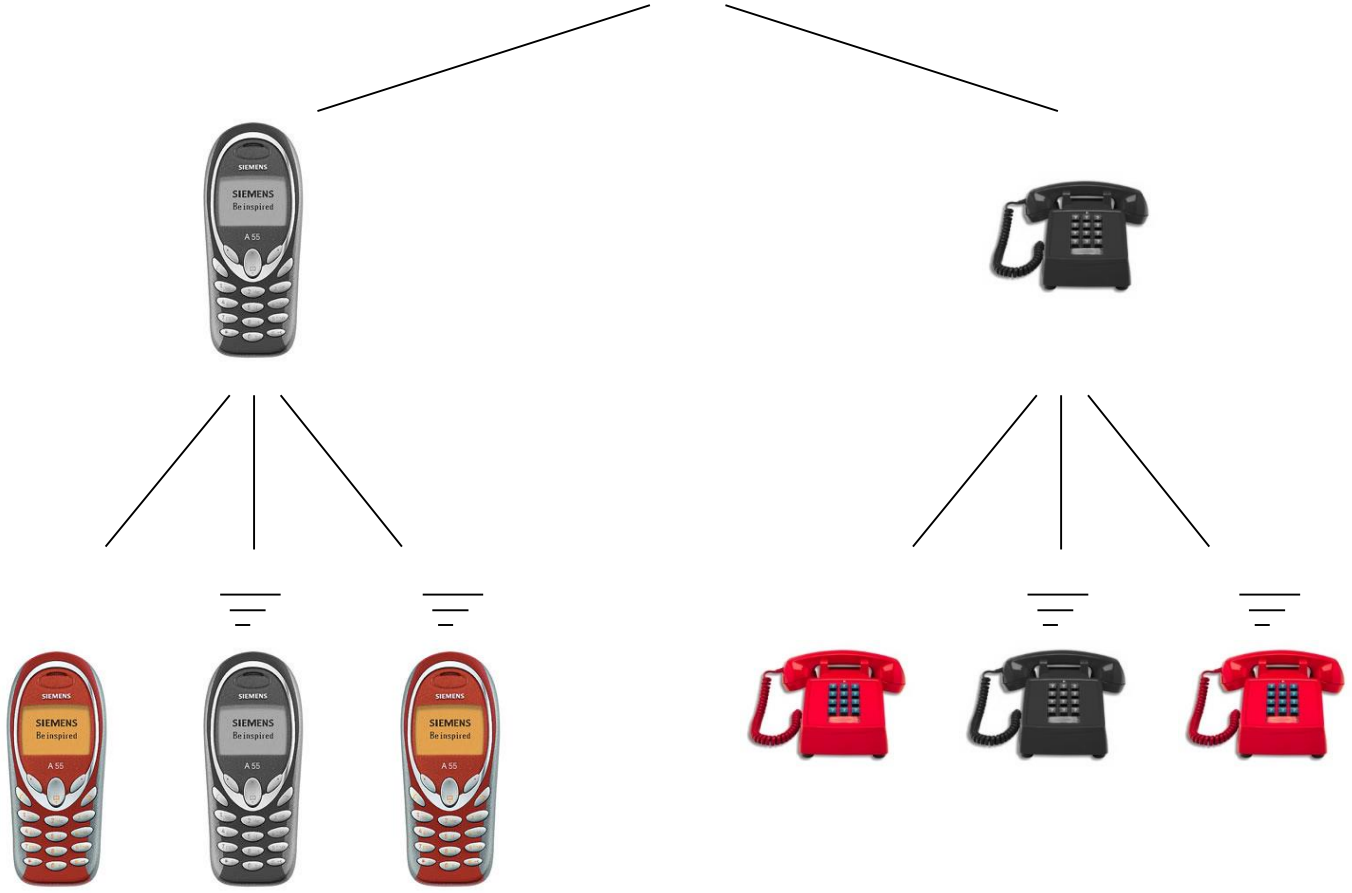
