**Composite Pattern**

그릇(복수)과 내용물(단수)을 동일시(= 혼합물, Composite)해서 재귀적인 구조 구축.

- Tree 구조의 Data(= 재귀적)
  - File System
  - Test Case 3개가 있을 때, 이를 모두 합쳐서 Test하고 싶은 경우
  - GUI 프로그래밍에서 View안에 자식 View를 갖는 경우

Architecture, Modeling, DesignPattern, Structural

**Component**

```
public abstract class Entry {
    public abstract String getName();
    public abstract int getSize();
    public Entry add(Entry entry) throws Exception {
        throw new Exception();
    }
    public void printList() {
        printList("");
    }
    protected abstract void printList(String prefix);

    @Override
    public String toString() {
        return getName() + " (" + getSize() + ")";
    }
}
```

**Leaf**

내용물. 내부에는 다른 것을 넣을 수 없음.

```
public class File extends Entry {

    private String name;
    private int size;
    public File(String name, int size) {
        this.name = name;
        this.size = size;
    }
    @Override
    public String getName() {
        return name;
```

```
    }

    @Override
    public int getSize() {
        return size;
    }

    @Override
    protected void printList(String prefix) {
        System.out.println(prefix + "/" + this);
    }

}
```

**Composite**

그릇. Leaf나 Composite 역할을 넣을 수 있음.

```
public class Directory extends Entry {

    private String name;
    private ArrayList<Entry> directory = new ArrayList<>();
    public Directory(String name) {
        this.name = name;
    }
    @Override
    public String getName() {
        return name;
    }

    @Override
    public int getSize() {
        return Stream.of(directory).mapToInt(entry -> entry.size()).sum();
    }
    /* Directory와 File을 같은 내용물로 간주 */
    public Entry add(Entry entry) {
        directory.add(entry);
        return this;
    }

    @Override
    protected void printList(String prefix) {
        System.out.println(prefix + "/" + this);
        directory.forEach(entry -> entry.printList(prefix + "/" + name));
    }

}
```

**Client**

```
public class Main {
    public static void main(String[] args) {
        try {
            Directory rootdir = new Directory("root");
            Directory bindir = new Directory("bin");
            Directory tmpdir = new Directory("tmp");
            Directory usrdir = new Directory("usr");
            rootdir.add(bindir);
            rootdir.add(tmpdir);
            rootdir.add(usrdir);
            bindir.add(new File("vi", 10000));
            bindir.add(new File("latex", 20000));
            rootdir.printList();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}


/*
/root (3)
/root/bin (2)
/root/bin/sub1 (10000)
/root/bin/sub2 (20000)
/root/tmp (0)
/root/usr (0)
*/
```

From:
https://ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
**https://ledyx.xyz/doku.php?id=design_pattern:composite_pattern&rev=1612667712**

Last update: **2026/10/24 07:03**