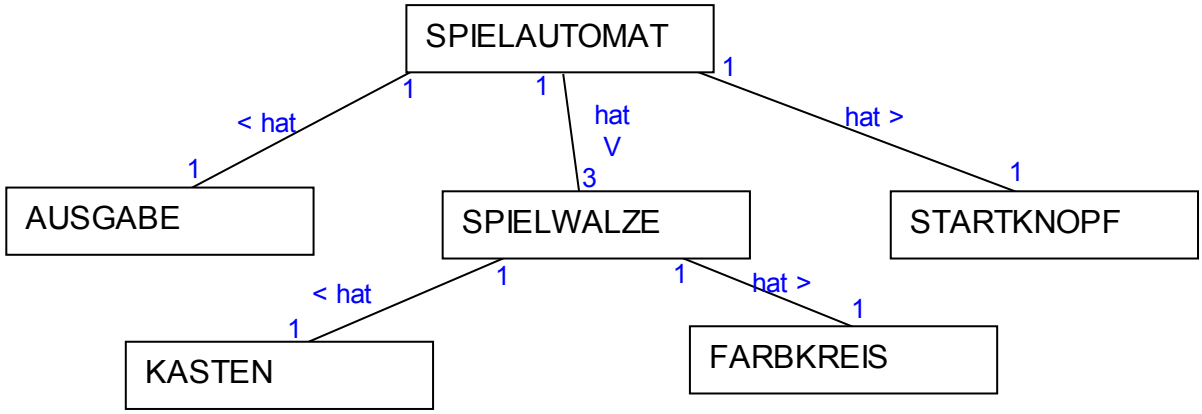
**Grundwissen Informatik 10.Jahrgangsstufe**
**Gymnasium SOB**

## Übersicht:

- Ablaufmodellierung (Algorithmen)
- Objektorientiert Modellierung
- Zusammenführung beider Techniken

Vor der Implementierung steht die Modellierung.
Modellierung: Ausschnitte der Wirklichkeit werden zielgerichtet vereinfacht und strukturiert dargestellt. Man analysiert, wie dieser Ausschnitt so dargestellt werden kann, dass er im Computer umgesetzt werden kann.
Eine geeignete Darstellung z.B. für einen Spielautomaten ist wieder das Klassendiagramm:




Die einzelnen Klassen stellen die Baupläne für die Objekte dar. Das erweiterte Klassendiagramm für den Kasten, das zusätzlich Konstruktor, Zugriffsrecht, Datentypen und Ergebnistypen enthält, könnte so aussehen:

| KASTEN |
| --- |
| private int links<br>private int oben<br>private int breite<br>private int hoehe |
| public KASTEN()<br>public void zeichne()<br>public void setzeLinks(int linksNeu)<br>public int gibUmfang()<br>… |

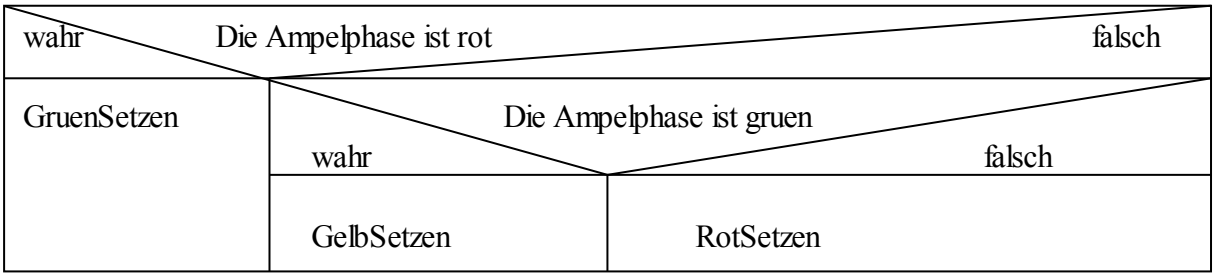
Das erweiterte Klassendiagramm kann in Java unter Verwendung der Oberfläche

BlueJ übertragen werden:

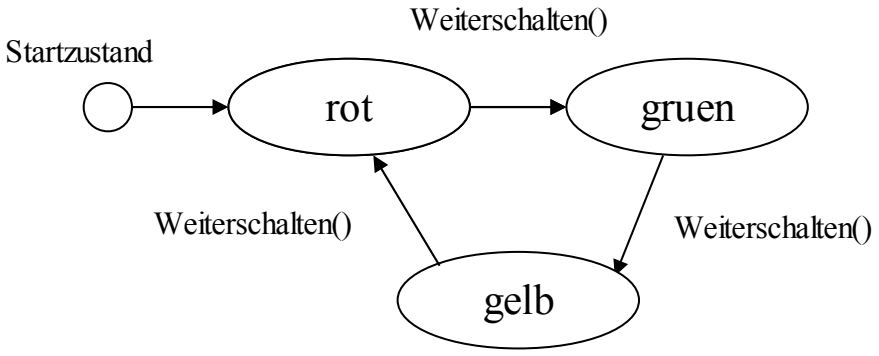
```
public class KASTEN
{     private int links; private int oben;
      public KASTEN()
      { links = 50; oben = 100; }
      public void zeichnen()
      { Zeichenfenster.gibFenster().zeichneRechteck(links,oben,80,50); }
      public void setzeLinks(int linksNeu)
      { links = linksNeu;}
}
```

