# 1. Begriffe und Definitionen

Objekt:

Ein Exemplar von Dingen, Personen oder Begriffen der realen Welt oder der Vorstellungswelt.  
Ein Objekt hat einen eindeutigen Namen, bestimmte Eigenschaften und reagiert mit einem definierten Verhalten auf seine Umwelt.

Klasse:

Eine Klasse beschreibt eine bestimmte Art von Objekten. Sie dient als „Bauplan“, in dem festgelegt ist, welche Attribute und welche Methoden die erzeugten Objekte haben.

Attribut:

Die Eigenschaften der Objekte werden durch die Werte ihrer Attribute beschrieben.

Das Attribut selbst ist unveränderlich.

Attributwert:

Der Attributwert ist der momentane Wert eines Attributs und kann verändert werden.

Methode:

Objekte können auf Befehl gewisse Operationen (Methoden) ausführen.

Parameter:

Bei vielen Methoden muss man durch ein oder mehrere Argumente (Parameter) festlegen, mit welchen Eingaben sie ausgeführt werden sollen.

Punktnotation: Objektname.Methodenname();

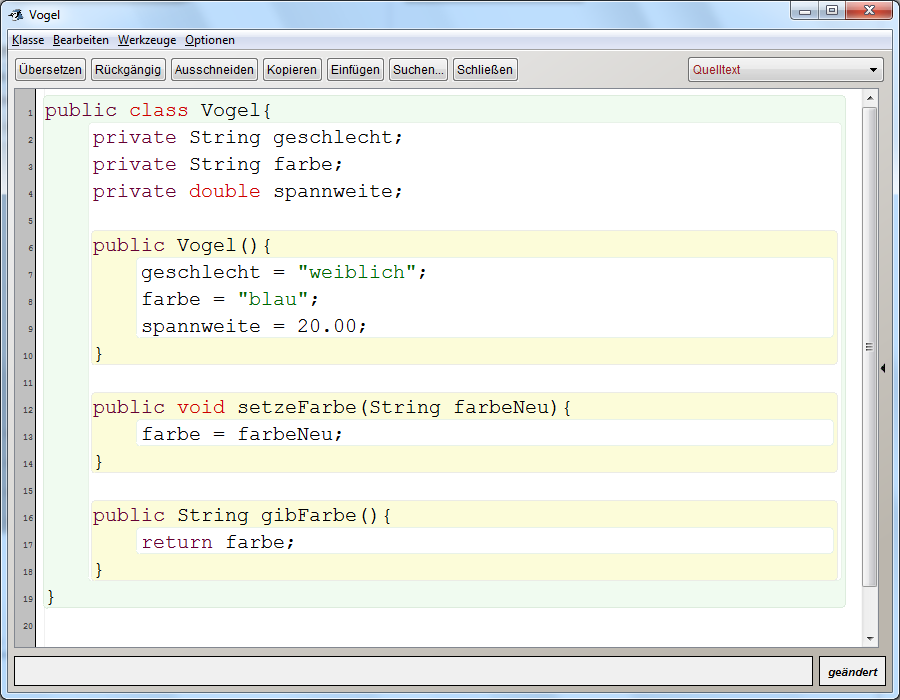
Modellierung:

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.

# 2. Objekte und Klassen

|  |  |
| --- | --- |
| Objektkarten bestehen aus  - eindeutigem Objektnamen  - Attributen  - Attributwerten  (- Methoden) | Klassenkarten bestehen aus  - eindeutigem Klassennamen  - Attributen  - Methoden  - Zugriffsmodifikator, Parameter und Rückgabewerte bei erweiterter Klassenkarte |
| Bsp.: Objektkarte | Bsp.: (erweiterte) Klassenkarte |

Umsetzung von Klassenkarten in Java:



Methodendeklaration (Signatur), (Rückgabewert ist umstritten)

Rückgabewert (Ergebnistyp)

