Solution 2 – Composite Design Pattern

```
package composite;

import java.util.*;

abstract class Dentry {

  protected String name;
  protected Dentry parent = null;

  public Dentry(String s) {
    name = s;
  }

  public String toString() {
    return name;
  }

  public abstract int size();

  public void add(Dentry c) {}

  public void remove(Dentry c){}

  public Iterator getEntries() {
    return null;
  }

  public String pathName() {
    return parent==null
           ? name
           : parent.pathName()+"."+name;
  }

  protected void setParent(Dentry p) {
     parent = p;
  }
}
```

```
class Directory extends Dentry {

  protected List entries = new ArrayList();

  Directory(String name) {
    super(name);
  }

  public int size() {
    Iterator I = getEntries();
    int acc = 0;
    while (I.hasNext()) {
      acc += ((Dentry)I.next()).size();
    }
    return acc;
  }

  public void add(Dentry c)
    { entries.add(c); c.setParent(this); }

  public void remove(Dentry c)
    { entries.remove(c);}

  public Iterator getEntries()
    { return entries.iterator() ; }

}


class File extends Dentry {

    private int fsize;

    public File(String name, int s) {
          super(name);
          fsize = s;
    }

    public int size() { return fsize; }


}
```