



# Deodorizing Primitive Obsession



Dario Campagna

Head of Research and Development

# Implicit Tree

You implicitly form a tree structure, using a primitive representation, such as String.

- Bloated code, difficult to work with
- Coupling between the code that builds the tree and how the tree is represented
- Issues related to primitive obsession

```
xml.append("<orders>");
for (int i=0; i < orders.getOrderCount(); i++) {
    Order order = orders.getOrder(i);
    xml.append("<order>");
    xml.append(" id='");
    xml.append(order.getOrderld());
    xml.append("'>");
    for (int j=0; j < order.getProductCount(); j++) {
        Product product = order.getProduct(j);
        xml.append("<product>");
        xml.append(" id='");
        xml.append(product.getID());
        xml.append("'");
        xml.append(" color='");
        xml.append(getColorFor(product));
        xml.append("'");
        if (product.getSize() != ProductSize.NOT _APPLICABLE) {
            xml.append(" size='");
            xml.append(getSizeFor(product));
            xml.append("'");
        }
        xml.append(">");
        xml.append("<price>");
        xml.append(" currency='");
        xml.append(getCurrencyFor(product));
        xml.append("'>");
        xml.append(product.getPrice());
        xml.append("</price>");
        xml.append(product.getName());
        xml.append("</product>");
    }
    xml.append("</order>");
}
xml.append("</orders>");
```

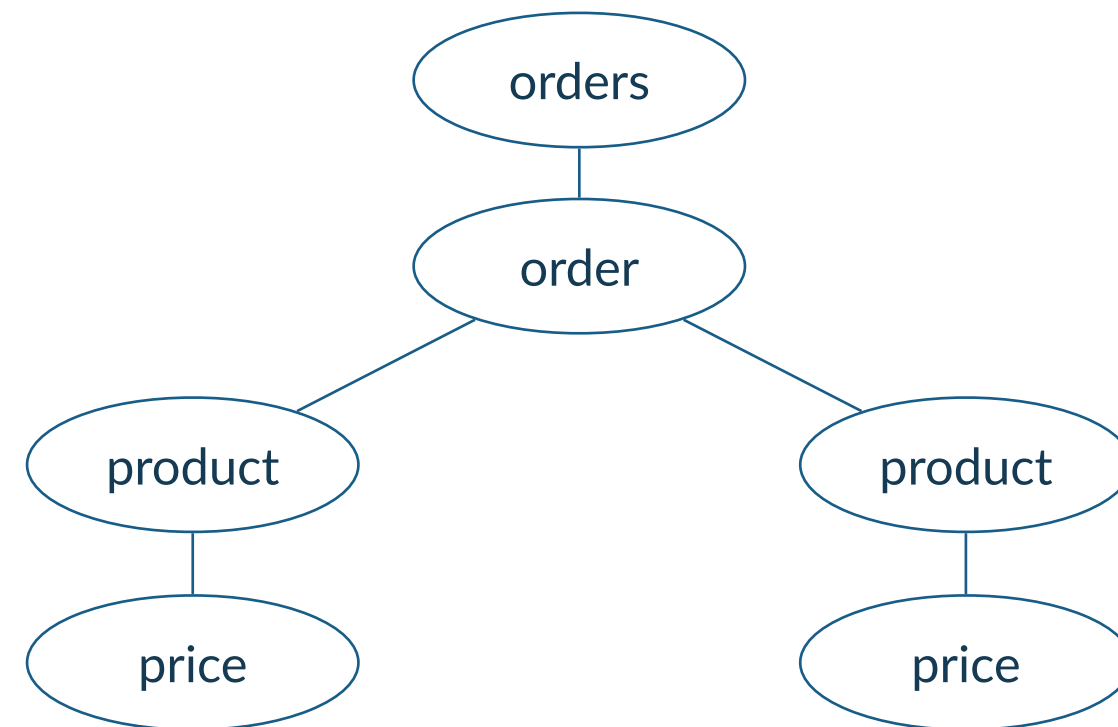


# Implicit Tree

Data or code forms an implicit tree when it's not explicitly structured as a tree but may be represented as a tree.

```
String expectedResult =
"<orders>" +
  "<order id='321'>" +
    "<product id='f1234' color='red' size= 'medium'>" +
      "<price currency='USD'>" +
        "8.95" +
      "</price>" +
      "Fire Truck" +
    "</product>" +
    "<product id='p1112' color='red'>" +
      "<price currency= 'USD'>" +
        "230.0" +
      "</price>" +
      "Toy Porsche Convertible" +
    "</product>" +
  "</order>" +
"</orders>";
```

The code in the previous slide outputs strings representing XML data.



