Composite Design Pattern

Boudewijn, Daniel, Jacco, René

2019-09-30

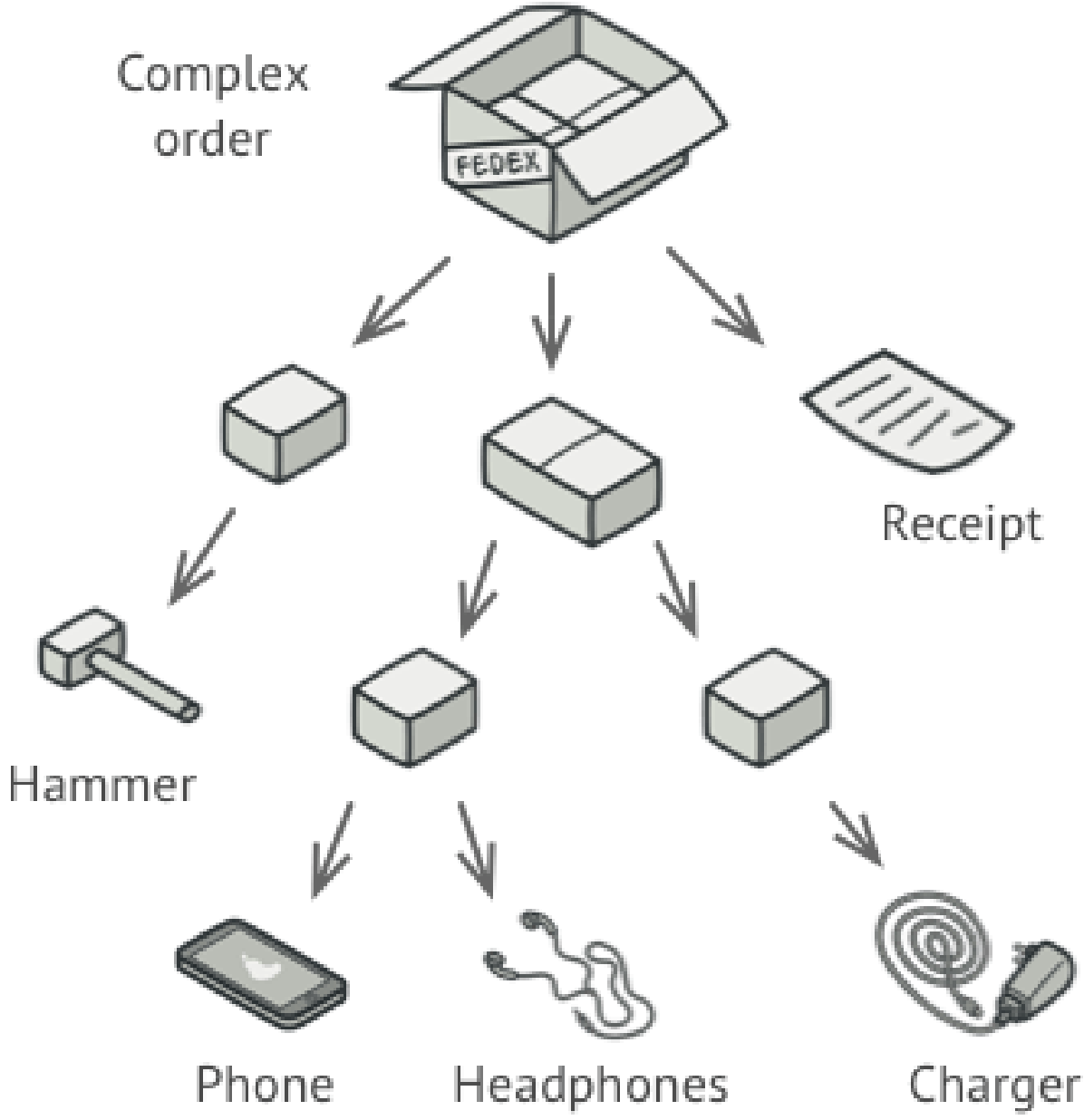
Catalog - Jacco

Name	Composite Pattern
Intent	Composite objects into tree structures
Motivation	Treat Branches and Leaves uniformly
Applicability	Part-Whole hierarchy
Structure	Recursive composition of primitive objects into composite objects
Participants	Component, Leaf, Composite, Client
Collaborations	?
Consequences	?
Implementation	?
Examples	?
Related patterns	Decorator

