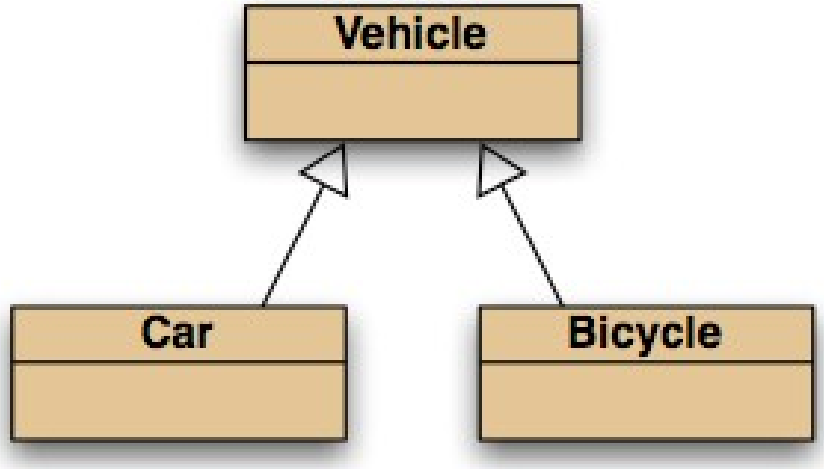
Vererbung und Subtyping

Subklassen und Subtypen

- Klassen definieren immer auch (Variablen-)Typen
- Unterklassen **(Subklassen)** definieren in diesem Sinne Untertypen oder **Subtypen**.
- Objekte von Unterklassen können auch anstelle von Objekten der Superklasse verwendet werden. (Diesen Vorgang nennt man **Substitution**).

