Objektorientierte Programmierung

Ludwig Ettner

17. April 2024

Übersicht

1. Einführung
2. Objekte und Klassen
3. Attribute
4. Methoden/Funktionen
5. UML
6. Vererbung
7. Beispiel: Implementierung in Python

Einführung

- In dieser Präsentation werden wir die Grundlagen der objektorientierten Programmierung (OOP) untersuchen.
- OOP ist ein Paradigma zur Strukturierung von Programmcode, das auf dem Konzept von Objekten und Klassen basiert.
- Wir werden zunächst die Konzepte von Objekten und Klassen kennenlernen und dann anhand eines Beispiels Schritt für Schritt durchgehen.

Beispiel: Klasse Charakter

- Beispiel an einem Computerspiel
- Spielcharakter als Objekt
- Besitzt folgende Eigenschaften
  - Lebenspunkte
  - Position in der Weld
  - name des Charakters
- Kann folgende Aktionen ausführen
  - Bewegen
  - Angreifen

Klassen

Definitionen

- **Klasse**: Ein Bauplan für die Erzeugung von Objekten. Definiert Attribute und Methoden.
- **Objekt**: Eine Instanz einer Klasse. Besitzt bestimmte Attribute und kann Methoden ausführen.

Python-Implementierung:

```
1 class Charakter:
2     # Attribute und Funktionen
3
```

Attribute

Definitionen

- **Attribute**: Eigenschaften eines Objekts, die in der Klasse definiert werden.
- Diese Werte die das Objekt besitzt, können verändert werden.

Python-Implementierung:

```
1 class Charakter:
2     def __init__(self, name, lebenspunkte=100, position=(0,
      0)):
3         # muss bei der Instanziierung uebergeben werden
4         self.name = name
5         # standartwert 100
6         self.lebenspunkte = lebenspunkte
7         # standartwert (0, 0)
8         self.position = position
9
```

Methoden/Funktionen

Definitionen

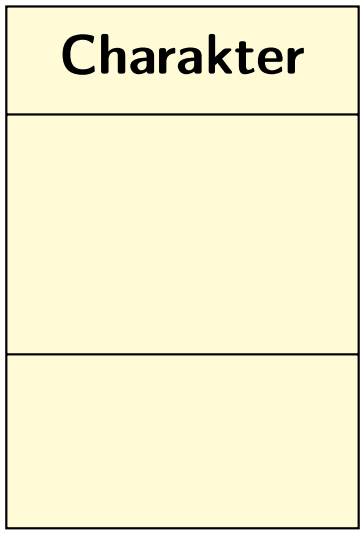
- **Mathoden**: Methoden sind Funktionen, die in einer Klasse definiert werden und auf Objekten dieser Klasse ausgeführt werden können.
- Sie können auf die Attribute des Objekts zugreifen und diese verändern.
- Methoden können auch Parameter entgegennehmen.

Methoden/Funktionen (Fortsetzung)

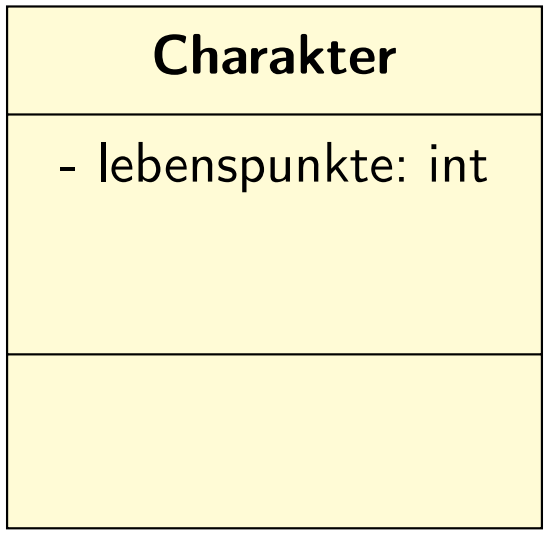
Python-Implementierung:

```
1  class Charakter:
2      def __init__(self, name, lebenspunkte=100, position=(0,
       0)):
3          self.name = name
4          self.lebenspunkte = lebenspunkte
5          self.position = position
6
7      def bewegen(self, x, y):
8          self.position = (self.position[0] + x, self.position
       [1] + y)
9
10     def angreifen(self, ziel):
11         # Implementierung des Angriffs
12         pass
13
```

