Design Patterns

Composite

COMP3607
Object Oriented Programming II

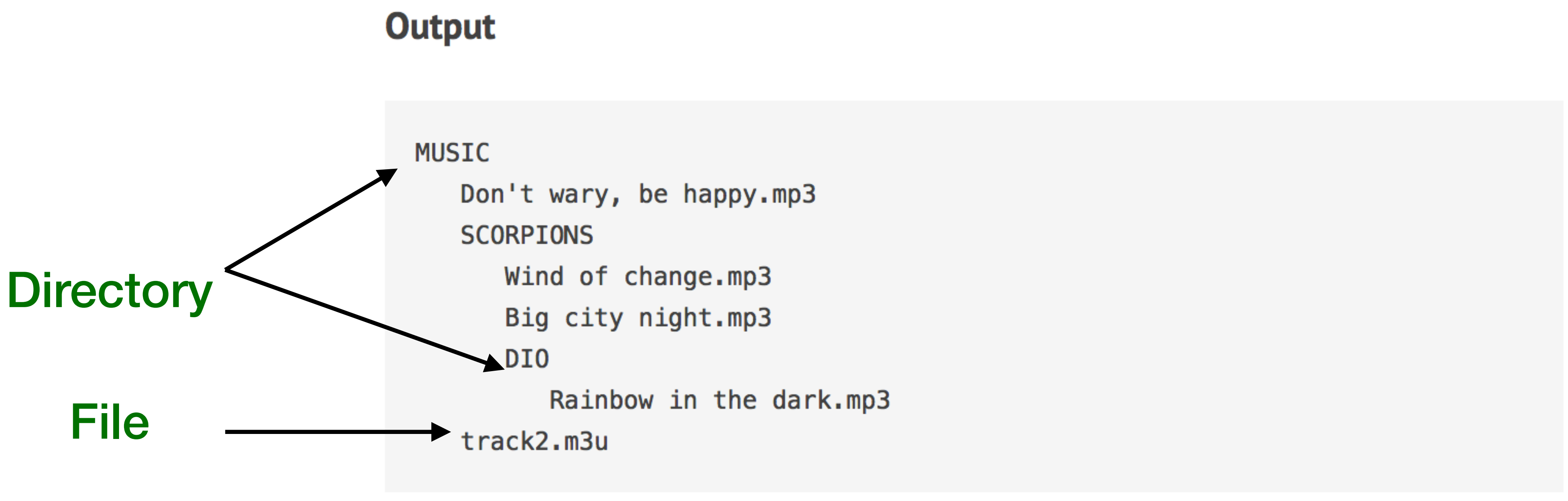
Week 6

Outline

- Composite Design Pattern

Exercise

Design a solution for listing all of the files and directories stored on a disk. Output should be presented as:



Output

```
MUSIC
   Don't wary, be happy.mp3
   SCORPIONS
      Wind of change.mp3
      Big city night.mp3
      DIO
         Rainbow in the dark.mp3
   track2.m3u
```

