

## APPCs without traversals (in pure Java)

```
APPC CompositePattern {  
  
    interface Component {  
        public void operation ();  
    }  
  
    interface Leaf extends Component {}  
  
    personality Composite extends Component {  
        public iterationOperation {  
            java.util.Iterator children = getChildren();  
            while (children.hasNext()) {  
                try {  
                    Component c = (Component) children.next();  
                    c.operation();  
                    inner(c);  
                }  
                catch (Throwable exception) {  
                    System.err.println(„ „);  
                }  
            }  
        }  
        protected void inner(Component);  
        protected java.util.Iterator getChildren();  
    }  
}  
  
APPC Summing extend CompositePattern {  
    interface Component {  
        public float getGross();  
        protected void setGross(float);  
    }  
  
    personality Leaf {  
  
        public void operation() {  
            setGross(getNet());  
        }  
        protected float getNet();  
    }  
  
    personality Composite {  
        public void inner(Component c) { // the most recent Component  
            setGross(getGross() + c.getGross());  
        }  
    }  
}
```


```

APPC LimitChecking {
    personality LimitedParty {
        public boolean overdrawnCapacity () {
            if (getCapacity() < getActualGross()) return true
            else return false;
        }

        protected float getCapacity();
        protected float getActualGross();
    }
}

```

Let us now connect:

1. All three, Summing, LimitChecking, and  David's CCG
2. First Summing and LimitChecking in a compound APPC and then the latter with David's CCG

1.

APPC PackagingWithSummingAndCapacityChecking connects Packaging, Summing, LimitChecking {

```

Summing.Component personifiedBy (or playedBy) Packaging.Box {
    public float getGross() {
        return weight; }
    protected void setGross(float w) {
        weight = w; }
}

```



```

Summing.Leaf personifiedBy Packaging.Item {
    protected float getNet() {
        return weight; }
}

```

```

Summing.Composite personifiedBy Packaging.Paper {
    protected getChildren() {
        ...
        // or to Item;
    }
}

```

```

LimitChecking.LimitedItem personifiedBy {
    Packaging.Paper {
        getCapacity() { return capacity; }
    }

    Summing.Composite {
        getActualGross() { operation(); }
    }
}

```



2.

```
APPC SummingCapacityChecking connects Summing, CapacityChecking {  
  
    LimitChecking.LimitedItem personifiedBy Summing.Composite {  
        protected float getActualGross() {  
            operation();  
        }  
    }  
}
```

APPC PackagingWithSummingAndCapacityChecing connects Packaging,  
SummingCapacityChecking {



```
    Summing.Component personifiedBy Packaging.Box {  
  
        public float getGross() {  
            return weight;  
        }  
        protected void setGross(float w) {  
            weight = w; }  
    }  
}
```

