```
# $Id: Makefile,v 1.4 2014-04-10 17:30:42-07 - - $

JAVASRC    = edfile.java dllist.java auxlib.java
SOURCES    = ${JAVASRC} Makefile README
MAINCLASS  = edfile
CLASSES    = ${JAVASRC:.java=.class}
JARCLASSES = ${CLASSES} dllist\$$*.class
JARFILE    = edfile
LISTING    = Listing.ps
SUBMITDIR  = cmps012b-wm.s14 asg2

all : ${JARFILE}
	- checksource ${SOURCES}

${JARFILE} : ${CLASSES}
	echo Main-class: ${MAINCLASS} >Manifest
	jar cvfm ${JARFILE} Manifest ${JARCLASSES}
	chmod +x ${JARFILE}
	- rm Manifest

%.class : %.java
	- cid + $<
	javac $<

clean :
	- rm ${JARCLASSES} Manifest

spotless : clean
	- rm ${JARFILE}

ci : ${SOURCES}
	- checksource ${SOURCES}
	- cid + ${SOURCES}

lis : ${SOURCES}
	mkpspdf ${LISTING} ${SOURCES}

submit : ${SOURCES}
	submit ${SUBMITDIR} ${SOURCES}

again:
	gmake --no-print-directory spotless ci all lis

```

```
$Id: README,v 1.1 2014-04-10 17:01:54-07 - - $
```