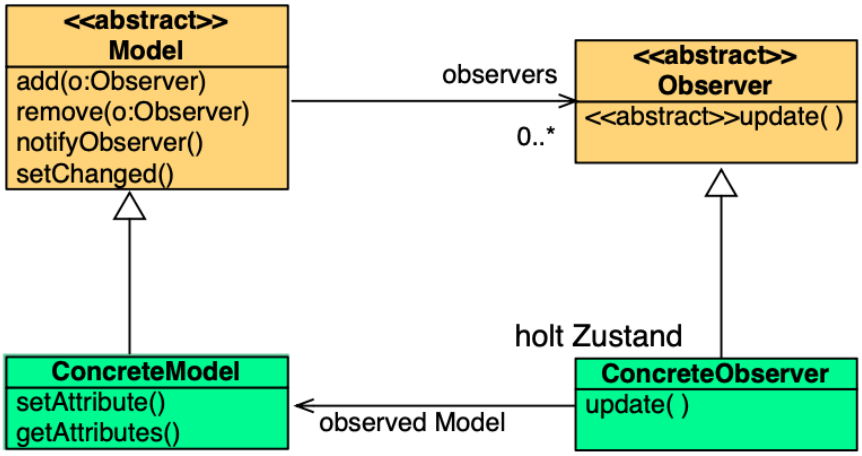
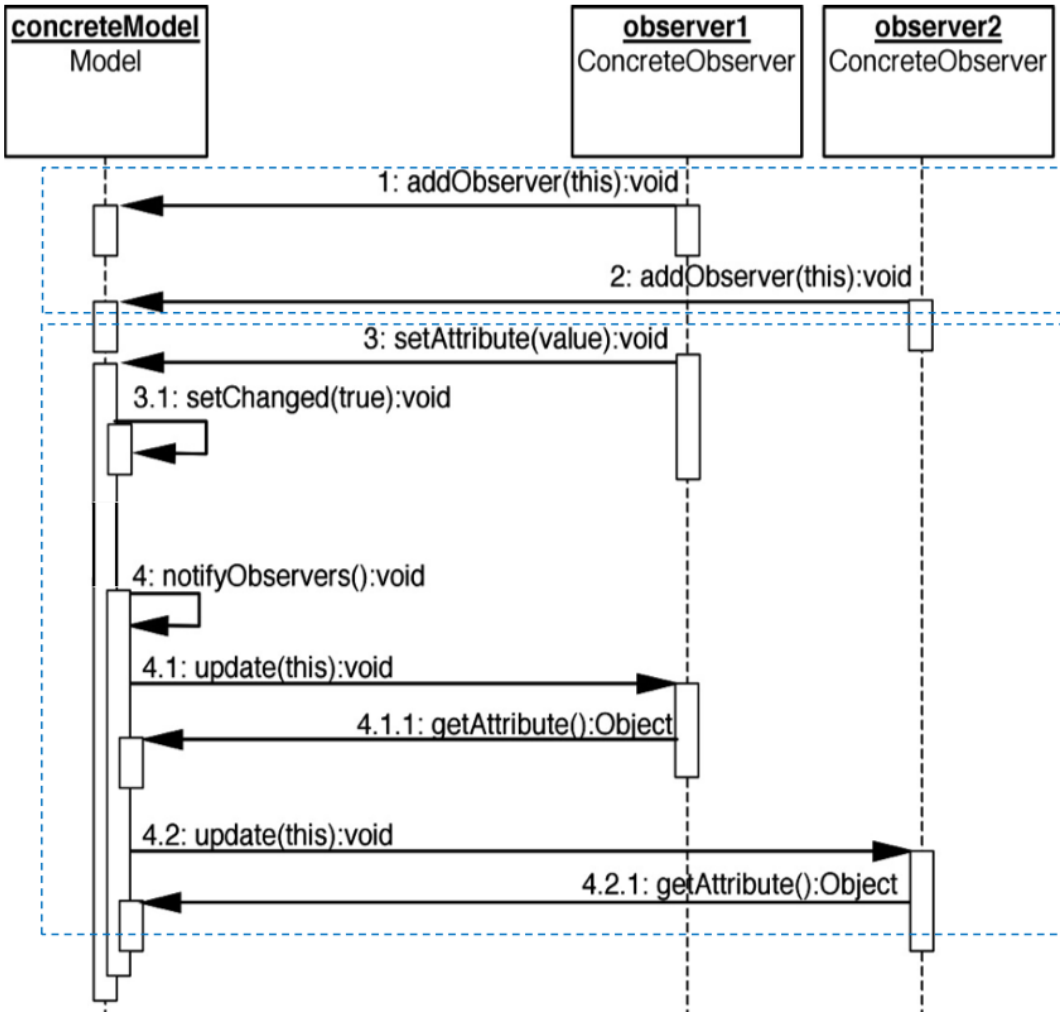
Observer Pattern

a) Im Klassendiagramm mit Kardinalitäten, Beziehungen, Notwendigen Methoden (abstrakt kennzeichnen)

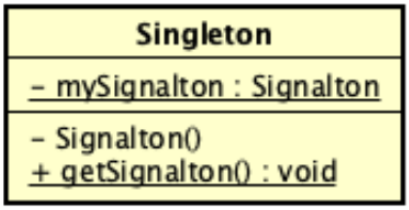
b) Observer1 & Observer2 registrieren sich am ConcreteModell. Observer1 ändert einen Wert. Zeichnen der Abläufe im Sequenzdiagramm.