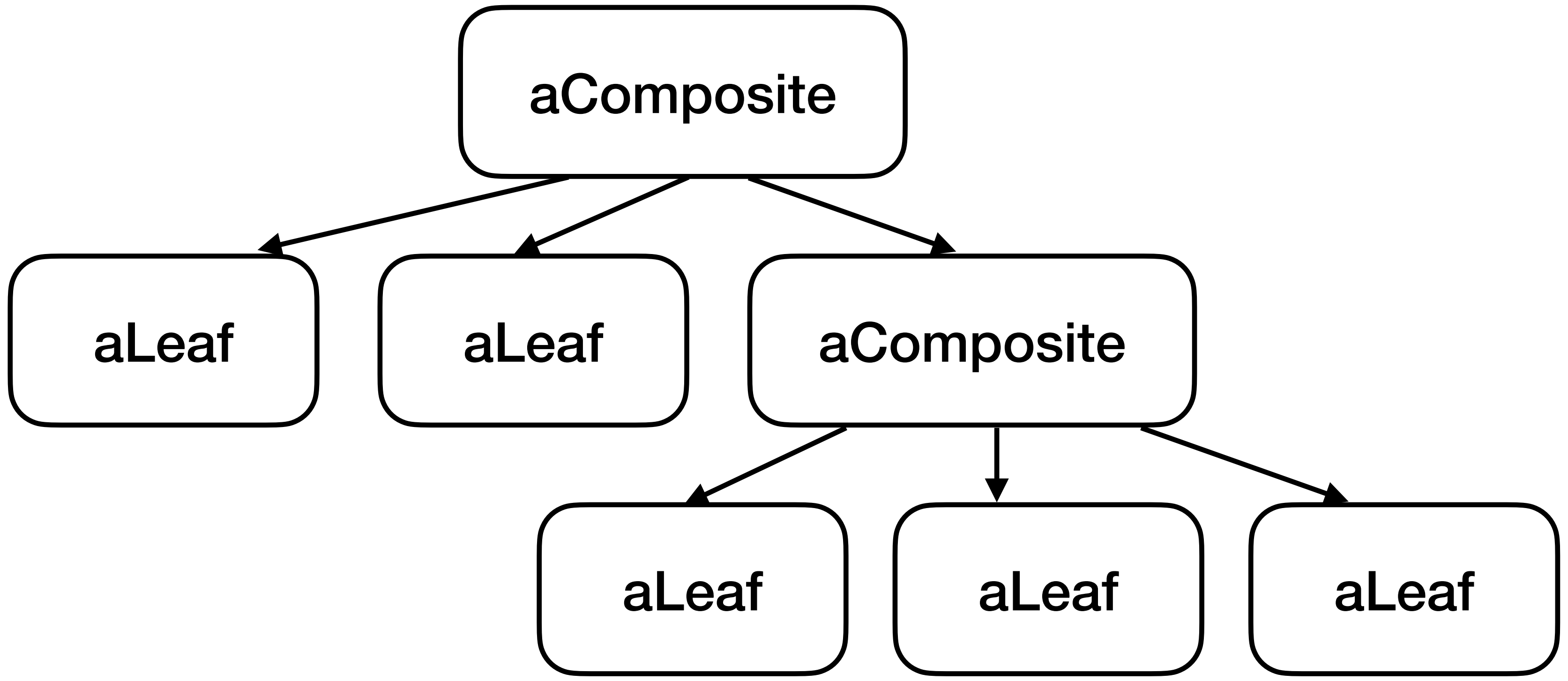
2

https://sourcemaking.com/design_patterns/composite/java/1

Composite Design Pattern

The Composite design pattern is used to compose objects into tree structures to represent part-whole hierarchies.

It lets clients treat individual objects and compositions of objects uniformly.