Die Methoden `gibFenster()` und `zeichneRechteck(…)` stellt dabei die Klasse ZEICHENFENSTER als Schnittstellen zur Verfügung. Als Schnittstelle einer Klasse werden alle öffentlichen Attribute und Methoden verstanden. Die Schnittstelle einer Klasse beschreibt alle Komponenten, die ihre Objekte zur Benutzung durch Objekte anderer Klassen zur Verfügung stellen. An dieser Stelle erfolgt die Unterscheidung in Funktionen mit und ohne Eingangsparameter, sowie mit und ohne Ergebnistyp.

Die Kontrollstrukturen der 7.Jgst. werden in Java umgesetzt, die graphische Darstellung in **Struktogrammen** wie in der 7.Jgst. veranschaulicht.
Beispiel für die Methode `Weiterschalten()`:




Zusätzlich wird die Wertzuweisung als Zustandsübergang eingeführt. Der Zustand eines Objekts ist durch die Zustände seiner Attribute festgelegt.
Beispiel **Zustandsdiagramm** einer vereinfachten Ampelschaltung:

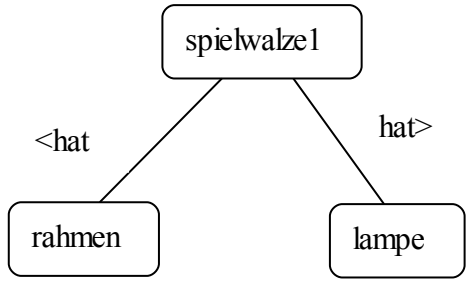



Die Funktion Weiterschalten() verändert den Zustand des Attributs ampelPhase.

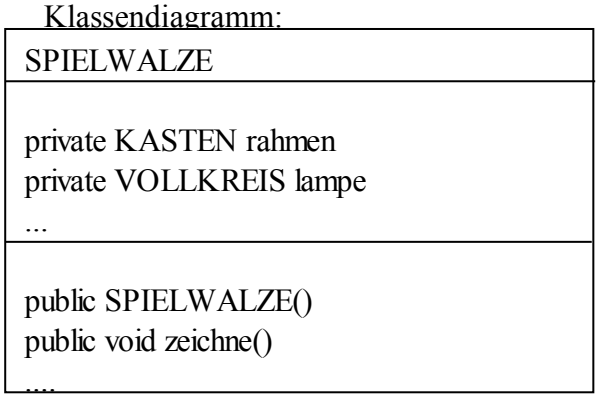
Neben den einfachen Variablen `int, double, boolean,…` mit den Bestandteilen Bezeichner, Wert, Typ, Zuweisung wird zur leichteren Umsetzung der 1: n – Beziehung der Typ Feld eingeführt.
Programmtechnisch wird eine Aggregation dadurch implementiert, dass das enthaltende Objekt lediglich eine Referenz auf das enthaltene Objekt und nicht das tatsächliche Objekt erhält.

Beispiel:
Ein Objekt der neuen Klasse SPIELWALZE soll, wie in folgender Graphik gezeigt, ein Objekt der Klasse KASTEN und ein Objekt der Klasse VOLLKREIS enthalten.

Objektdiagramm:




Klassendiagramm:




In der Klasse SPIELWALZE müssen Attribute vom Objekttyp VOLLKREIS und KASTEN vereinbart werden.
Java – Quelltext:

```
public class SPIELWALZE
{
    private VOLLKREIS lampe;
    private KASTEN rahmen;
     .....
}
```

Im Konstruktor der neuen Klasse müssen den Attributen `rahmen` und `lampe` die Referenzen auf die erzeugten Objekte (als Attributwerte) zugewiesen werden. Dies erfolgt durch den Aufruf der Konstruktoren von KASTEN und VOLLKREIS.