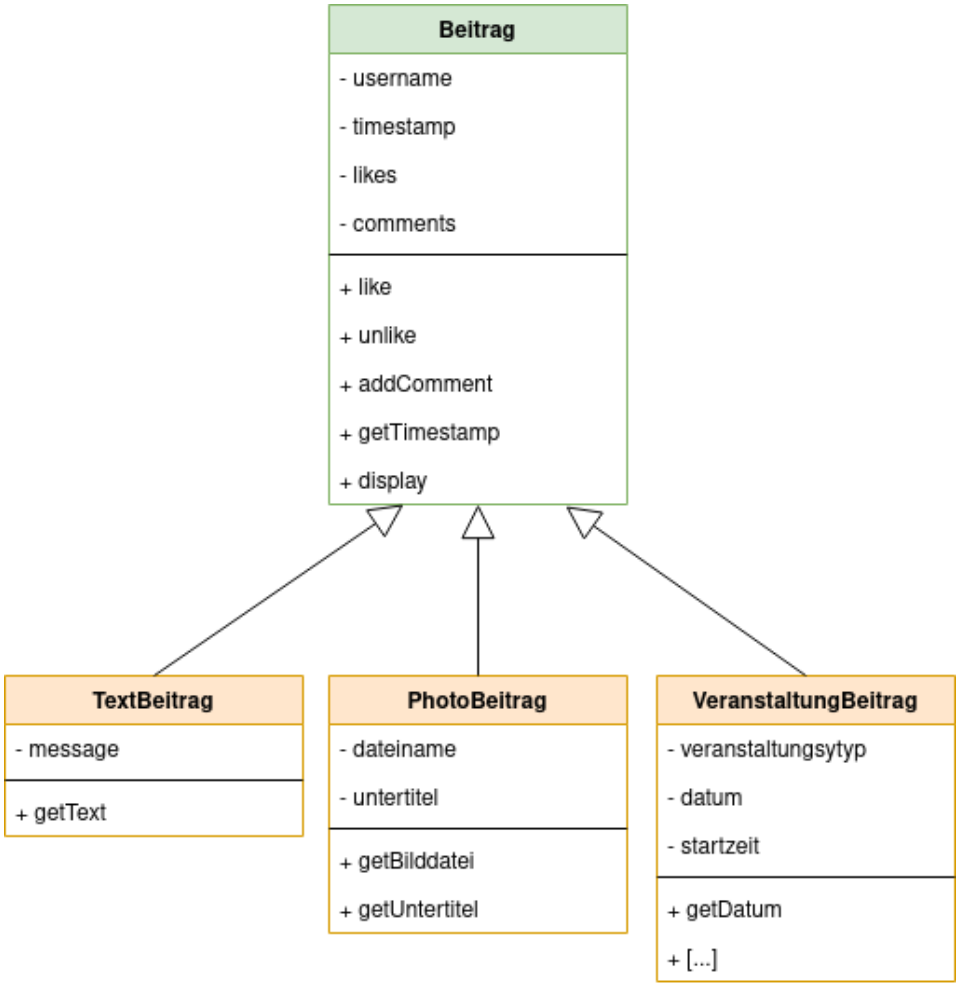
Subtypen und Wertzuweisungen

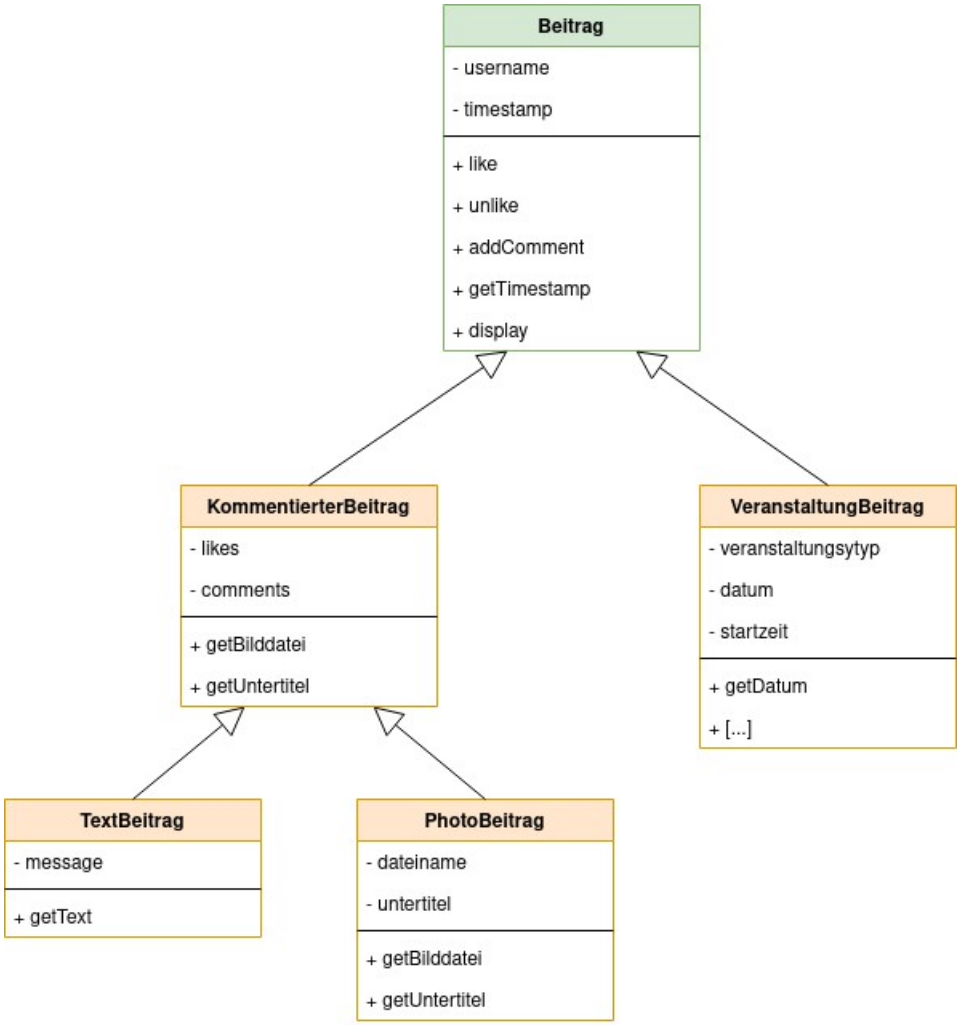
Objekte einer Subklasse können auch Variablen vom Typ der Superklasse zugeordnet werden.




```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Mehr Beitragstypen/Mehr Unterklassen





Im Code...

```
public class NewsFeed
{
    private ArrayList<Beitrag> posts;

    /**
     * Leerer NewsFeed
     */
    public NewsFeed()
    {
        posts = new ArrayList<>();
    }

    /**
     * Neuen Beitrag hinzufuegen
     */
    public void addPost(Post post)
    {
        posts.add(post);
    }
    ...
}
```

~~Dopplungen~~

Im Code...

```
/**
  * Zeige die Beitraege in einer Liste an
  * Zuerst mal nur auf der Textkonsole.
  */
 public void showFeed()
 {
        // display all posts
        for(Beitrag post : posts) {
            post.display();
            System.out.println();
        }
 }
```

Dopplungen

Subtyping

Aus...