Tree structure of the XML data may be represented as a tree.



# Composite

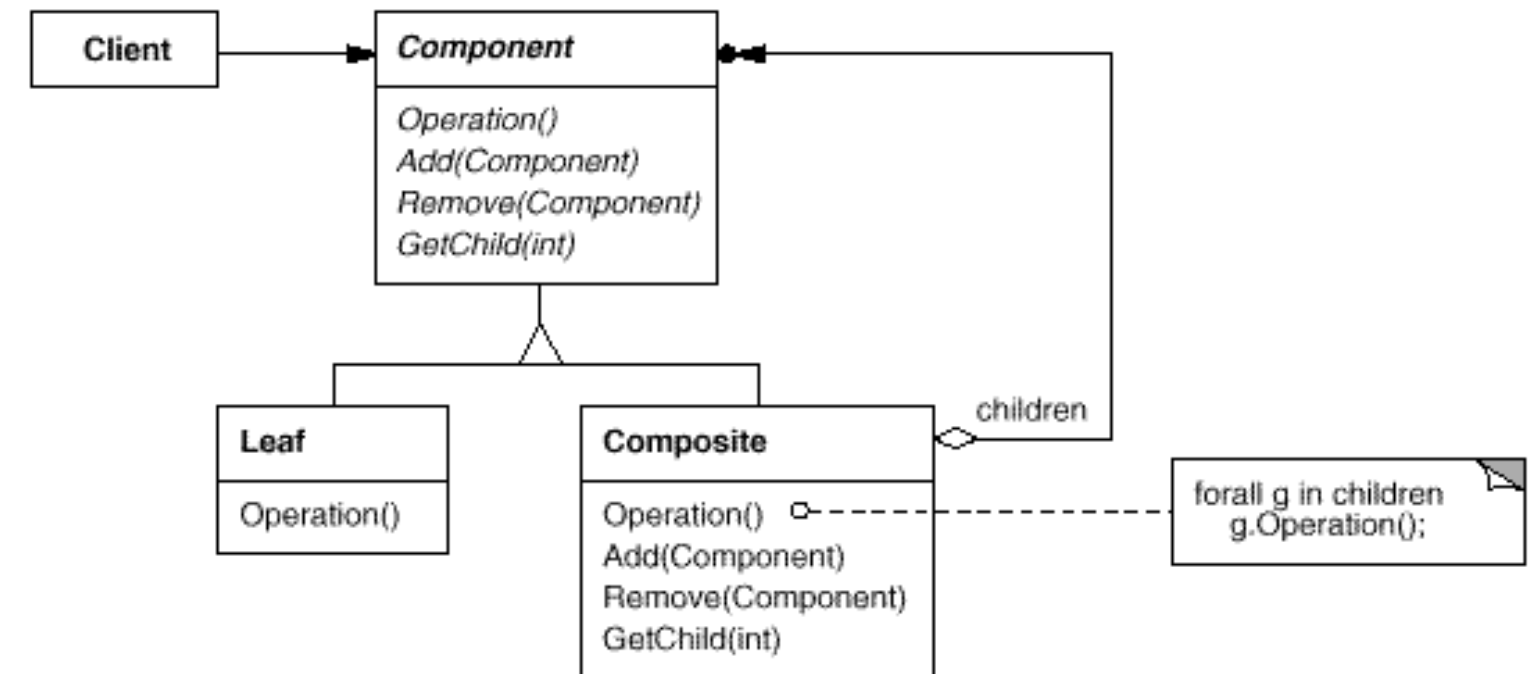
Compose objects into tree structures to represent part-whole hierarchies. Composite lets client treat individual objects and compositions of objects uniformly.

