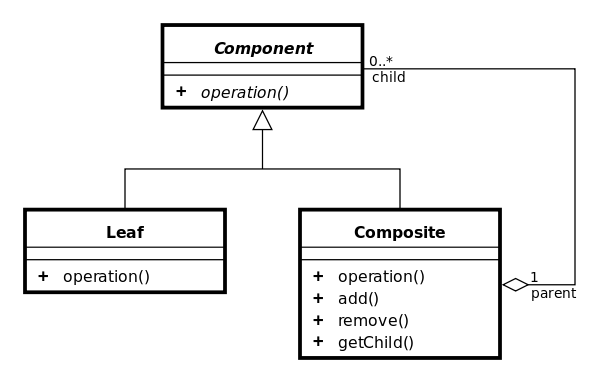
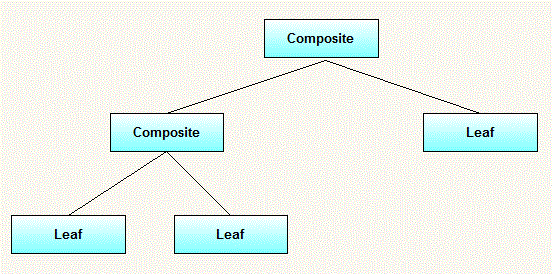
**Composite (Structural) Design Pattern**

**GOF : Compose objects into tree structure to represent part-whole hierarchies. Composite lets clients treat individual objects and composition of objects uniformly**.

# **Class Diagram**



Tree structure is given below.



Example : Directory and File, both have the ls() method. Java code is given below.

**class** File **implements** AbstractFile {

**private** String name;

**public** File(String name) {

**this**.name = name;

}

**public** **void** ls() {

System.***out***.println(name);

}

}

**import** java.util.ArrayList;

**public** **class** Directory **implements** AbstractFile {

**private** String name;

**private** ArrayList<AbstractFile> files = **new** ArrayList<AbstractFile>();

**public** Directory(String name) {

**this**.name = name;

}

**public** **void** add(AbstractFile f) {

files.add(f);

}

**public** **void** ls() {

System.***out***.println(name);

**for** (AbstractFile file : files) {

file.ls();

}

}

}

**public** **interface** AbstractFile {

**public** **void** ls();

}

The test program is given below.

**public** **class** Test {

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

Directory dirOne = **new** Directory("dir111");

Directory dirTwo = **new** Directory("dir222");

File a = **new** File("a");

File b = **new** File("b");

File c = **new** File("c");

File d = **new** File("d");

dirOne.add(a);

dirOne.add(dirTwo);

dirOne.add(b);

dirTwo.add(c);

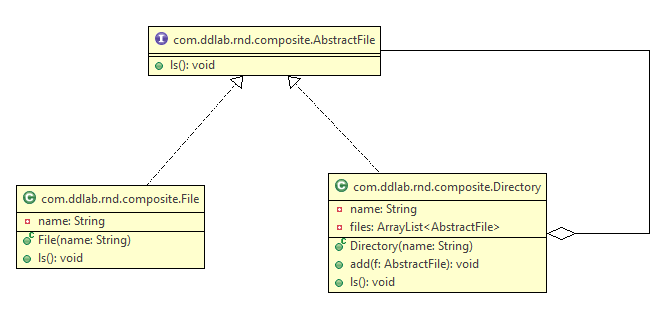
dirTwo.add(d);

dirOne.ls();

}

}

**The UML class diagram is given below.**



Another example : Manager and Employee.