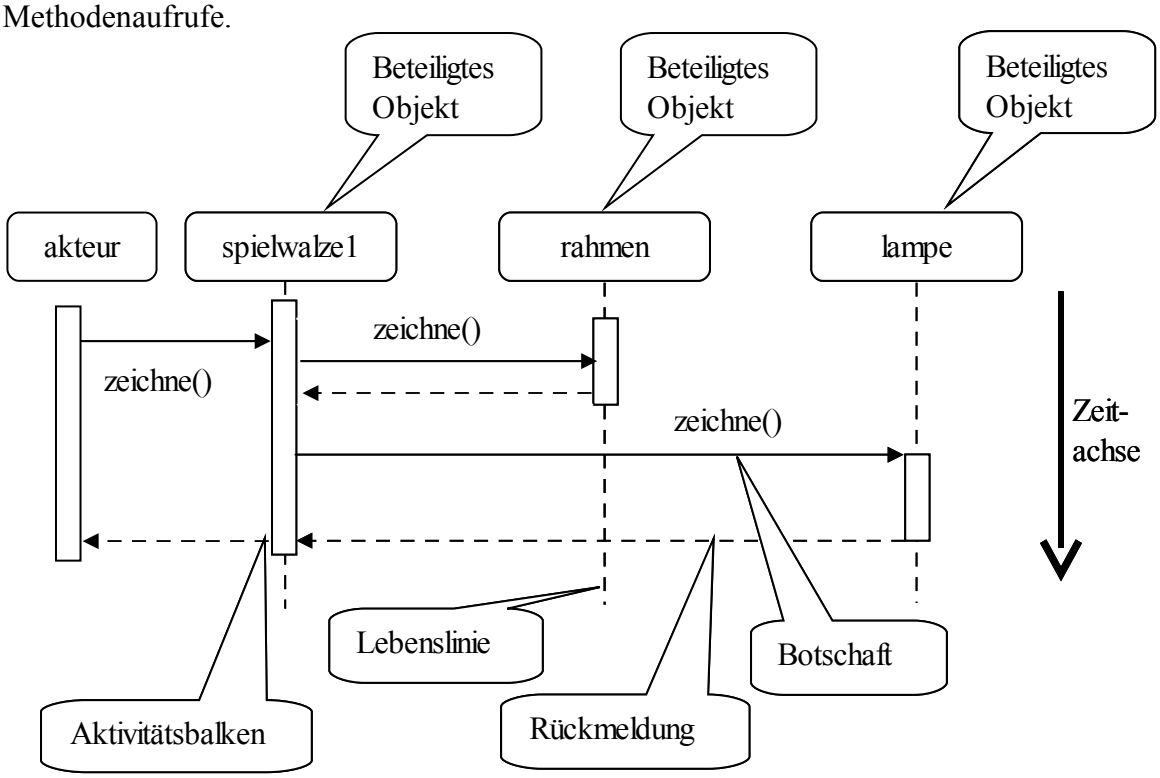
```
public SPIELWALZE()
    {
        lampe = new VOLLKREIS(55, 55, 20, 4);
        rahmen = new KASTEN(30, 30, 150, 50);

    }
```

In den Methoden der Klasse SPIELWALZE können dann die öffentlichen Methoden der referenzierten Objekte aufgerufen werden.

```
public void zeichne()
    {
        lampe.zeichne();
        rahmen.zeichne();
    }
```

Ein **Sequenzdiagramm** (s. S3) veranschaulicht die Kommunikation der Objekte über Methodenaufrufe.




Insbesondere ist bei der Kommunikation zwischen Objekten auf die Datenkapselung über die Zugriffsrechte zu achten.

Sind Klassen **Spezialisierungen** einer bisherigen Klasse, werden sie als **Unterklassen** bezeichnet. In den Unterklassen werden nur die neuen Attribute und Methoden aufgeführt. Methoden und Attribute, die in den Unterklassen neu implementiert werden, müssen dort eingefügt werden. Methoden, die den gleichen Namen tragen, wie in der Oberklasse, aber eine andere Funktion bekommen, werden überschrieben. Alle anderen Methoden und Objekte werden von den Unterklassen geerbt.
Zur Umsetzung der Spezialisierung als Vererbung in Java wird die Beziehung zur Oberklasse mit dem Schlüsselwort `extends` wiedergegeben. Im Konstruktor der Unterklasse wird der Konstruktor der Oberklasse mit `super` aufgerufen.
`Super` ist die Referenz auf den Oberklassenanteil des Objekts. Oberklassenmethoden werden allgemein mit `super.methodenname()` aufgerufen. Da der Konstruktor keinen eigenen Namen hat bleibt `super()`.
Damit die Unterklasse Zugriff auf die Attribute der Oberklasse erhält, muss deren Zugriffsrecht von `private` auf `protected` gelockert werden. Damit wird den Unterklassen die Verwendung der Attribute der Oberklassen erlaubt.
Das Polymorphismuskonzept ermöglicht die Speicherung von Referenzen der Unterklassen in einem Feld von Oberklassenobjekten.

Den Abschluss bildet wieder ein größeres Projekt mit dem die bisherigen Programmier – und Modellierungstechniken eingeübt werden kann.