## Motivation

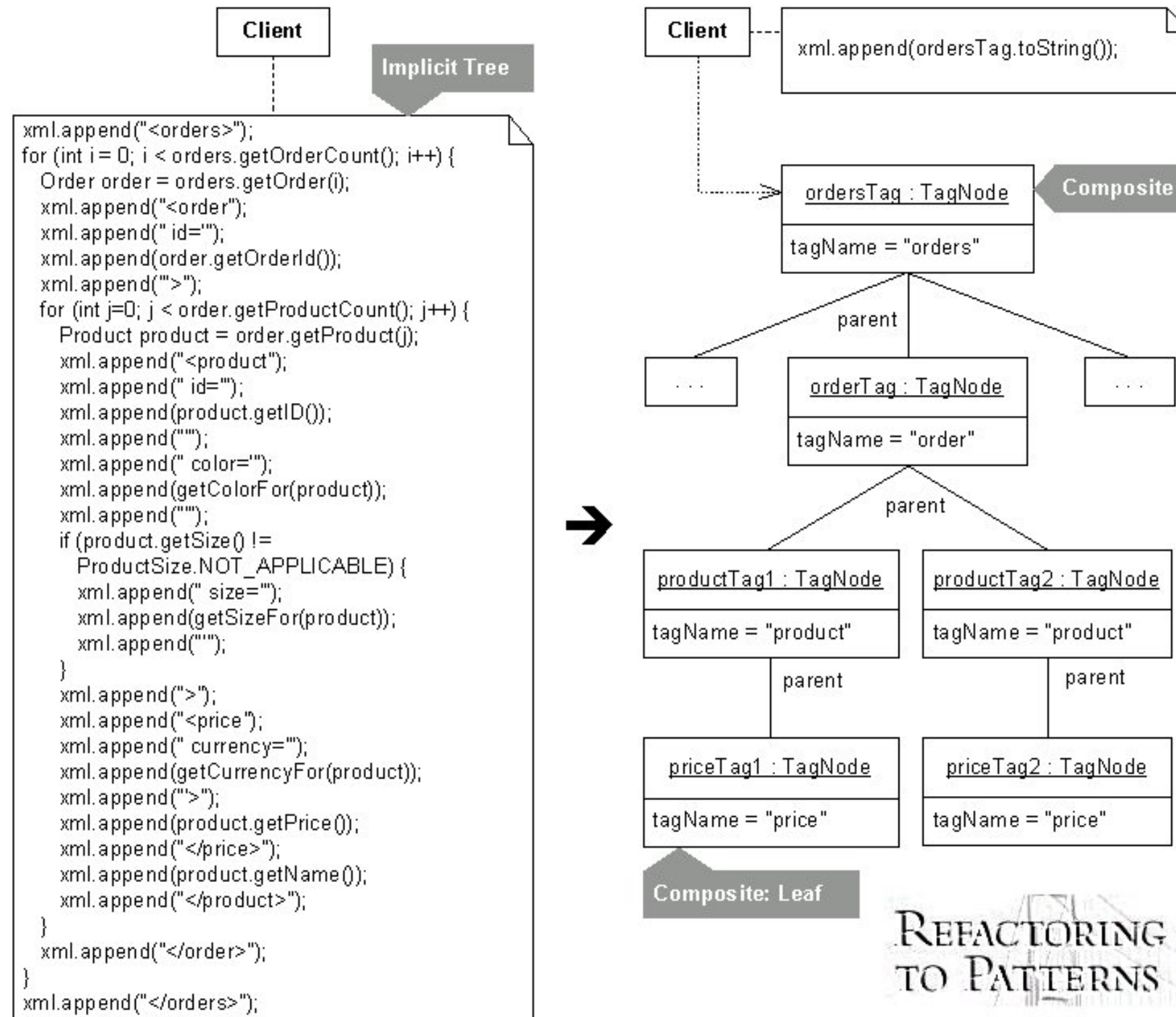
- Graphic application for building complex diagrams out of simple components.
- Application that manages tree-like object structures.

## Applicability

- Representation of part-whole hierarchies of objects.
- Clients able to ignore the difference between composition of objects and individual objects.



# Replace Implicit Tree with Composite



# Replace Implicit Tree with Composite

Benefits	Liabilities
Encapsulate repetitive instructions like formatting, adding, or removing nodes.	Complicates a design when its simpler to construct an implicit tree.
Provides a generalized way to handle a proliferation of similar logic.	
Simplifies construction responsibilities of a client.	





# Replace Implicit Tree with Composite – Mechanics

1. Identify an **implicit leaf** and create a **leaf node** class by applying refactorings or TDD.
  - ✓ Compile and test
2. Replace every occurrence of the implicit leaf with an instance of the leaf node.
  - ✓ Compile and test that the implicit tree still functions correctly
3. Repeat 1 and 2 for any other implicit leaf. Make sure all leaf node share a common interface.
4. Identify an **implicit parent** and create a **parent node** class by applying refactorings or TDD.
  - ✓ Compile and test that the implicit tree still functions correctly
5. Replace every occurrence of the implicit parent with code that uses a parent node instance.
6. Repeat 4 and 5 for any other implicit parents.



# Replace Implicit Tree with Composite – Example

Let's apply this refactoring to the `OrdersWriter` class in the *implicit-tree* branch of the following repository.

<https://github.com/dario-campagna/replace-implicit-tree-with-composite>

- Example from Refactoring to Patterns
- Code comes from a shopping system