2

Problem Scenario

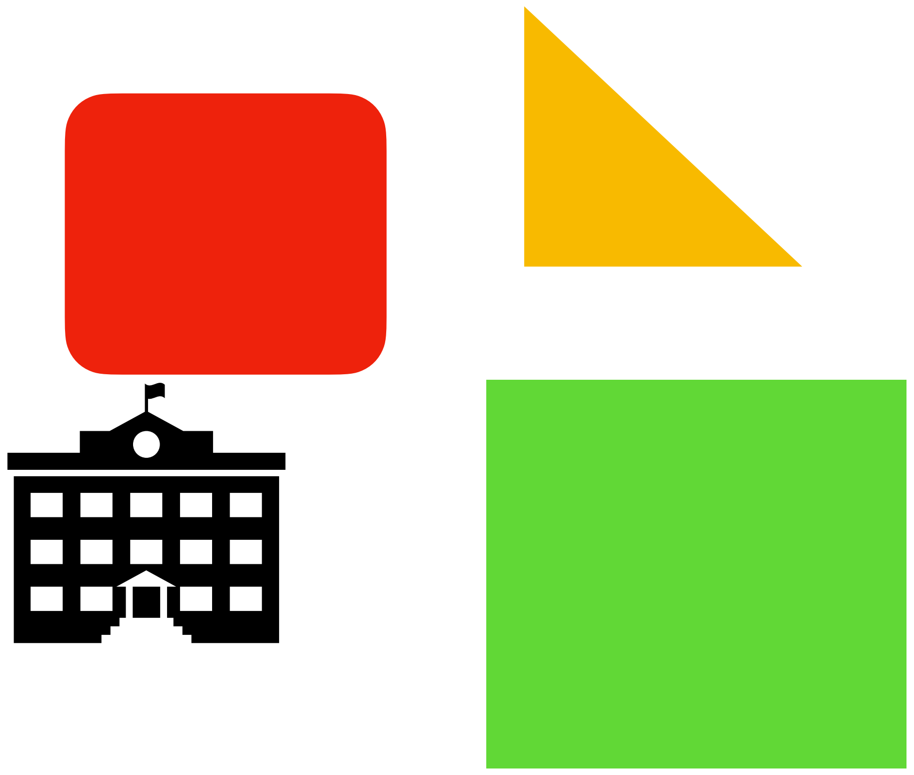
Suppose we are building a graphics application (drawing editor) to let users build complex diagrams out of simple components.

Users can group components to make larger ones which can be grouped further to make still larger ones and so on.

How can we achieve this?

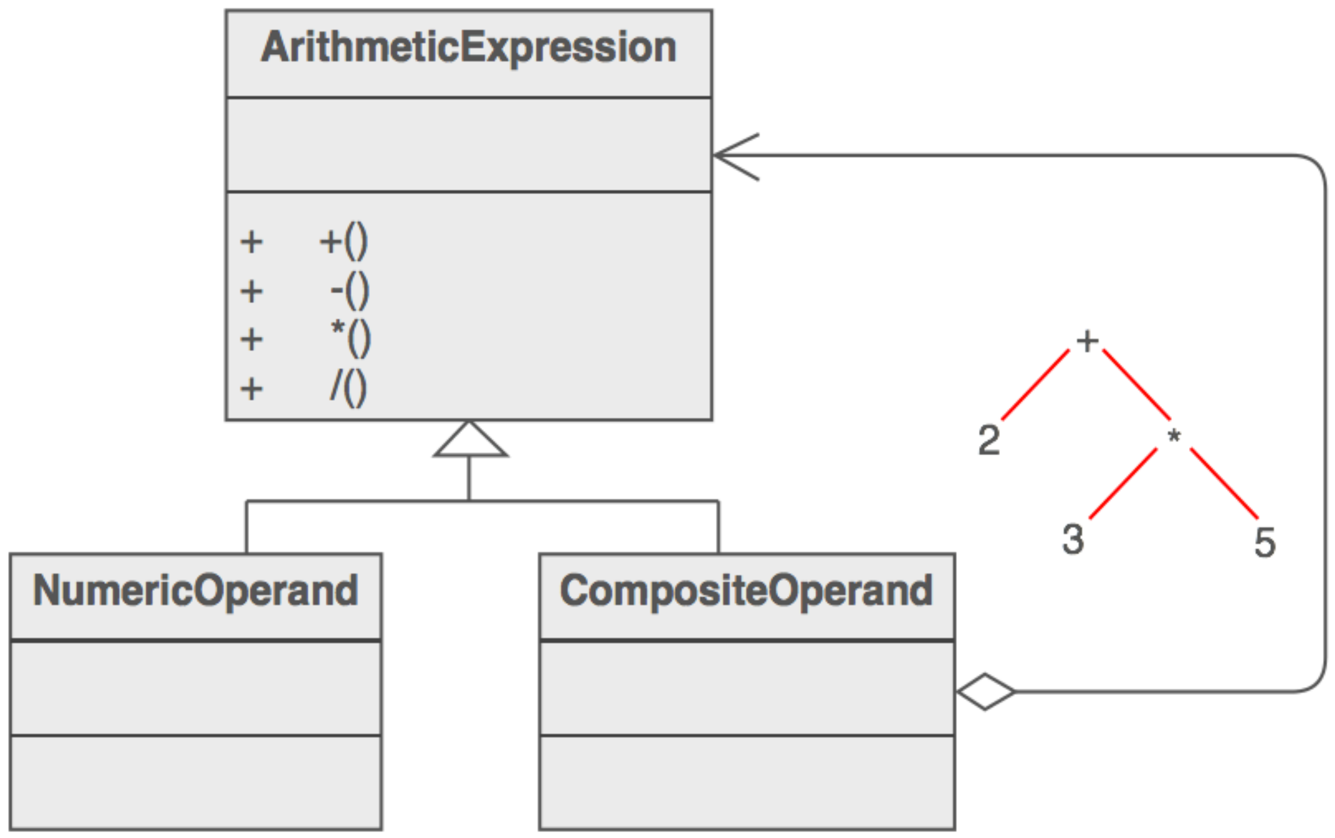
Possible options:

- Simple implementation: Use classes for Graphical Primitives (Lines, Text) and other container classes for these primitives. Problem?
- An abstract class that represents both primitives and their containers.



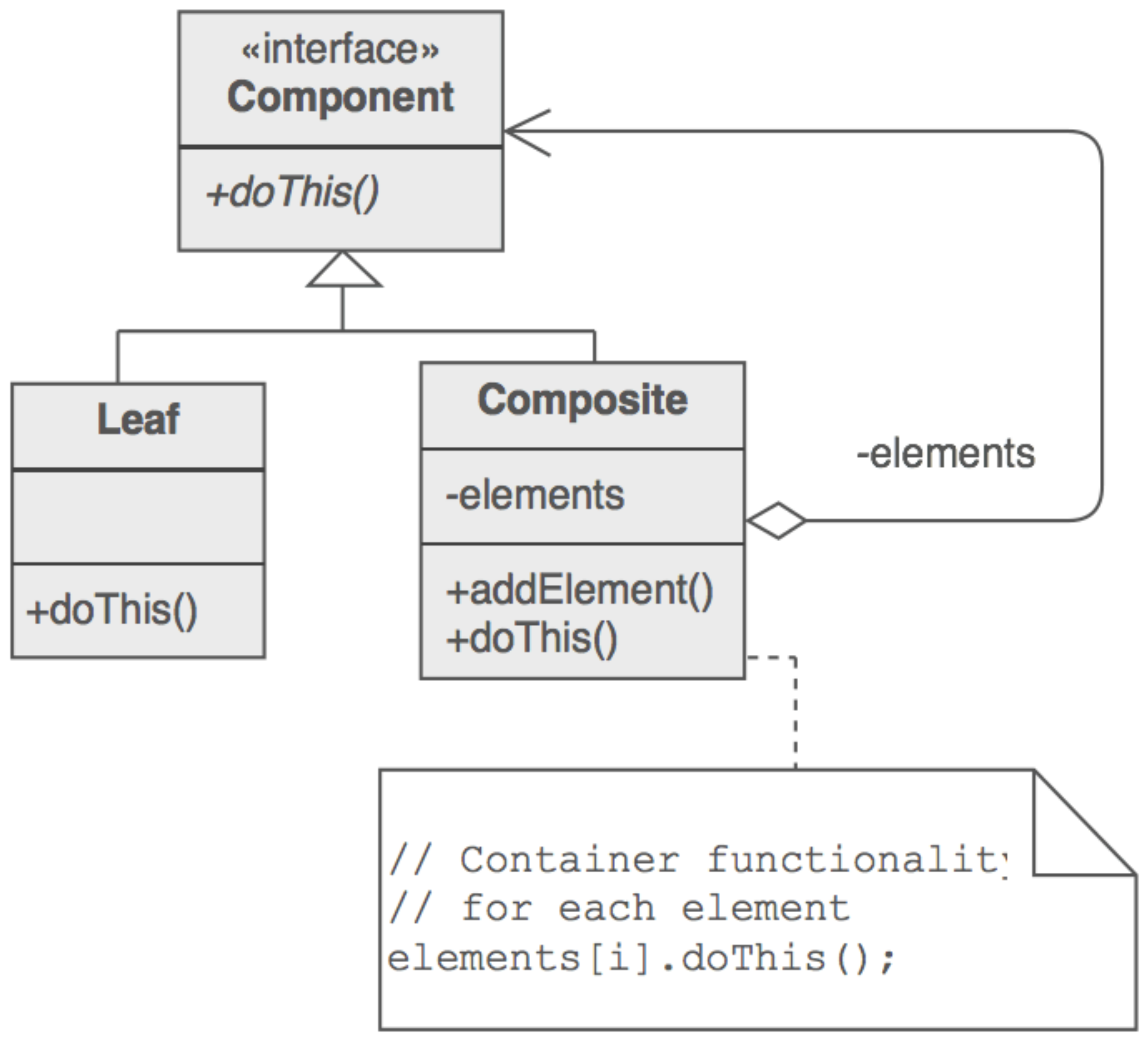
Another Example

The Composite composes objects into tree structures and lets clients treat individual objects and compositions uniformly. Although the