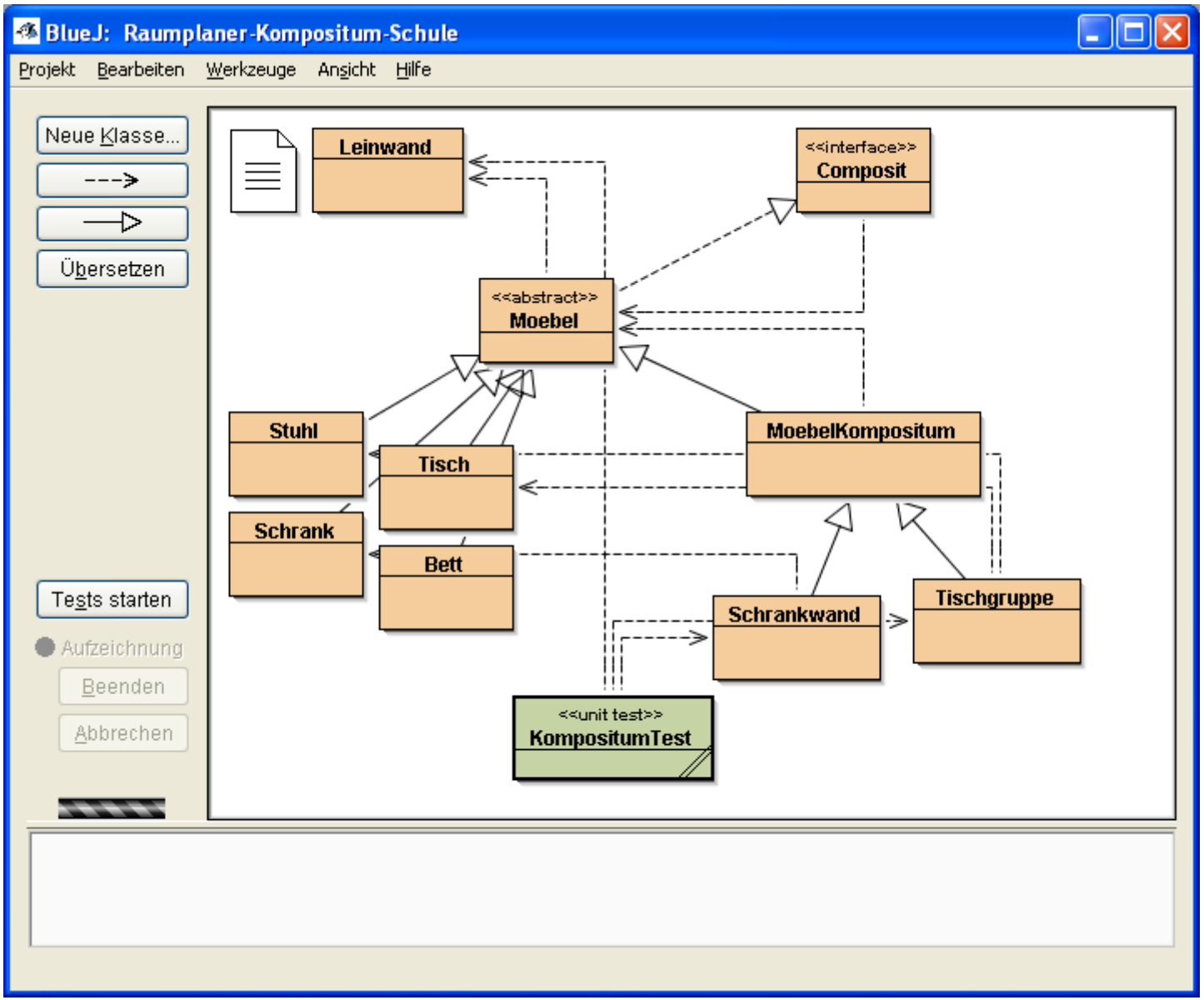
Raumplaner - Kompositum



Raumplaner - Kompositum

```
public interface Composit
{
	/**
	 * fuegt ein Kindobjekt hinzu.
	 *
	 * @param  obj    das Kindobjekt
	 * @return        void
	 */
	public void fuegeHinzu(Moebel moebel);

	...

}
```

Raumplaner - Kompositum

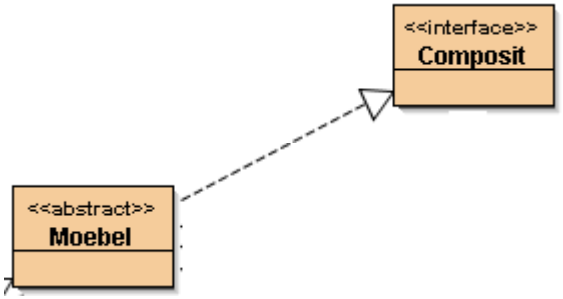
```
public interface Composit
{
        ...

        /**
         * entfernt ein Kindobjekt.
         *
         * @param  obj    das Kindobjekt
         * @return        void
         */
        public void entferne(Moebel moebel);

        ...
}
```

Raumplaner - Kompositum

```
public interface Composit
{
	...

	/**
	 * gibt das Kindobjekt an der Position i zurück
	 *
	 * @param  i Position
	 * @return		das Kindobjekt
	 */
	public Moebel gibKindobjekt(int i);
}
```

