

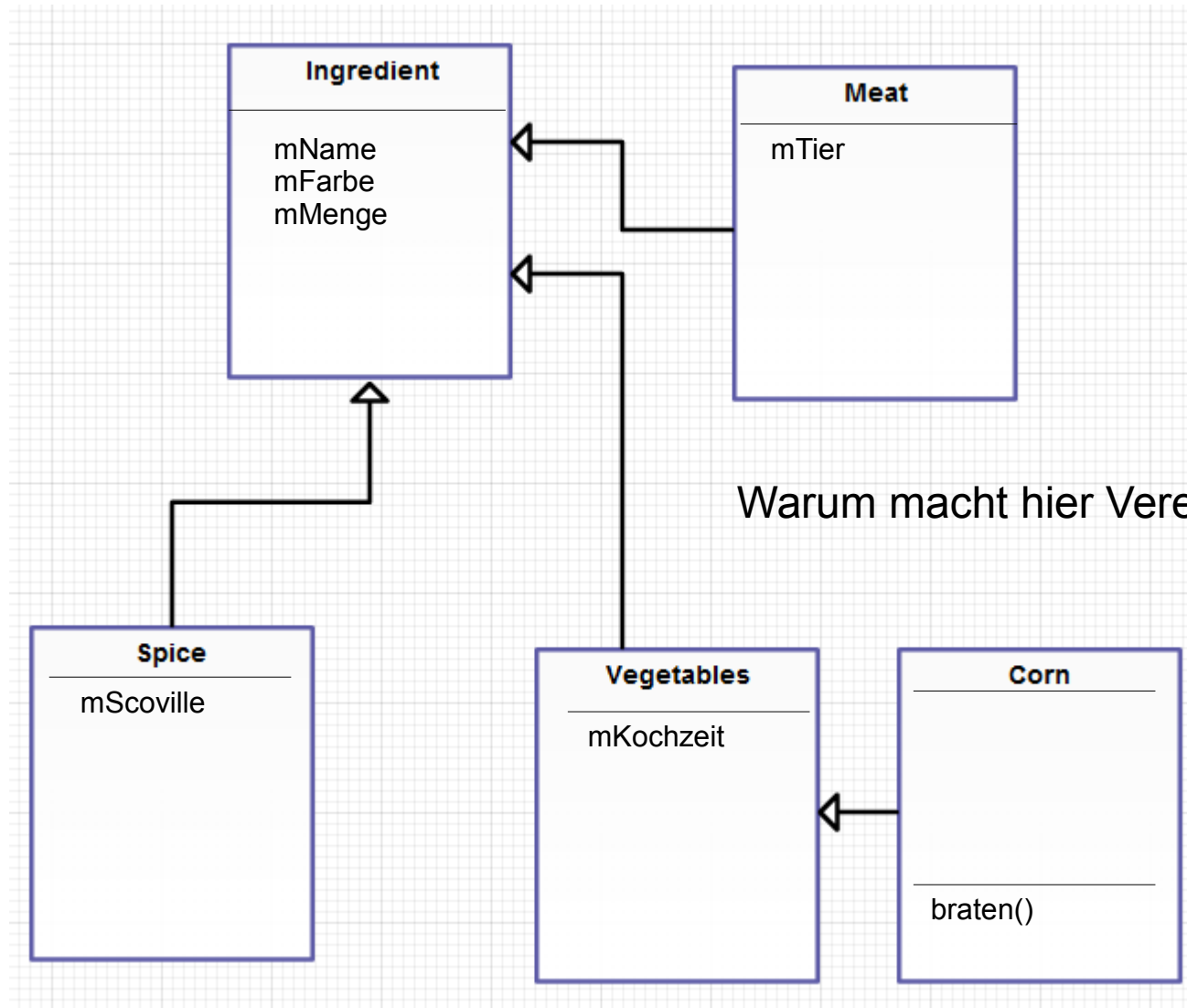
Vererbung

- Ziele
 - Konzept
 - Spezialisierung & Generalisierung von Klassen
 - Methoden überschreiben
 - Darstellung in UML Klassendiagramm
 - Praktische Anwendung
 - Begriff: Polymorphie

Vererbung / Spezialisierung

- Eine Klasse kann von einer anderen Klasse erben
 - Attribute und Methoden der Mutter werden vererbt und sind im Kind nutzbar
- geeignet, wenn unterschiedliche Rollen viele Gemeinsamkeiten haben, aber in einigen wenigen Dingen (Zustände oder Methoden) sich unterscheiden

Generalization / Specialization



Warum macht hier Vererbung Sinn?

