Tutorial 2 - Composite Design Pattern

1) A directory for a file system contains two sorts of entry, files and directories. Using the Composite Design Pattern, identify the set of classes that can be used to represent directories. Develop an implementation in Java for each class. Both directories and files should implement the method **`int size()`** which will return the number of bytes required to store a file or for a directory, the total size of the set of files it contains.

2) Extend your implementation in 1) such that each entry in the directory system maintains a reference to its parent directory. Use this parent reference to implement a method **String pathName()** which returns the pathname of a file or directory.

Note:

For aggregation, use the following interfaces:

```
public interface List extends Collection {

      boolean add(Object o);    // add object to list

      boolean remove(Object o); // remove object from list

      Iterator iterator();      // return Iterator over list
}

public interface Iterator {

      boolean hasNext();   // true if more elements

      Object next();       // next element in list
}
```

`Usage:`

```
import java.util.*;
```

`Declaration:`

```
List aList = new ArrayList(10);
```

`Iterating over list:`

```
Iterator I =  aList.iterator();
while (I.hasNext()) {
    Object o = I.next();
}
```