 Two use cases to demonstrate two structural design pattern.

1. **Voltage Adapter**

import java.util.function.Consumer;

// Target Interface (Expected by the client)

interface Device {

void supply220V();

}

// Adaptee (Existing system that provides 110V)

class PowerSource {

void supply110V() {

System.out.println("Supplying 110V power.");

}

}

// Adapter that converts 110V to 220V

class VoltageAdapter implements Device {

private PowerSource powerSource;

public VoltageAdapter(PowerSource powerSource) {

this.powerSource = powerSource;

}

@Override

public void supply220V() {

System.out.println("Adapter in action:");

System.out.println("Adapter converts 110V to 220V.");

}

}

public class Main {

public static void main(String[] args) {

PowerSource powerSource = new PowerSource();

Device adapter = new VoltageAdapter(powerSource);

System.out.println("Using device with adapter:");

useDevice(adapter);

}

public static void useDevice(Device device) {

device.supply220V();

}

}

1. **File System**

import java.util.ArrayList;

import java.util.List;

// Component interface

abstract class Component {

String name;

Component(String name) {

this.name = name;

}

abstract void show();

}

// Leaf (File)

class File extends Component {

File(String name) {

super(name);

}

@Override

void show() {

System.out.println("File: " + name);

}

}

// Composite (Directory)

class Directory extends Component {

private List<Component> children;

Directory(String name) {

super(name);

this.children = new ArrayList<>();

}

void addComponent(Component component) {

children.add(component);

}

@Override

void show() {

System.out.println("Directory: " + name);

for (Component child : children) {

child.show();

}

}

}

public class Main {

public static void main(String[] args) {

Directory root = new Directory("root");

File file1 = new File("file1.txt");

File file2 = new File("file2.txt");

Directory subdir = new Directory("subdir");

File file3 = new File("file3.txt");

root.addComponent(file1);

root.addComponent(file2);

root.addComponent(subdir);

subdir.addComponent(file3);

System.out.println("Showing file system:");

root.show();

}

}