```
public void addTextBeitrag(TextBeitrag text)

public void addPhotoBeitrag(PhotoBeitrag photo)

[...]
```

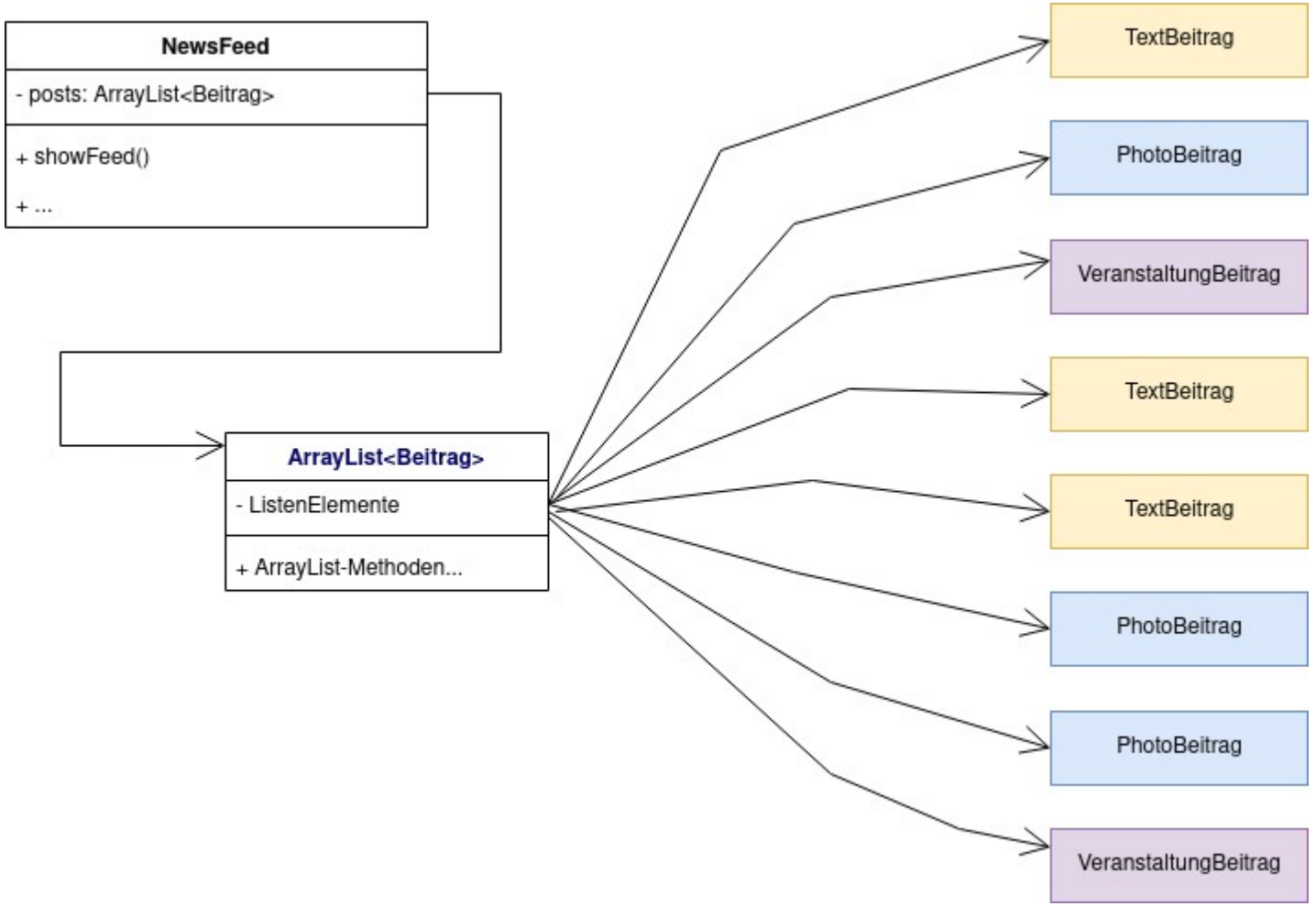
... wird:

```
public void addPost(Beitrag post)
```

Methodenaufruf mit Subtyping

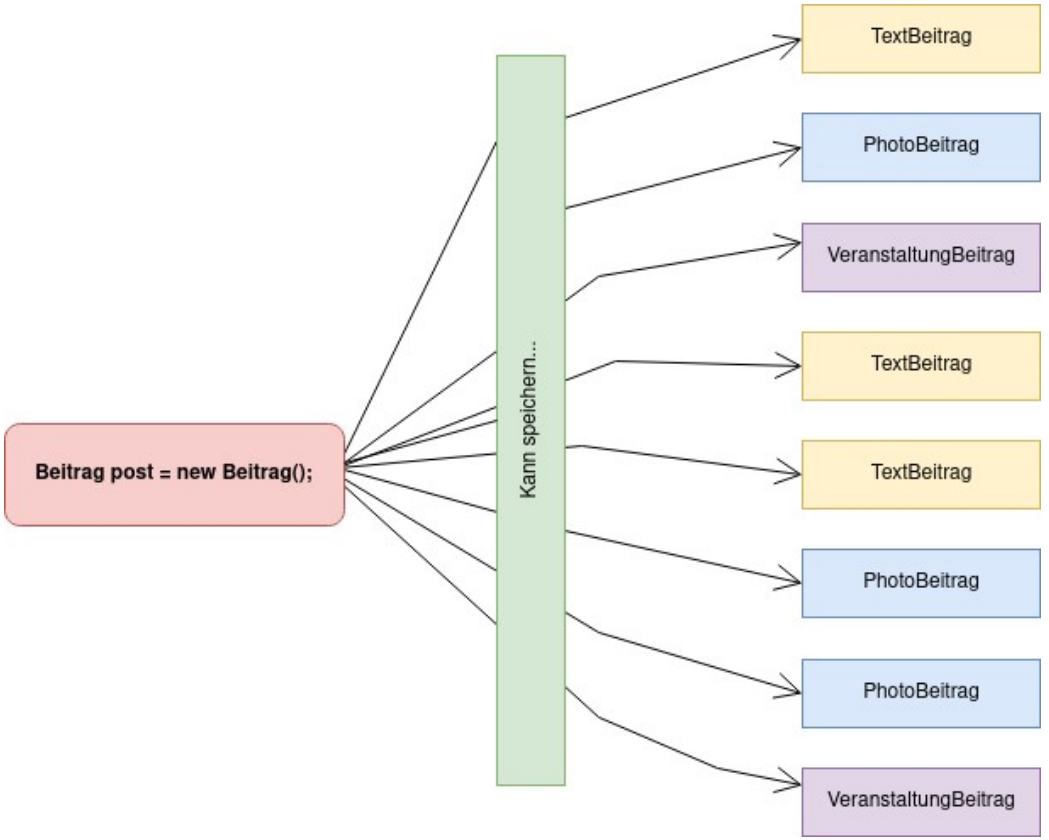
```
TextBeitrag myMsg = new TextBeitrag(… Parameter …);
feed.addPost(myMsg);

PhotoBeitrag myPhoto = new PhotoBeitrag(…);
TextBeitrag myMsg2 = new TextBeitrag(…);

feed.addPost(myPhoto);
feed.addPost(myMsg2);
```

Objektdiagramm



Polymorphie I - Variablen

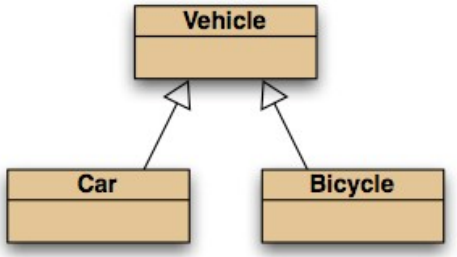
Objektvariablen in Java sind **polymorph**: Sie können Objekte verschiedender Typen speichern – Objekte des **deklarierten Typs** und **aller seiner Subtypen**.



Casting

Man kann Objekte des Subtyps in Variablen des Supertyps speichern – **aber nicht andersherum!**

```
Vehicle v;
Car c = new Car();
v = c;   // correct
c = v;   // compile-time error!
```



Durch **Casting** kann man eine solche Zuweisung ermöglichen – wenn **`v`** tatsächlich vom Typ **`Car`** ist:

```
c = (Car) v;
```

Casting

- Casting geht durch Voranstellen des Zieltyps in runden Klammern.
- Das Objekt selbst wird durch den Vorgang nicht verändert.
- Zur Laufzeit wird geprüft, ob das Objekt tatsächlich vom gefoderten Typ ist. Wenn nicht: `ClassCastException.`
- Sparsam einsetzen!