```
// $Id: edfile.java,v 1.2 2010-10-19 15:10:27-07 - - $

import static java.lang.System.*;

class edfile{

   public static void main (String[] args) {
      dllist lines = new dllist ();
      out.printf ("%s: ", auxlib.PROGNAME);
      if (args.length == 0) {
         lines.setposition (dllist.position.FIRST);
      }else {
         auxlib.usage_exit ("[-e] [filename...]");
      }
   }

}

```

```java
// $Id: dllist.java,v 1.4 2010-10-19 15:10:27-07 - - $

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();
   }

}

```

```
// $Id: auxlib.java,v 1.1 2010-10-05 15:37:29-07 - - $
//
// NAME
//    auxlib - Auxiliary miscellanea for handling system interaction.
//
// DESCRIPTION
//    Auxlib has system access functions that can be used by other
//    classes to print appropriate messages and keep track of
//    the program name and exit codes.  It assumes it is being run
//    from a jar and gets the name of the program from the classpath.
//    Can not be instantiated.
//

import static java.lang.System.*;
import static java.lang.Integer.*;

public final class auxlib{
   public static final String PROGNAME =
                  basename (getProperty ("java.class.path"));
   public static final int EXIT_SUCCESS = 0;
   public static final int EXIT_FAILURE = 1;
   public static int exitvalue = EXIT_SUCCESS;

   //
   // private ctor - prevents class from new instantiation.
   //
   private auxlib () {
      throw new UnsupportedOperationException ();
   }

   //
   // basename - strips the dirname and returns only the basename.
   //            See:  man -s 3c basename
   //
   public static String basename (String pathname) {
      if (pathname == null || pathname.length () == 0) return ".";
      String[] paths = pathname.split ("/");
      for (int index = paths.length - 1; index >= 0; --index) {
         if (paths[index].length () > 0) return paths[index];
      }
      return "/";
   }

   //
   // Functions:
   //    whine   - prints a message with a given exit code.
   //    warn    - prints a stderr message and sets the exit code.
   //    die     - calls warn then exits.
   // Combinations of arguments:
   //    objname - name of the object to be printed (optional)
   //    message - message to be printed after the objname,
   //              either a Throwable or a String.
   //
   public static void whine (int exitval, Object... args) {
      exitvalue = exitval;
      err.printf ("%s", PROGNAME);
      for (Object argi : args) err.printf (": %s", argi);
      err.printf ("%n");
   }
   public static void warn (Object... args) {
      whine (EXIT_FAILURE, args);
   }
   public static void die (Object... args) {
      warn (args);
      exit ();
   }

   //
   // usage_exit - prints a usage message and exits.
   //
   public static void usage_exit (String optsargs) {
      exitvalue = EXIT_FAILURE;
      err.printf ("Usage: %s %s%n", PROGNAME, optsargs);
      exit ();
   }

   //
   // exit - calls exit with the appropriate code.
   //        This function should be called instead of returning
   //        from the main function.
   //
   public static void exit () {
      System.exit (exitvalue);
   }

   //
   // identity - returns the default Object.toString value
   //            Useful for debugging.
   //
   public static String identity (Object object) {
      return object == null ? "(null)"
           : object.getClass().getName() + "@"
           + toHexString (identityHashCode (object));
   }

}
```

```
# $Id: Makefile,v 1.6 2010-10-19 15:10:56-07 - - $

JAVASRC    = edfile.java dllist.java auxlib.java
SOURCES    = ${JAVASRC} Makefile README
MAINCLASS  = edfile
CLASSES    = edfile.class dllist.class auxlib.class
JARCLASSES = ${CLASSES} dllist\$$node.class dllist\$$position.class
JARFILE    = edfile
LISTING    = ../asg2j-edfile.code.ps
SUBMITDIR  = cmps012b-wm.f10 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}
	testsubmit ${SUBMITDIR} ${SOURCES}

```

```
$Id: README,v 1.1 2010-10-05 15:37:29-07 - - $
```