Wertzuweisung

Definition der Attribute

Konstruktor

Parameter

Zugriffsrecht

Klassendefinition

# 3. Datentypen, Rückgabewerte und Zugriffsrechte

a) primitive (elementare) Datentypen

- byte, short, int, long

- float oder double

- char

- boolean

b) Referenztypen (Objekttypen)

- String

- Felder  
 Bsp: Deklaration: private int[] gewinn;

Initialisierung: gewinn = new int[100]; (mit 100 Plätzen)

Wertzuweisung an 2. Stelle: int[1] = 5;

c) Rückgabewerte

- Rückgabewerte können sowohl primitive Datentypen als auch Referenztypen sein

- falls kein Wert zurückgegeben werden soll: Rückgabetyp: void

- hinter dem Schlüsselwort return stehen die Werte bzw. Variablen , die zurückgegeben werden sollen.

d) Zugriffsrechte

- private: Auf das Attribut kann nur innerhalb der definierenden Klasse zugegriffen werden.

- public: Auf das Attribut kann von allen Klassen aus, sowohl von der definierenden als auch von anderen, zugegriffen werden.

- protected: Das Zugriffsrecht protected erlaubt die Verwendung der Attribute der Oberklasse auch in den Unterklassen.

- Zugriffsrechte dienen vor allem der Datenkapselung.

- Datenkapselung ist bei größeren Projekten eine Voraussetzung für effiziente Zusammenarbeit.

Sie sorgt dafür, dass jede Arbeitsgruppe sich nur um die Schnittstellen kümmern muss; es ist nicht

nötig, über Interna der Klassen anderer Arbeitsgruppen Bescheid zu wissen.

# 4. Beschreibung von Abläufen durch Algorithmen

Algorithmus:

Ein Algorithmus ist eine Verarbeitungsvorschrift, die aus einer endlichen Folge von eindeutig ausführbaren Anweisungen besteht. Unter gleichen Voraussetzungen liefert die Ausführung eines Algorithmus stets gleiche Ergebnisse.

a) Sequenz

|  |  |
| --- | --- |
| Struktogramm: | Java-Code: |
|  | Anweisung1;  Anweisung2;  Anweisung3; |

b) bedingte Anweisung (Fallunterscheidung)

|  |  |
| --- | --- |
| Struktogramm: | Java-Code: |
|  | if(Bedingung){  Sequenz1;  }  else{  Sequenz2;  } |

Kann auch ohne else-Zweig sein.

c) vorprüfende Wiederholung (Wiederholung mit Anfangsbedingung)

|  |  |
| --- | --- |
| Struktogramm: | Java-Code: |
|  | while(Bedingung){  Sequenz;  } |

d) nachprüfende Wiederholung (Wiederholung mit Endbedingung)

|  |  |
| --- | --- |
| Struktogramm: | Java-Code: |
|  | do{  Sequenz;  }  while(Bedingung); |

e) Zählschleife

|  |  |
| --- | --- |
| Struktogramm: | Java-Code: |
|  | for(int var=0; var<=Endwert; var++){  Sequenz;  } |

