import java.util.*;

/**
* Directory of JFiles.
*
* A Directory is a JFile that maintains a
* table of the JFiles it contains.
*
* @version 10
*/

public class Directory extends JFile {

private TreeMap jfiles;  // table for JFiles in this Directory

/**
* Construct a Directory.
*
* @param name the name for this Directory (in its parent Directory)
* @param creator  the owner of this new Directory.
* @param parent   the Directory in which this Directory lives.
*/

public Directory( String name, User creator, Directory parent) {
super( name, creator, parent );
jfiles = new TreeMap();
}

/**
* The size of a Directory is the number of JFiles it contains.
*
* @return the Directory's size.
*/

public int getSize() {
return jfiles.size();
}

/**
* Suffix used for printing Directory names;
* we define it as the (system dependent)
* name separator used in path names.
*
* @return the suffix for Directory names.
*/

public String getSuffix() {
return JFile.separator;
}

/**
* Add a JFile to this Directory. Overwrite if a JFile
* of that name already exists.
*
* @param name the name under which this JFile is added.
* @param afile the JFile to add.
*/

public void addJFile(String name, JFile afile) {
jfiles.put( name, afile );
setModDate();
}

/**
* Get a JFile in this Directory, by name .
*
* @param filename the name of the JFile to find.
* @return the JFile found.
*/

public JFile retrieveJFile( String filename ) {
JFile aFile = (JFile)jfiles.get( filename );
return aFile;
}

/**
* Remove a JFile in this Directory, by name.
*
* @param filename the name of the JFile to remove
*/

public void removeJFile( String filename ) {
jfiles.remove( filename );
}

/**
* Get the contents of this Directory as an array of
* the file names, each of which is a String.
*
* @return the array of names.
*/

public String[] getFileNames() {
return (String[])jfiles.keySet().toArray( new String[0] );
}
}