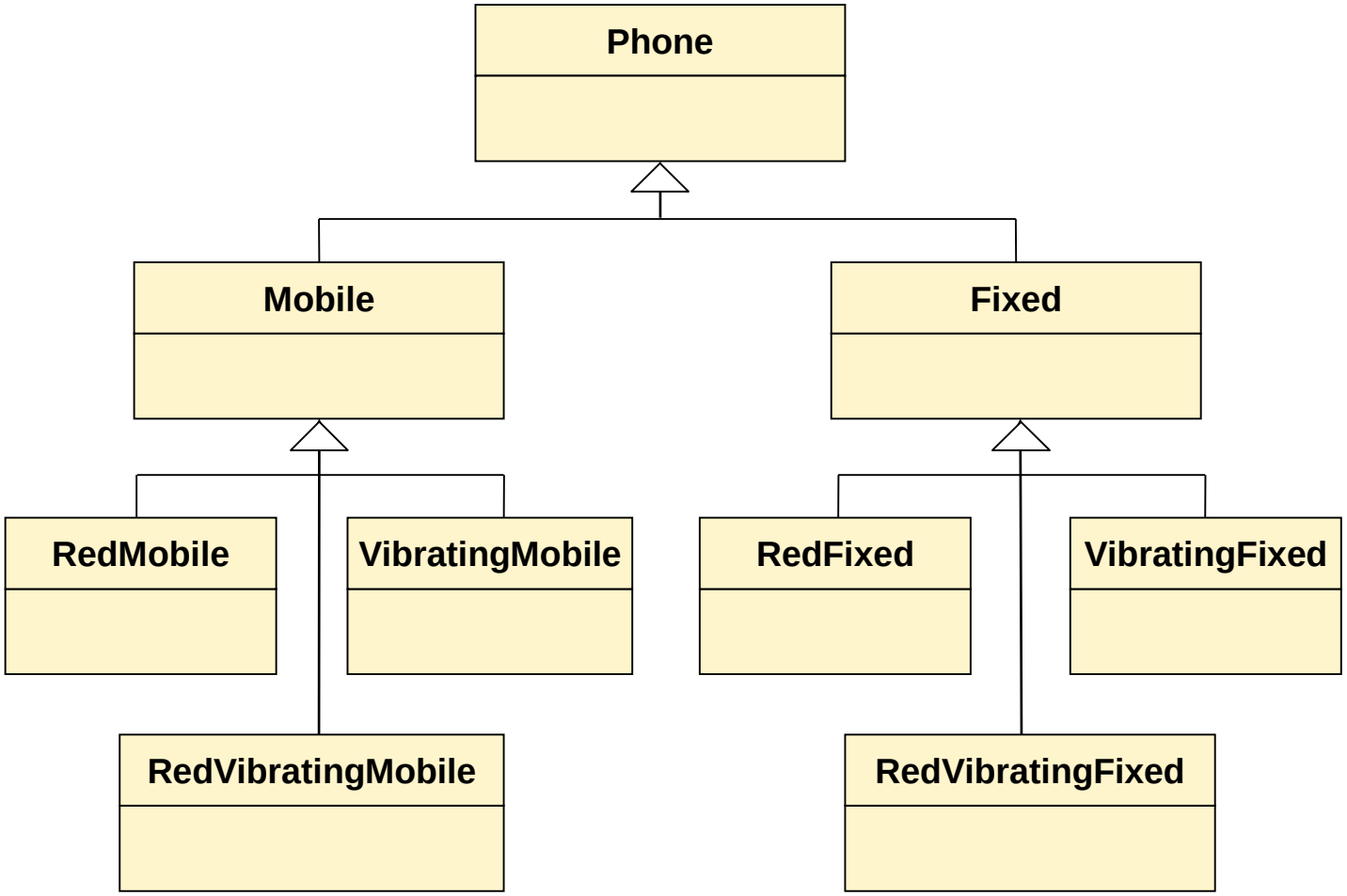
Decorator Pattern