SINGLETON PATTERN

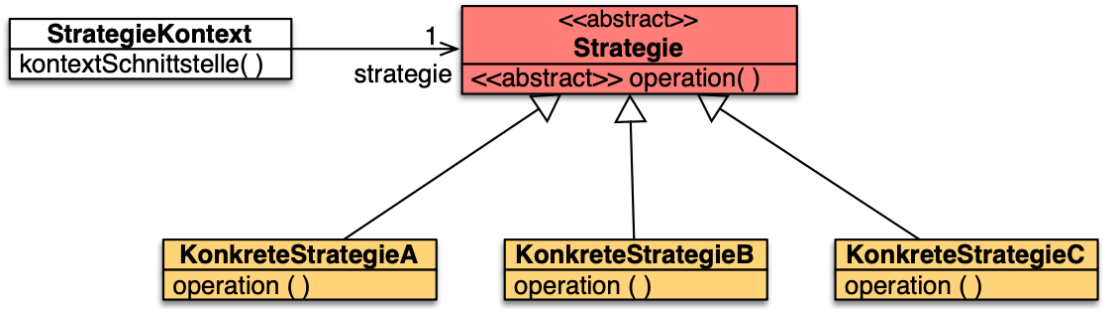
a.)



b.)

```
public class Singleton {
private static Singleton obj = null;
private Singleton() { }
public static Singleton getSingleton (){
    if (obj == null)
        obj = new Singleton();
    return  obj;
    }
}
```

STRATEGIE PATTERN

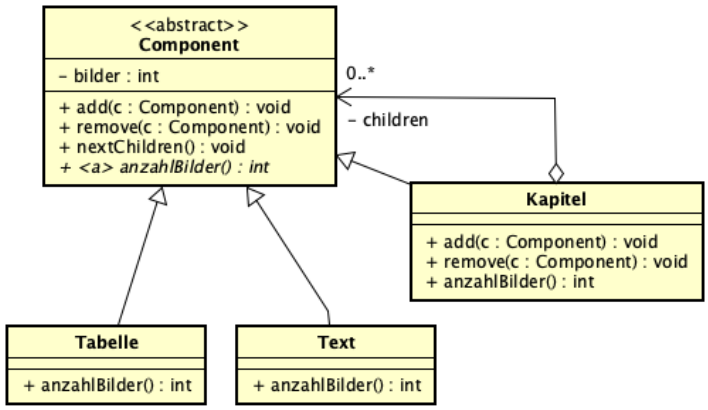


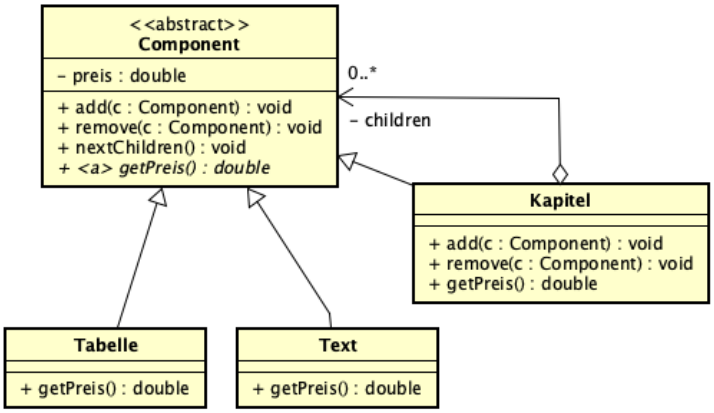

```
public class Anlagemanagment{
    private Strategie startegie;

    public void setStrategie(Strategies){
        this.startegie = s;
    }

    public double berechneMWST (double d) {
        if ( this.startegie == null)
            return 0;
        return startegie.berechneMWST(d);
    }
}
```

```
abstract class Strategie{
    public abstract double berechneMWST
(double);
}

public class S_Ger extends Strategie {
    public double berechneMWST (double d){
        return d*0,19;
    }
}
```

Kompositum Pattern

- Kapitel (z.B. eines Buches) als das Composite und Inhalte als Leaves || anzahlBilder()






```
public class Kapitel extends Component {
    ArrayList<Component> children = new
                          ArrayList<>();
    public Kapitel (int i){
        bilder = i;
    }
    public int anzahlBilder(){
        bilder = 0;
        for(Component k: children)
            bilder += k.bilder;
        return bilder;
    }
    public add(Component c){
        children.add(c);
    }
    public remove(Component c){
        children.remove(c);
    }
}
```

```
abstract class Component{
    private int bilder;

    public Component(int i){
        bilder = i;
    }
    public abstract int anzahlBilder();

    public add(Component c){}
    public remove(Component c){}

}
public class Tabelle extends Component{
    public Tabelle(int i){
        super(i);
    }
    public int anzahlBilder(){
        return this.bilder;
    }
}
```

– eine Hardware-Komponente (z.B. ein PC) besteht aus einem einzelnen Bauteil (mit Namen und Preis) oder aus einem Teilsystem
– ein Teilsystem hat einen Namen und besteht aus einzelnen Bauteilen und kann wiederum Teilsysteme besitzen. Der Preis eines Teilsystems ergibt sich aus den Preisen seiner Komponenten.