Vererbung

```
public class Mutter
{
    public int mGeld=100;
}
```

```
public class Kind extends Mutter
{
    System.out.print(mGeld);
}
```

Das Kind nutzt die Eigenschaften der Mutter und kann diese erweitern

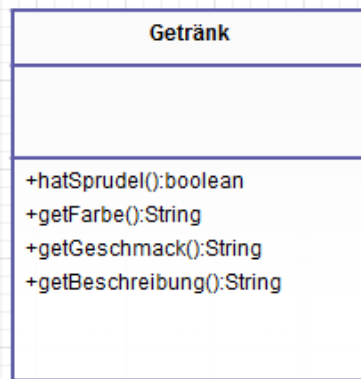
```
Public class Kind extends Mutter
{
    public int mEigenesGeld=mGeld+20;
}
```

Methoden überschreiben

- Kinder können die Methoden der Eltern überschreiben und damit ersetzen
 - Wird genutzt für Spezialanpassungen

Beispiel Methoden überschreiben

```
public class Getraenk {
```



```
    public String getFarbe()
    {
        return "schwarz";
    }
```

```
    public boolean hatSprudel()
    {
        return true;
    }
```

```
    public String getGeschmack()
    {
        return "süß";
    }
```

```
    public String getBeschreibung()
    {
        return "Beschreibung ist: Farbe="+getFarbe()+" ,
        Geschmack= "+getGeschmack()+" mit
        Sprudel="+hatSprudel();
    }
```

```
}
```

Klasse Wasser und Fanta

- Wasser

```
public class Wasser extends Getraenk {  
    public String getFarbe()  
    {  
        return "farblos";  
    }  
    public boolean hatSprudel()  
    {  
        return false;  
    }  
    public String getGeschmack()  
    {  
        return "nichts";  
    }  
}
```

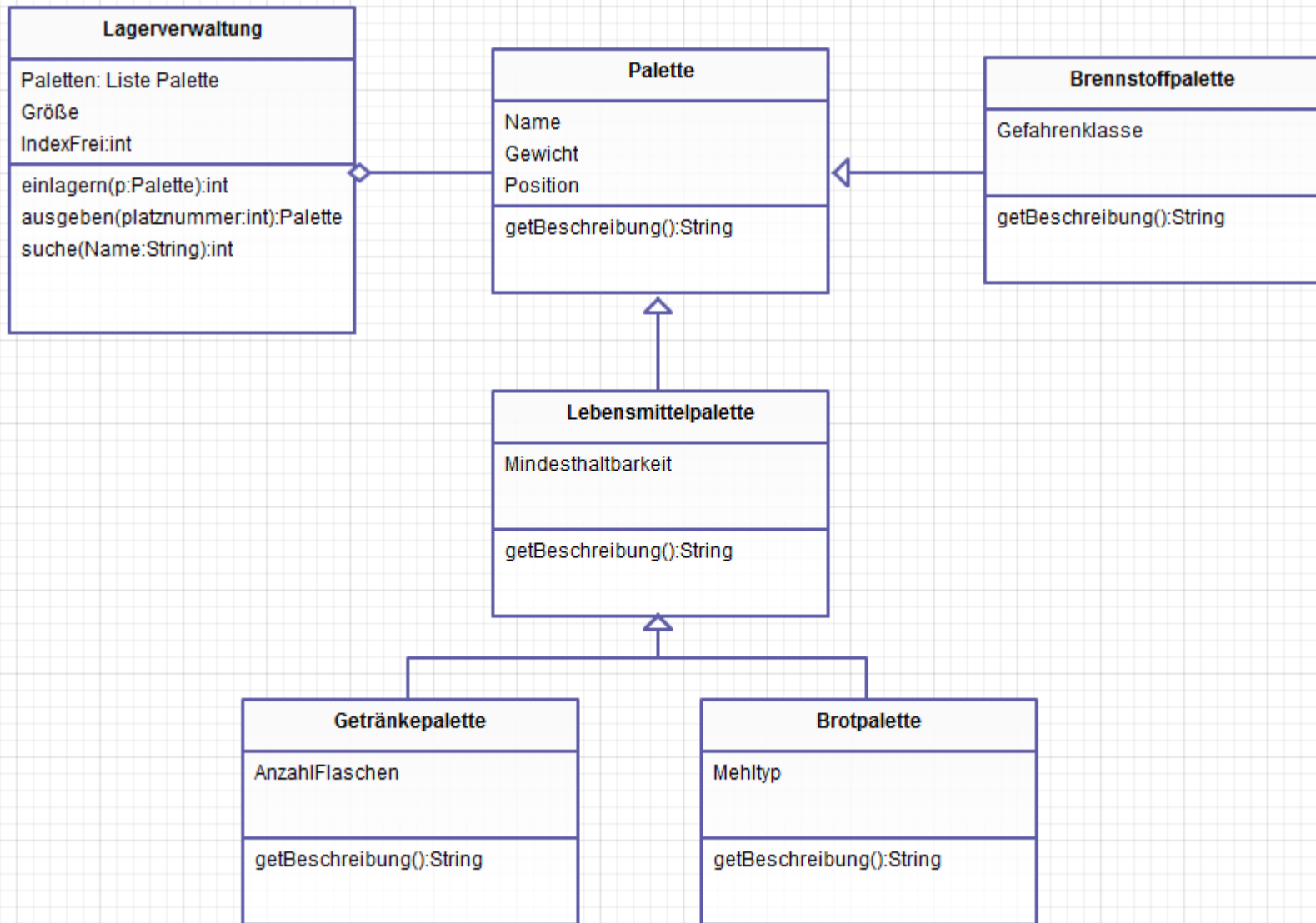
- Fanta

```
public class Fanta extends Getraenk  
{  
    public String getFarbe()  
    {  
        return "gelb";  
    }  
}
```