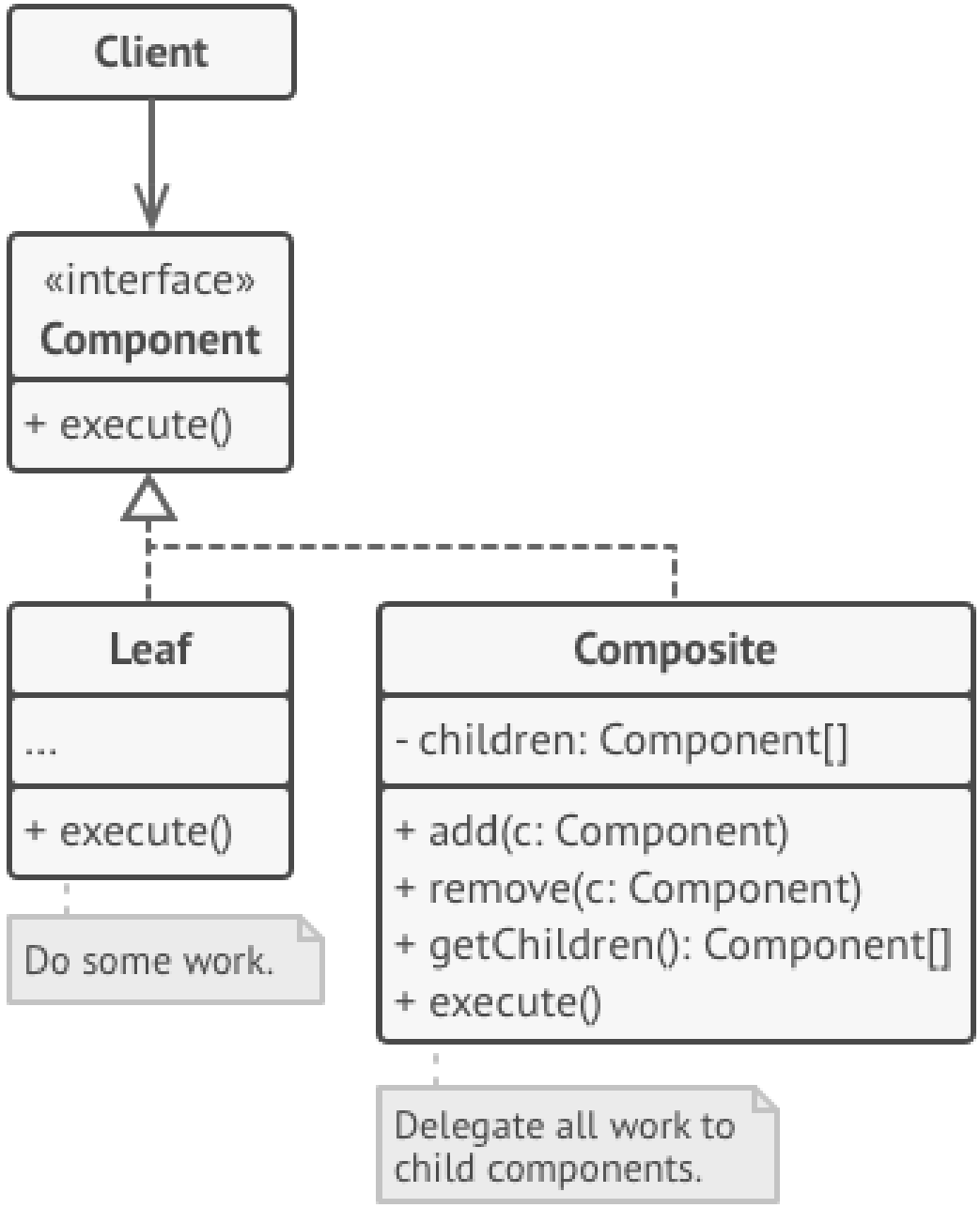
Participants

Component	declares the interface for objects in the composition and for accessing and managing its child components. It also implements default behavior for the interface common to all classes as appropriate.
Leaf	defines behavior for primitive objects in the composition. It represents leaf objects in the composition.
Composite	stores child components and implements child related operations in the component interface.
Client	manipulates the objects in the composition through the component interface.