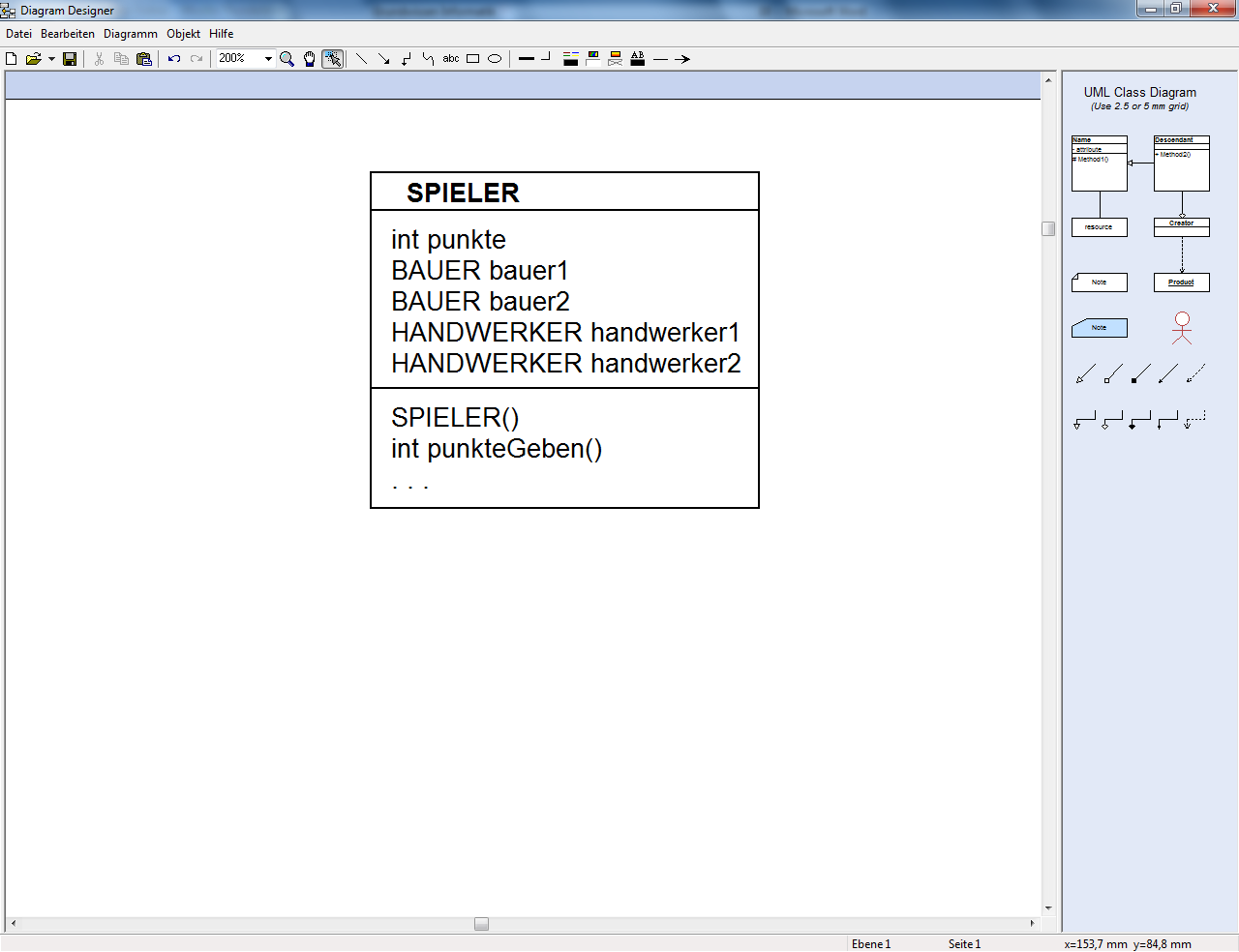
# 

# 5. Objektbeziehung / Referenzattribut

Objektbeziehungen werden durch Referenzattribute realisiert, die eine Referenz auf ein anderes Objekt speichern.