- **Motivation**
- **Structure**

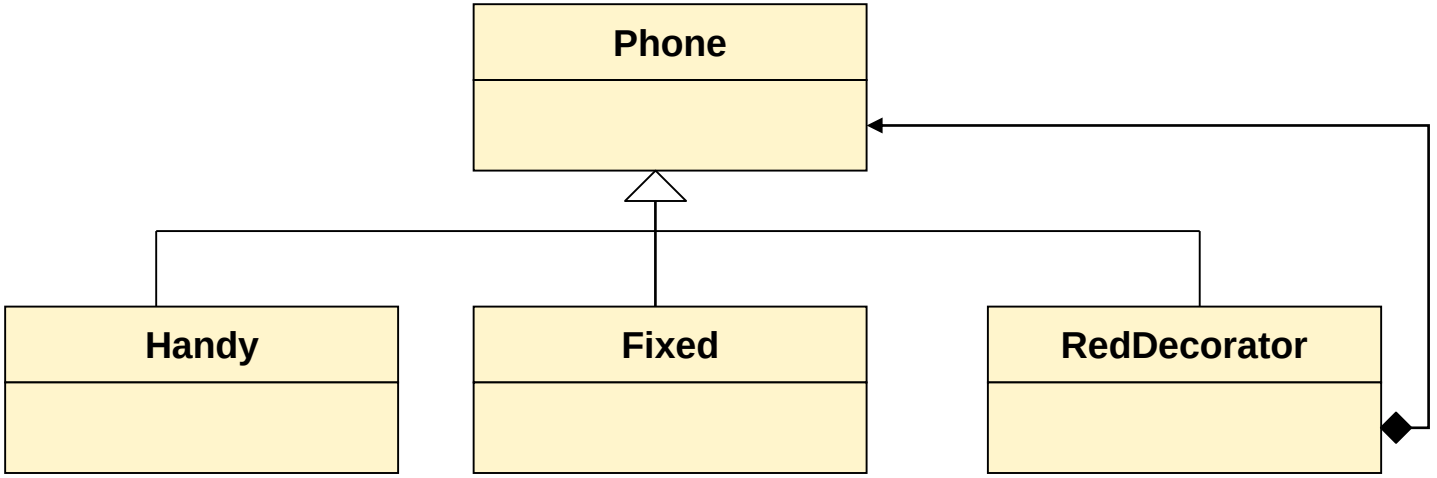
Vibrating & Skin Phones



Implementation with Inheritance



