```
// $Id: bitreecalc.java,v 1.1 2013-10-16 12:58:29-07 - - $

class bitreecalc {

   public static void main (String[] args) {
   }

}

```

```java
// $Id: bitree.java,v 1.3 2013-10-17 18:34:13-07 - - $
//
// NAME
//    class bitree - starter class for bitree implementation.
//

class bitree {
   char symbol;
   bitree left;
   bitree right;

   bitree (char symbol_, bitree left_, bitree right_) {
      symbol = symbol_;
      left = left_;
      right = right_;
   }

   bitree (char symbol_) {
      this (symbol_, null, null);
   }

   public String toString () {
      //FIXME
      return "" + symbol;
   }
}

```

```
// $Id: linked_stack.java,v 1.1 2013-10-16 12:58:29-07 - - $
//
// NAME
//    class linked_stack - implementation of stack
//

import java.util.NoSuchElementException;

class linked_stack<item_t> {

   private class node {
      item_t value;
      node link;
   }

   private node top = null;

   public boolean empty() {
      return top == null;
   }

   public item_t pop() {
      if (empty()) {
         throw new NoSuchElementException ("linked_stack.pop");
      }
      //FIXME
      return null;
   }

   public void push (item_t value) {
      //FIXME
   }

}

```

```
// $Id: symbol_table.java,v 1.3 2013-10-17 18:33:53-07 - - $
//
// NAME
//    class symbol_table
//       Symbol table with letter indices and double and tree values.
//

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

class symbol_table {

   //
   // Constants for use within this calss.
   //
   private static final char LO_LETTER = 'a';
   private static final char HI_LETTER = 'z';
   private static final int ARRAYLEN = HI_LETTER - LO_LETTER + 1;
   private double[] values = new double[ARRAYLEN];
   private bitree[] trees = new bitree[ARRAYLEN];

   //
   // Convert letter into array index.
   //
   private int aindex (char varname) {
      int index = Character.toLowerCase (varname) - LO_LETTER;
      if (index < 0 || index >= ARRAYLEN) {
         throw new IndexOutOfBoundsException (
                  format ("'%c' is out of bounds: '%c'..'%c'",
                         varname, LO_LETTER, HI_LETTER));
      }
      return index;
   }

   //
   // Constructor.  Defaults all values to NaN.
   //
   public symbol_table() {
      for (int index = 0; index < values.length; ++index) {
         values[index] = Double.NaN;
      }
   }


   //
   // Accessors.  Get the value or the tree from the symbol table.
   //
   public double get_value (char varname) {
      return values[aindex (varname)];
   }

   public bitree get_tree (char varname) {
      return trees[aindex (varname)];
   }

   //
   // Mutators.  Change the value and the tree in the table.
   //
   public void put (char varname, double value, bitree tree) {
      int index = aindex (varname);
      values[index] = value;
      trees[index] = tree;
   }

}
```

```
// $Id: auxlib.java,v 1.2 2013-10-17 18:33:53-07 - - $
//
// 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 "/";
   }

```

```java

   //
   // 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 1.2 2013-10-17 18:28:59-07 - - $

JAVASRC    = bitreecalc.java bitree.java linked_stack.java \
             symbol_table.java auxlib.java
SOURCES    = ${JAVASRC} Makefile README
MAINCLASS  = bitreecalc
CLASSES    = ${JAVASRC:.java=.class}
JARCLASSES = ${CLASSES} linked_stack\$$node.class
JARFILE    = bitreecalc
LISTING    = Listing.ps
SUBMITDIR  = cmps012b-wm.f13 asg3

all : ${JARFILE}

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

%.class : %.java
        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}

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

```

```
This directory contains code you should copy to your development
directory in order to begin work.

$Id: README,v 1.1 2013-10-16 12:58:29-07 - - $
```