example is abstract, arithmetic expressions are Composites. An arithmetic expression consists of an operand, an operator (+ - * /), and another operand. The operand can be a number, or another arithmetic expression. Thus, 2 + 3 and (2 + 3) + (4 * 6) are both valid expressions.



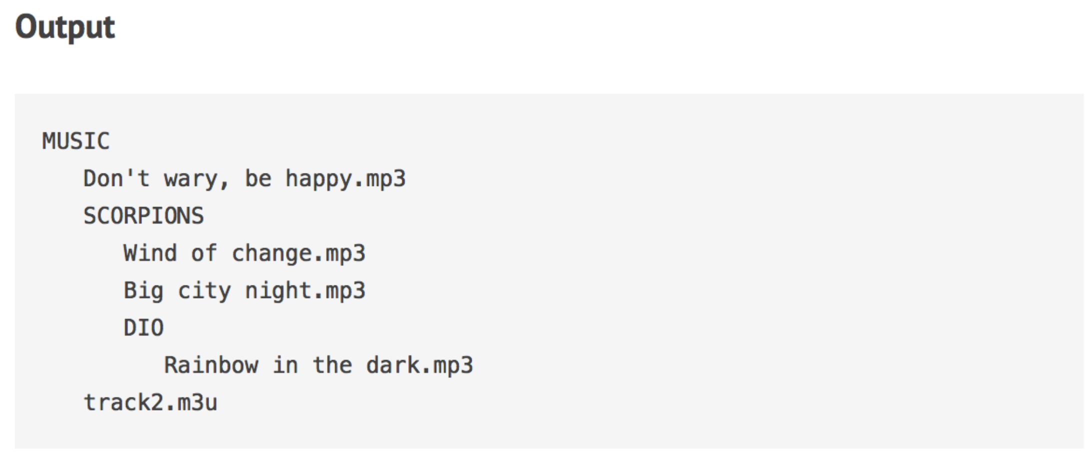
https://sourcemaking.com/design_patterns/composite

Composite Design Pattern (UML)