UML - Daniel


Complex
order
FEDEX
Receipt
Hammer
Phone
Headphones
Charger

UML - Daniel


Client
«interface»
Component
+ execute()
Leaf
...
+ execute()
Do some work.
Composite
- children: Component[]
+ add(c: Component)
+ remove(c: Component)
+ getChildren(): Component[]
+ execute()
Delegate all work to
child components.

ChessBoardLauncher - Boudewijn

```
public class ChessBoardLauncher {
    public static void main(String[] args) {
        ChessBoard boudysCB = new ChessBoard();
        boudysCB.setupChessBoard();
        System.out.println(boudysCB);
    }
}
```

ChessBoard

```
public class ChessBoard {
    private List<Field> fields;
    private List<Piece> pieces;

    public ChessBoard() {
        super();
    }

    public ChessBoard(List<Field> fields, List<Piece> pieces) {
        super();
        this.fields = fields;
        this.pieces = pieces;
    }
    public void setupChessBoard() {
        this.setupFields();
    }

    private void setupFields() {
        fields = new ArrayList<>();
        boolean isWhite = false;
        for (int i = Rank.getMinRank(); i <= Rank.getMaxRank(); i++) {
            for (char line : Line.getLines()) {
                fields.add(new Field(new Rank(i), new Line(line)));
            }
```

Field

```
public class Field {
    private boolean isWhite;
    private Rank rank;
    private Line line;

    public Field(Rank rank, Line line) {
        this(false, rank, line);
    }

    public Field(boolean isWhite, Rank rank, Line line) {
        super();
        this.isWhite = isWhite;
        this.rank = rank;
        this.line = line;
        setColor(setOctals());
    }
    public void setColor(int[] octals){
        this.setWhite(octals[0] % 2 != octals[1] % 2);
    }

    public int[] setOctals(){
        int[] octals = new int[2];
        octals[0] = (int)line.getLine() - 65;//A->0,B->1,...,H->7
        octals[1] = rank.getRank() - 1;//0..7
        return octals;
    }
```

Rank

```
public class Rank {
    private static final int[] ranks = {1, 8};
    private static final int MIN_RANK = ranks[0];
    private static final int MAX_RANK = ranks[1];
    private int rank;
    public boolean isRank(){
        for ( int rank : ranks ) {
          if (MIN_RANK <= rank && rank <= MAX_RANK) {
              return true;
          }
        }
        return false;
    }
```

Line

```
public class Line {
    private static final char[] lines = {'A','B','C','D','E','F','G','H'
        };
    private char line;

    public Line(char line) {
        this.line = Character.toUpperCase(line);
    }
    public boolean isLine(){
        for ( char line : lines ) {
            if (line == this.line) {
                return true;
            }
        }
        return false;
    }
```

Piece

```
public class Piece {
    private PieceName pieceName;
    private Field field;
    private boolean isBlack;//otherwise white

    public Piece(PieceName pieceName, Field field, boolean isBlack) {
        this.pieceName = pieceName;
        this.field = field;
        this.isBlack = isBlack;
    }
    public void setField(boolean isBlack) {
        this.isBlack = isBlack;
    }
```

PieceName

```
public enum PieceName {
    KING("King","K",100,1),
    QUEEN("Queen","Q",9,1),
    ROOK("Rook","R",5,2),
    BISHOP("Bishop","B",3,2),
    KNIGHT("Knight","K",3,2),
    PAWN("Pawn","P",1,8);

    private String fullName
    , abbrName;
    private int value
    , number;

    PieceName(String fullName, String abbrName, int value, int number) {
        this.fullName = fullName;
        this.abbrName = abbrName;
        this.value = value;
        this.number = number;
    }
```

Assignment - René