Implementation with a Decorator



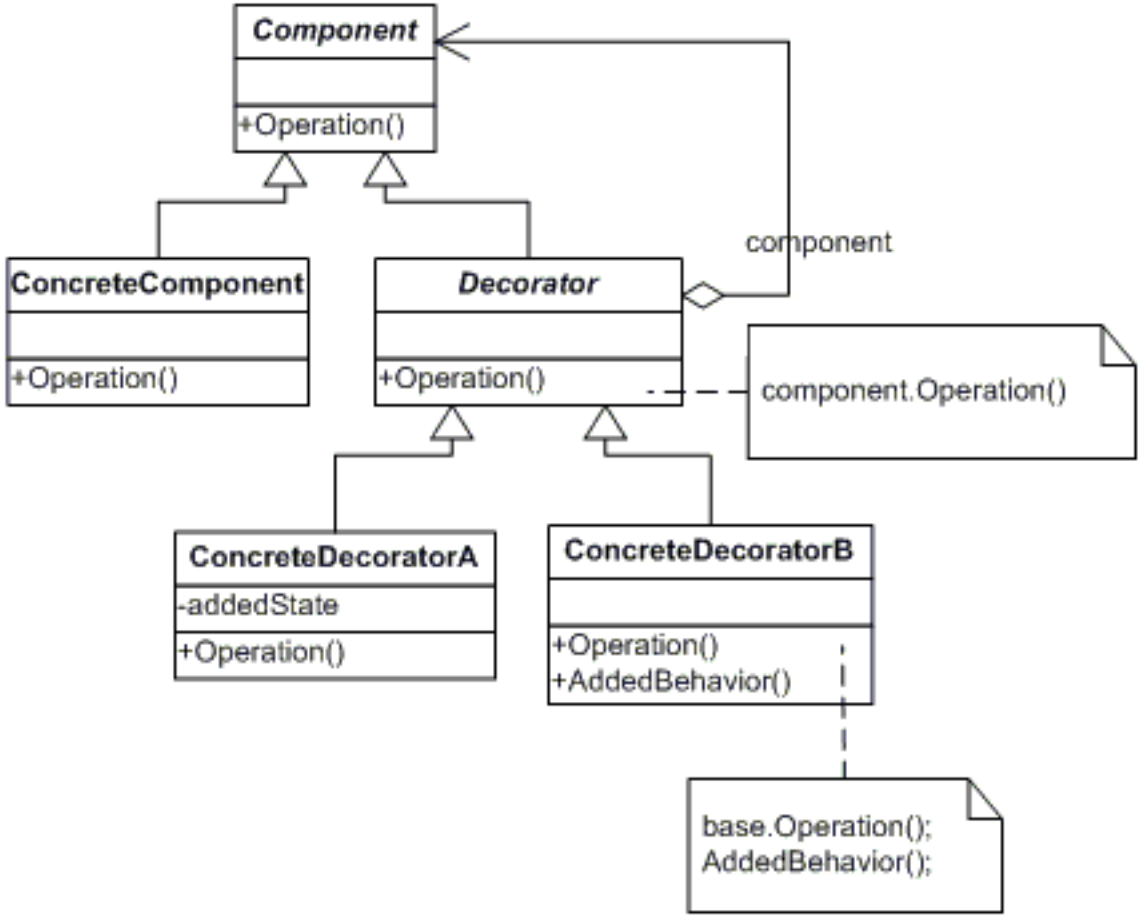

```
Phone myHandy = new RedDecorator(new Handy());
myHandy.dial("044 123 45 67");
```