2

Code Example



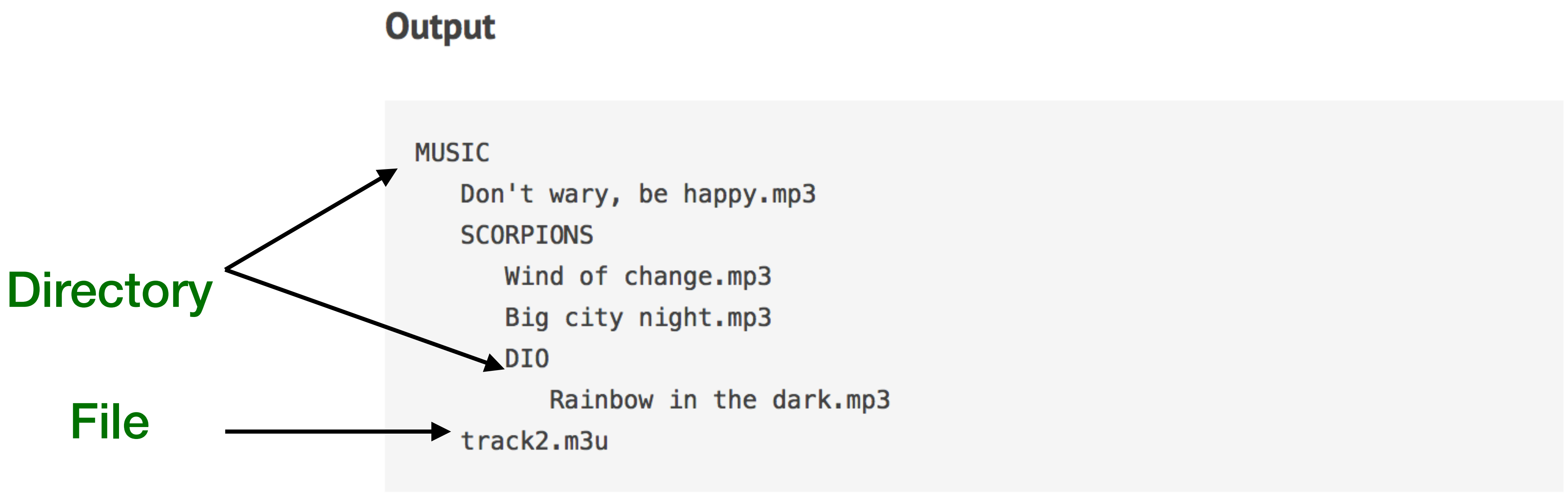
Output

```
MUSIC
  Don't wary, be happy.mp3
  SCORPIONS
    Wind of change.mp3
    Big city night.mp3
    DIO
      Rainbow in the dark.mp3
  track2.m3u
```

```
public interface Component{
     public abstract void doThis(); //common operation in hierarchy
}
public class Composite implements Component{
     private Collection <Component> toDo; //initialised in constructor
     public void addComponent(Component c){
        //code to add ‘c’ to the Collection
     }
     public void doThis(){
        //do something useful with the components in Collection (iterate)
     }
}
public class Leaf implements Component{
    public void doThis(){ //something happens here in Leaf
     }
}
```

Exercise

Design a solution for listing all of the files and directories stored on a disk using the Composite Design Pattern



Output

```
MUSIC
   Don't wary, be happy.mp3
   SCORPIONS
      Wind of change.mp3
      Big city night.mp3
      DIO
         Rainbow in the dark.mp3
   track2.m3u
```

2

https://sourcemaking.com/design_patterns/composite/java/1

Applicability: Composite Pattern

The Composite pattern is used when:

- You want to represent part-whole hierarchies of objects
- You want clients to be able to ignore the difference between compositions of objects and individual objects. Clients will treat all objects in the composite structure uniformly.

2

Consequences: Composite Pattern

Benefits and Liabilities

- Defines class hierarchies consisting of leaf and composite objects. Complex objects can be composed of leaf objects recursively.
- Makes clients simple: they can treat composite and leaf objects uniformly.
- Easier to add new kinds of components (as leaf or composite classes)
- Harder to restrict components of a composite object using types; run-time checks needed instead.

2

References

- UML: online reading resources
  - www.uml.org (Documentation, notation)
  - www.omg.org (Standards, protocols)
- UML Modelling and Drawing Tool:
  - http://staruml.io/