- main

```
Getraenk getraenk = new Getraenk();  
System.out.println(getraenk.getBeschreibung());  
  
Wasser wasser=new Wasser();  
System.out.println(wasser.getBeschreibung());  
  
Fanta fanta=new Fanta();  
System.out.println(fanta.getBeschreibung());
```

Task Lagerverwaltung



Task Lagerverwaltung

- Implementieren Sie die Klassen
 - Lebensmittelpalette
 - Brotpalette
 - Getränkepalette
 - Brennstoffpalette
 - in jeder Klasse die Methode getBeschreibung, die eine möglichst ausführliche Beschreibung ausgeben soll
- Erstellen Sie von jeder Klasse 2 Instanzen und befüllen Sie das Lager

Vererbung & Konstruktor

- das Kind kann eigene Konstruktoren verwenden, muss allerdings in diesen einen der Elternkonstruktoren aufrufen
 - keyword `super()`
 - wird dann benötigt, wenn Elternklasse einen speziellen Konstruktor erzwingt

Polymorphie

- Erst zur Laufzeit wird bestimmt, welche Methode aufgerufen wird

```
public class Haus {  
    String mName;  
    public Haus()  
    {mName="unbekannt";}  
    public float getHöhe()  
    {return 0;}  
    public String getName()  
    {return mName;}  
}
```

```
public class Leuchtturm  
    extends Haus {  
    public Leuchtturm()  
    {mName="Leuchtturm";}  
    }  
    public float getHöhe()  
    {return 12;}  
    }
```

```
public class Kiosk extends  
    Haus {  
    public Kiosk()  
    {mName="Kiosk";}  
    }  
    public float getHöhe()  
    {return 2;}  
    }
```

```
public class Stadt {  
    public String beschreibeHaus(Haus aHaus)  
    {  
        return "das Haus "+aHaus.getName()+" hat die Höhe"+aHaus.getHöhe()+" m";  
    }  
    public static void main(String[] args) {  
        Leuchtturm haus1=new Leuchtturm();  
        Kiosk haus2=new Kiosk();  
        Stadt stadt=new Stadt();  
        System.out.println(stadt.beschreibeHaus(haus1));  
        System.out.println(stadt.beschreibeHaus(haus2));  
    }  
}
```