Implementation with a Decorator

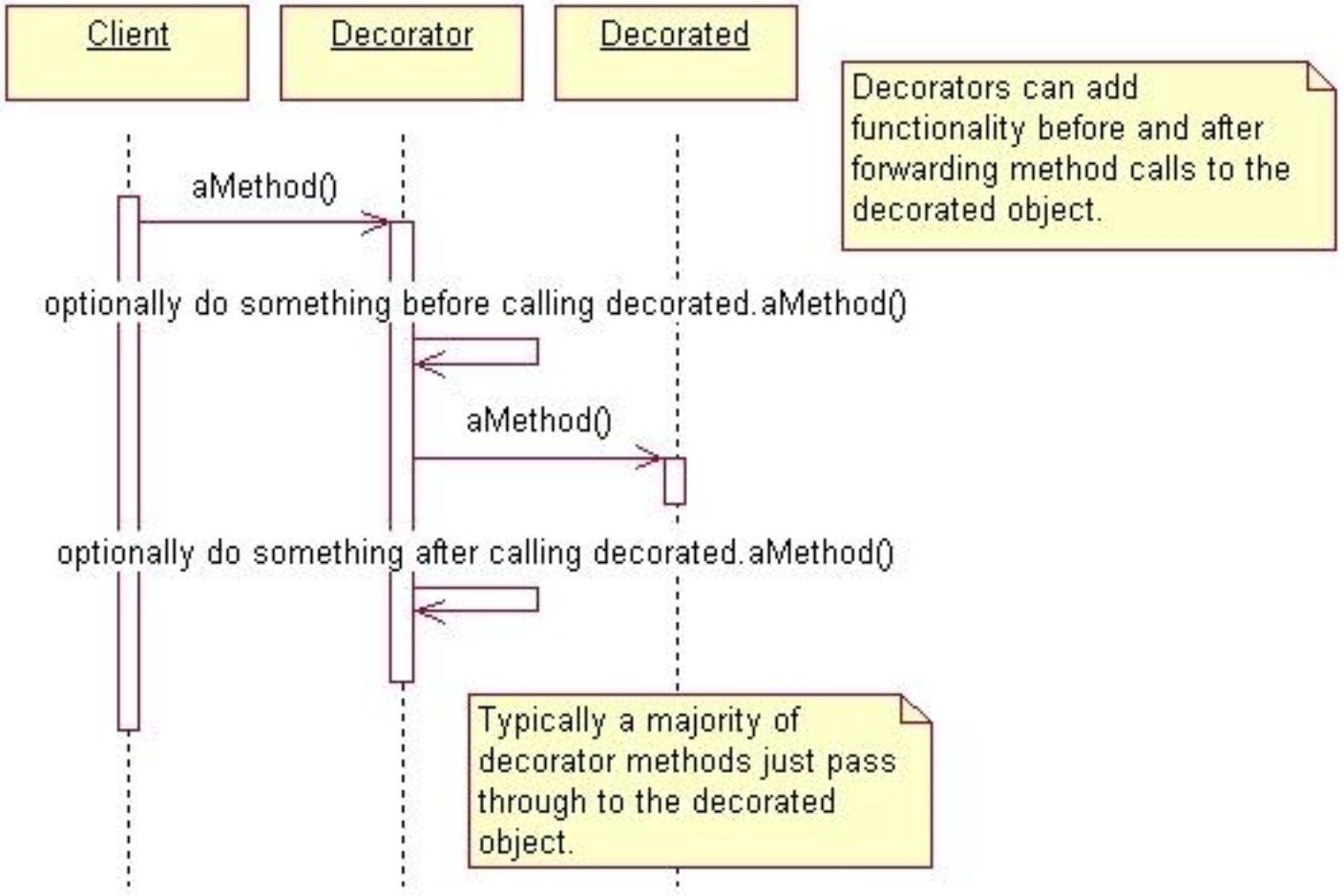
```
interface Phone {
   void dial(String nr);
   void draw(Graphics g);
}

class Mobile implements Phone {
   public void dial(String nr) {
      call(nr);
   }
   public void draw(Graphics g) {
      drawHandy(g);
   }
   ...
}
```

```
class RedDecorator
                  implements Phone {
   private final Phone inner;
   public RedDecorator(Phone p) {
      this.inner = p;
   }

   //wrapper method
   public void dial(String nr) {
      inner.dial(nr);
   }

   public void draw(Graphics g) {
      inner.draw(g);
      drawRedCover();
   }
   ...
}
```