Raumplaner - Kompositum

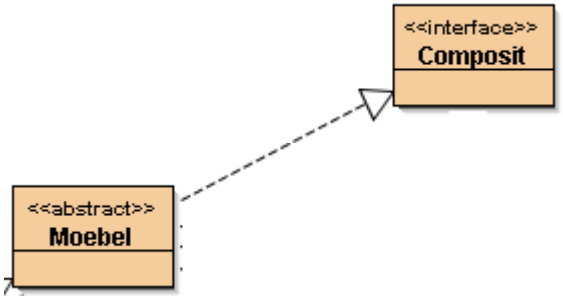
```
public abstract class Moebel
implements Composit
{
    ...

    // Die Kompositum-Methoden

        /**
         * fuegt ein Kindobjekt hinzu.
         *
         * @param  obj    das Kindobjekt
         * @return        void
         */
        public void fuegeHinzu(Moebel moebel){

        }

        ...
}
```



die Methode wird leer implementiert

Raumplaner - Kompositum

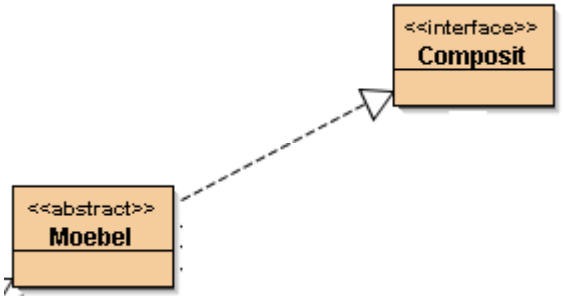
```
public abstract class Moebel
implements Composit
{
    ...

    // Die Kompositum-Methoden


        ...
        /**
         * entfernt ein Kindobjekt.
         *
         * @param  obj    das Kindobjekt
         * @return        void
         */
        public void entferne(Moebel moebel){

        }
        ...
}
```



die Methode wird ebenso leer implementiert

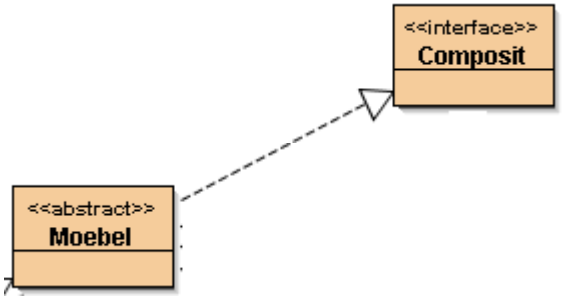
Raumplaner - Kompositum

```
public abstract class Moebel
implements Composit
{
    ...

    // Die Kompositum-Methoden


        ...
        /**
         * gibt das Kindobjekt an der Position i zurück
         *
         * @param  i Position
         * @return          das Kindobjekt
         */
        public Moebel gibKindobjekt(int i){
            return null;
        }
}
```



und auch diese Methode wird leer implementiert, muss allerdings etwas zurückgeben

Raumplaner - Kompositum



Was soll das?

Es ist sichergestellt, dass jede Instanz der Klasse Moebel die Methoden implementiert.

Zumindest bei Blättern
- also nicht zusammengesetzten Moebel-Objekten -
sind dies auch sinnvolle Implementationen.

Raumplaner - Kompositum


```
public class MoebelKompositum extends Moebel
{
    private ArrayList<Moebel> kompositum = new ArrayList<Moebel>();
```

erbt von Moebel,

verwaltet die Datenstruktur,

muss nun die geforderten Methoden sinnvoll implementieren!

Raumplaner - Kompositum


```
/**
 * fuegt ein Kindobjekt hinzu.
 *
 * @param  obj    das Kindobjekt
 * @return        void
 */
public void fuegeHinzu(Moebel moebel) {
    verberge();
    kompositum.add(moebel);
    zeige();
}
```

die Änderungen sollen auch sichtbar sein

Raumplaner - Kompositum


```
/**
 * entfernt ein Kindobjekt.
 *
 * @param  obj    das Kindobjekt
 * @return        void
 */
public void entferne(Moebel moebel){
    verberge();
    kompositum.remove(moebel);
    zeige();
}
```

die Änderungen sollen auch sichtbar sein

Raumplaner - Kompositum


```
/**
 * gibt das Kindobjekt an der Position i
zurück
 *
 * @param  i Position
 * @return          das Kindobjekt
 */
public Moebel gibKindobjekt(int i){
    return kompositum.get(i);
}
```

hier muss nun wirklich das Moebel - Objekt zurück gegeben werden!

override

```
/**
 * fuegt ein Kindobjekt hinzu.
 *
 * @param  obj    das Kindobjekt
 * @return        void
 */
public void fuegeHinzu(Moebel moebel){

}
```

Die Methode fuegeHinzu(...) aus Moebel ...

wird überschrieben durch die Methode fuegeHinzu(...) aus MoebelKompositum

```
/**
 * fuegt ein Kindobjekt hinzu.
 *
 * @param  obj    das Kindobjekt
 * @return        void
 */
public void fuegeHinzu(Moebel moebel) {
    verberge();
    kompositum.add(moebel);
    zeige();
}
```

override

Beim Überschreiben (override)
von Methoden oder Attributen
verdeckt (und damit ersetzt)
die Methode der erbenden Klasse
für ihre Instanzen
die Methode der vererbenden Klasse

Raumplaner - Kompositum

und ein
Nachtrag noch:

Raumplaner - Kompositum

Da MoebelKompositum von Moebel erbt,

muss die Klasse die Methode gibAktuelleFigur() implementieren!

```
/**
 * Berechnet das zu zeichnende Shape anhand der gegebenen Daten
 */
protected Shape gibAktuelleFigur() {
    // einen GeneralPath definieren
    GeneralPath figur = new GeneralPath();
    for (Iterator<Moebel> it= kompositum.iterator(); it.hasNext();)
    figur.append(it.next().gibAktuelleFigur(), false);
    return transformiere(figur);
}
```