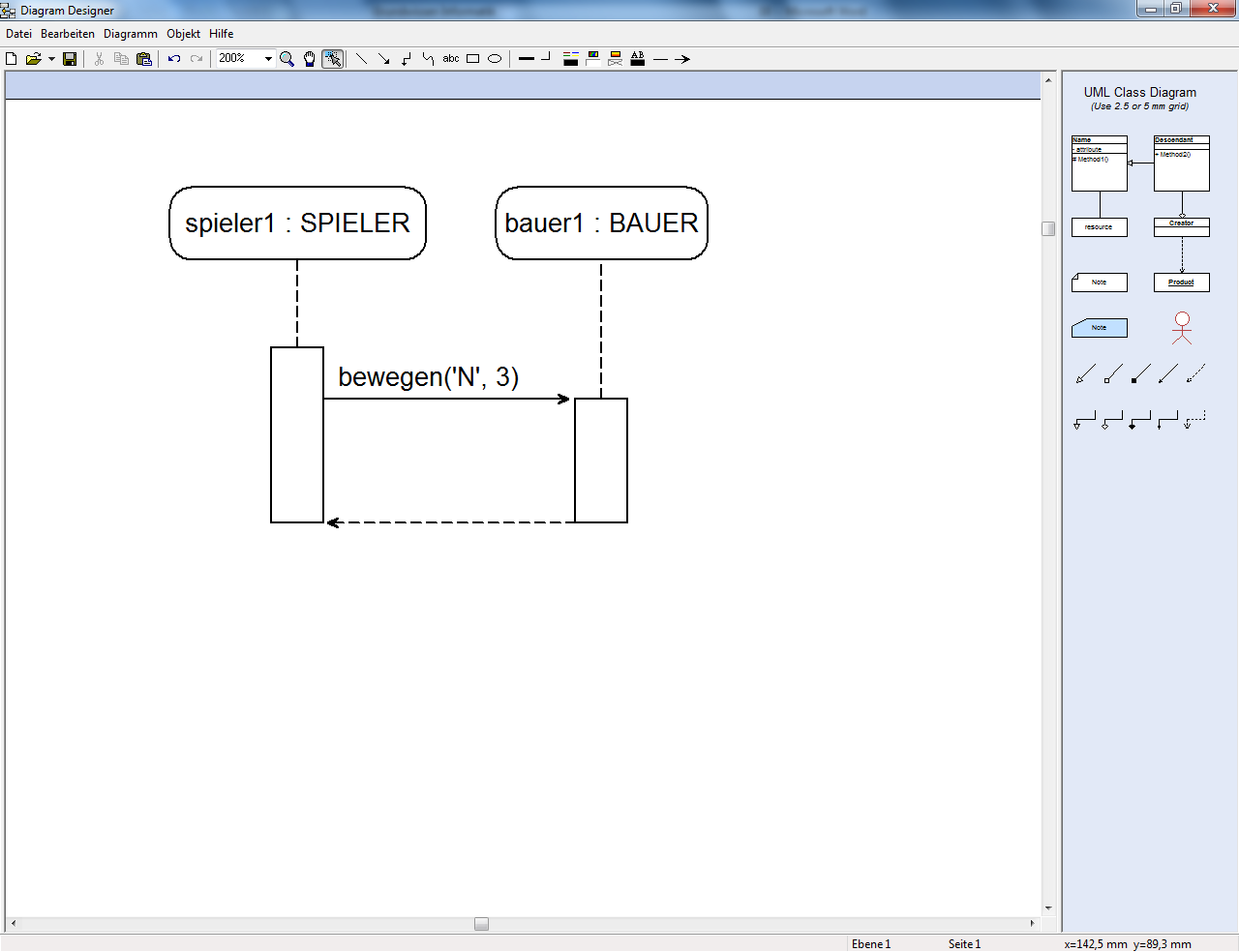
|  |  |
| --- | --- |
| Objektdiagramm mit Beziehungen: | Klassendiagramm mit Beziehungen:  **BAUER**  1 besitzt > 0..2  1 besitzt > 0..2  **SPIELER**  **HANDWERKER** |

Deklaration der Referenzattribute im Klassendiagramm:



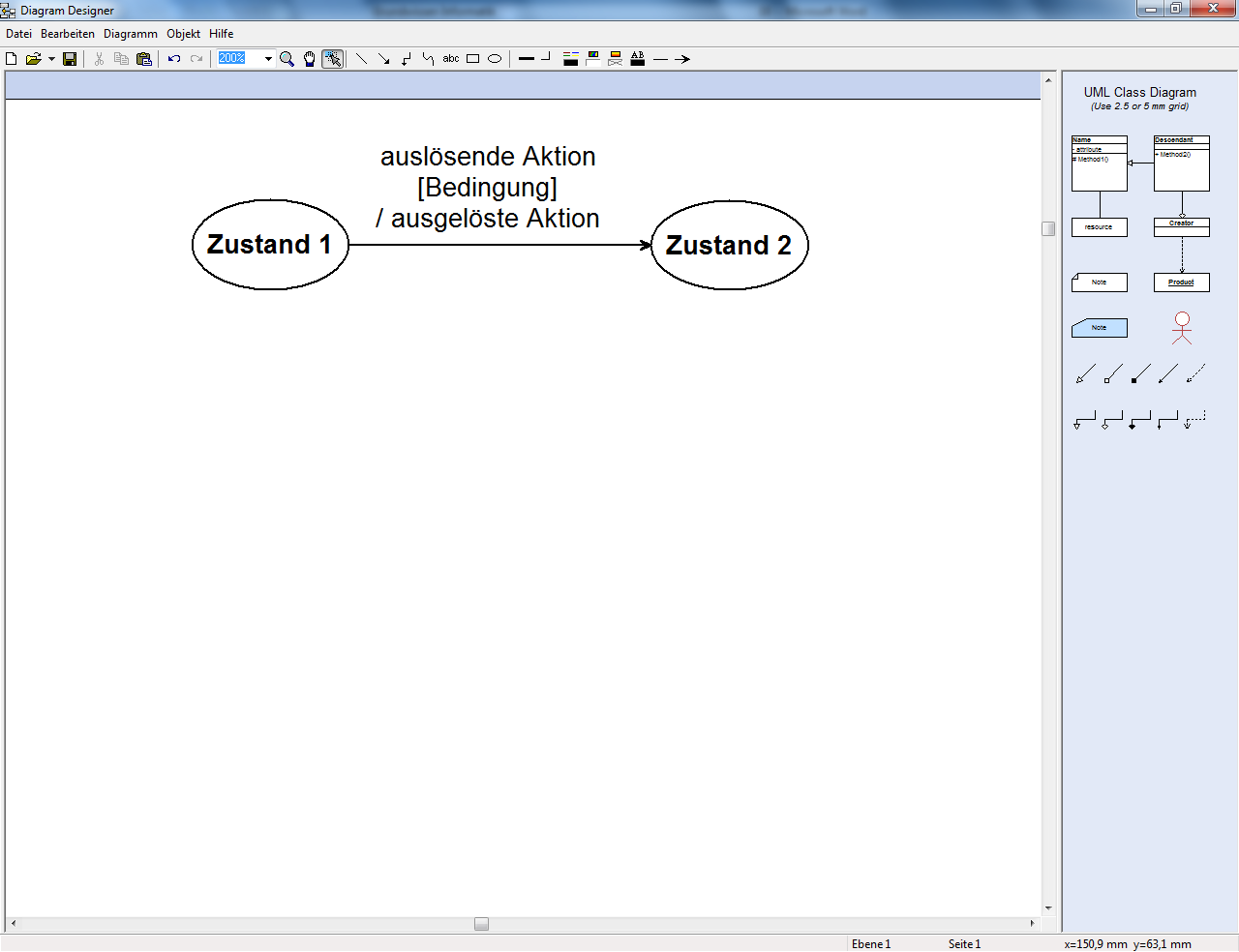
# 6. Objektkommunikation / Sequenzdiagramm

Methoden sind Schnittstellen zur Kommunikation zwischen Objekten. Diese Kommunikation über Methodenaufrufe lässt sich mit Sequenzdiagrammen veranschaulichen.



# 7. Strukturierung von Abläufen mit Zustandsübergangsdiagrammen

Der Zustand eines Objekts wird durch die Gesamtheit aller seiner Attributwerte festgelegt. Durch das Ändern von Attributwerten ändert sich der Zustand, das Objekt vollzieht einen Zustandsübergang. In der Programmierung erfolgt dieser durch die entsprechenden Methodenaufrufe.