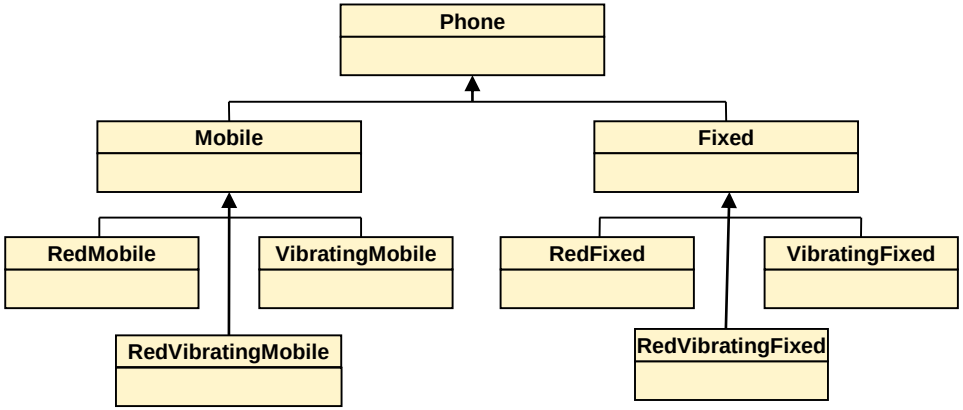
Decorator Pattern Structure

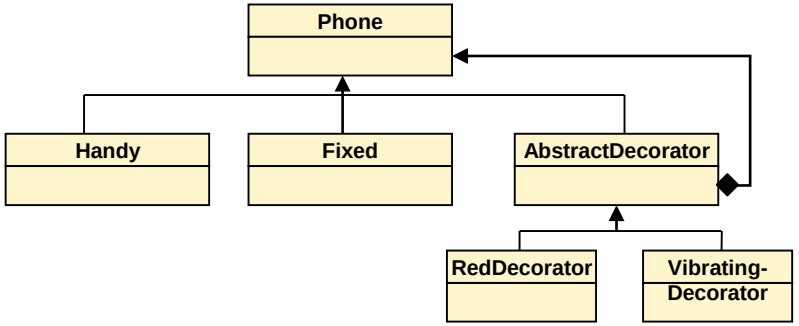


Decorator Pattern Dynamics



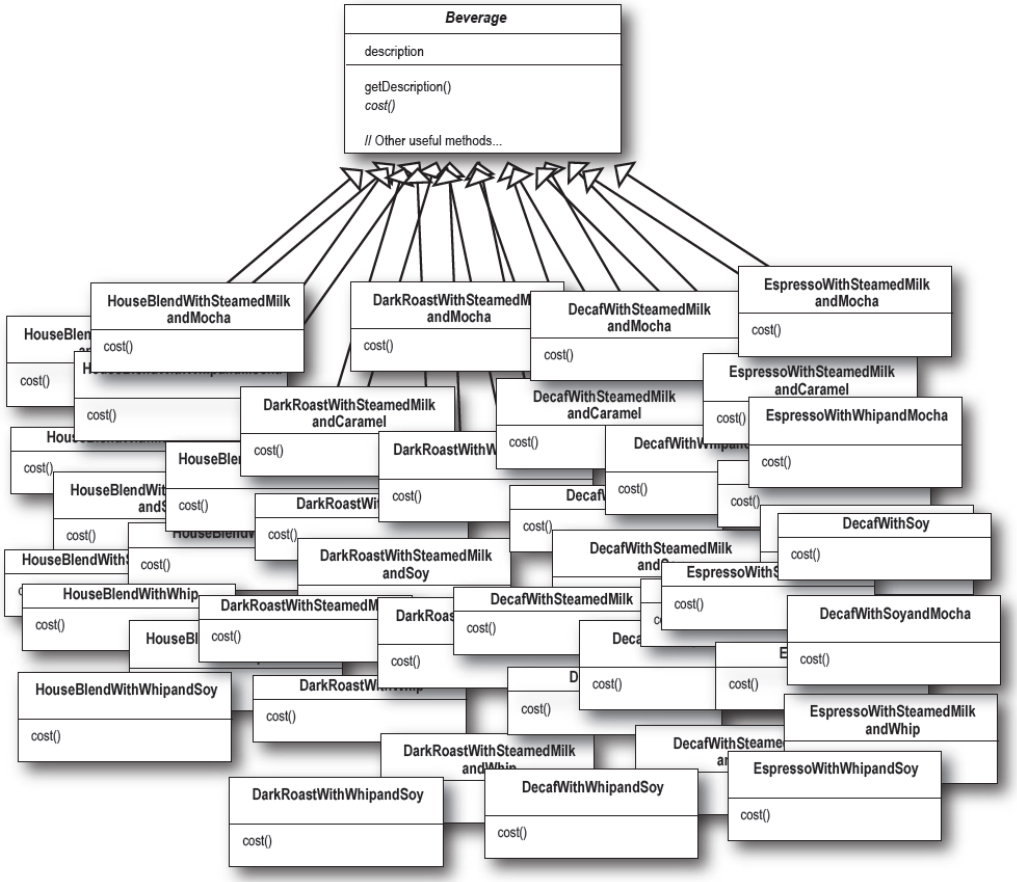
Number of Classes





Number of Properties	Number of Subclasses with inheritance	additional classes with decorator
1	1	2
2	3	3
100	1267650600228229401496703205375	101
n?	2^n-1	n+1

Number of Classes (with Inheritance)



© Head First Design Patterns, p81