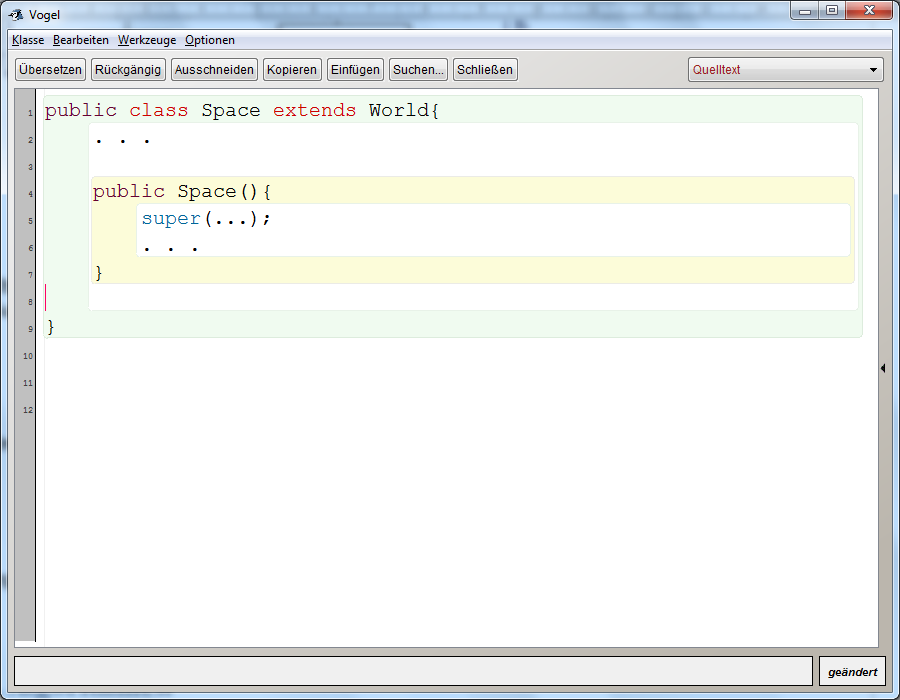


Startzustand

# 8. Beschreibung hierarchischer Klassenstrukturen durch Generalisierung und Spezialisierung

|  |  |
| --- | --- |
|  |  |

Implementierung von Unterklassen in Java durch Aufruf des Konstruktors der Oberklasse mit dem Schlüsselwort super(…):



Ruft Methode der Oberklasse mit gleichem Namen auf.

Polymorphismus:

* Ein Attribut oder Parameter vom Datentyp einer Oberklasse kann auch eine Referenz auf ein Objekt einer Unterklasse aufnehmen. Dies nennt man Polymorphismus.
* In einer Unterklasse kann eine in der Oberklasse vorhandene Methode neu definiert werden.
* Diese Methode der Oberklasse wird in den Unterklassen überschrieben.
* Beim Aufruf wird dann die Methode der Klasse des ausführenden Objekts verwendet. Auch das gehört zum Polymorphismuskonzept.

# 9. Abstrakte Klassen

* Klassen, von denen keine Objekte instanziert werden, können als abstrakt vereinbart werden.
* Abstrakte Klassen beschreiben, welche Methoden von allen Unterklassen aufgerufen werden können.
* Methoden, die in allen Unterklassen überschrieben werden, können als abstrakte Methoden in der Oberklasse ohne Methodenrumpf stehen.
* Die Schnittstelle einer Klasse beschreibt den öffentlich sichtbaren Teil einer Klasse, was sie leistet und wie sie benutzt werden kann, ohne dass ihre Implementierung sichtbar wird.

Beispiel:

public abstract class NIEDERSCHLAG

{

//Attribute

//Konstruktor

public abstract void zeichne();

**//Kein Methodenrumpf !!**

//Weitere Methoden

}

NIEDERSCHLAG {abstract}

protected double x

protected double y

protected double vx

protected double vy

public NIEDERSCHLAG(double xStart, double yStart)

public abstract void zeichne()

public void bewege(double zeit)