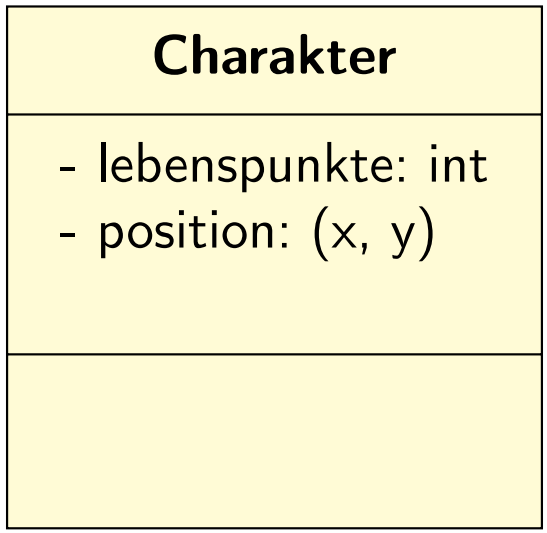
Beispiel: Klasse Charakter



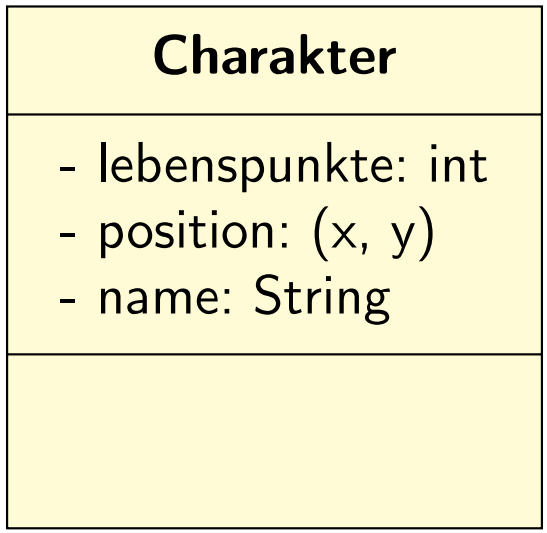
Beispiel: Klasse Charakter



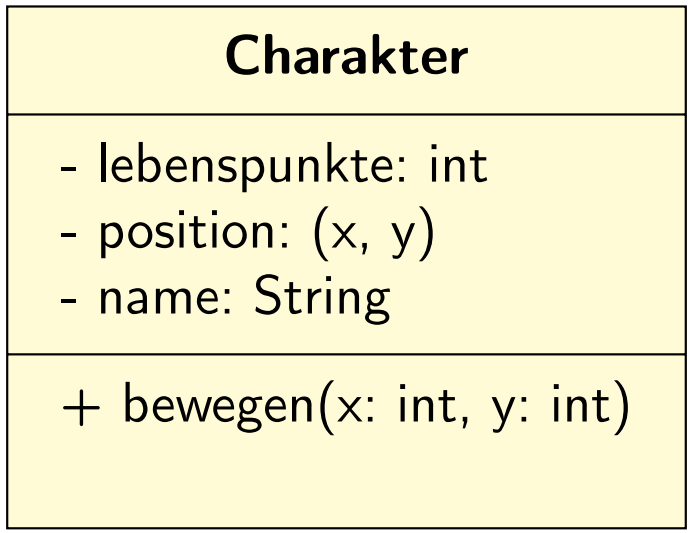
Beispiel: Klasse Charakter



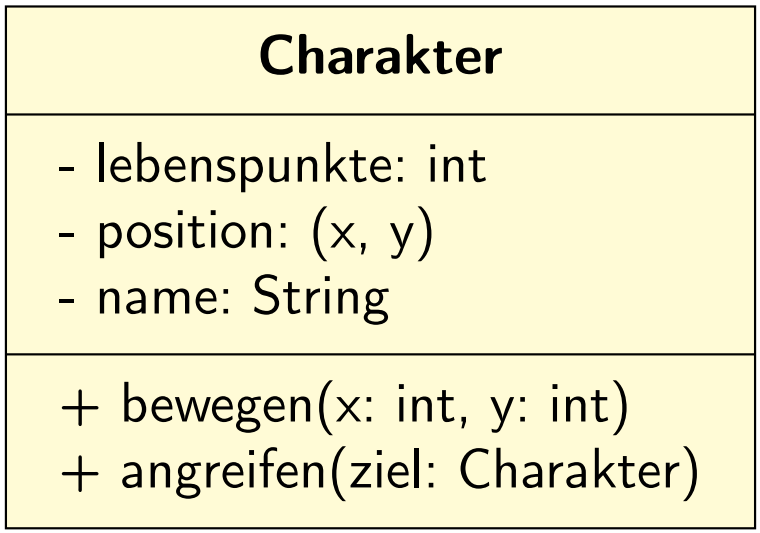
Beispiel: Klasse Charakter



Beispiel: Klasse Charakter



Beispiel: Klasse Charakter

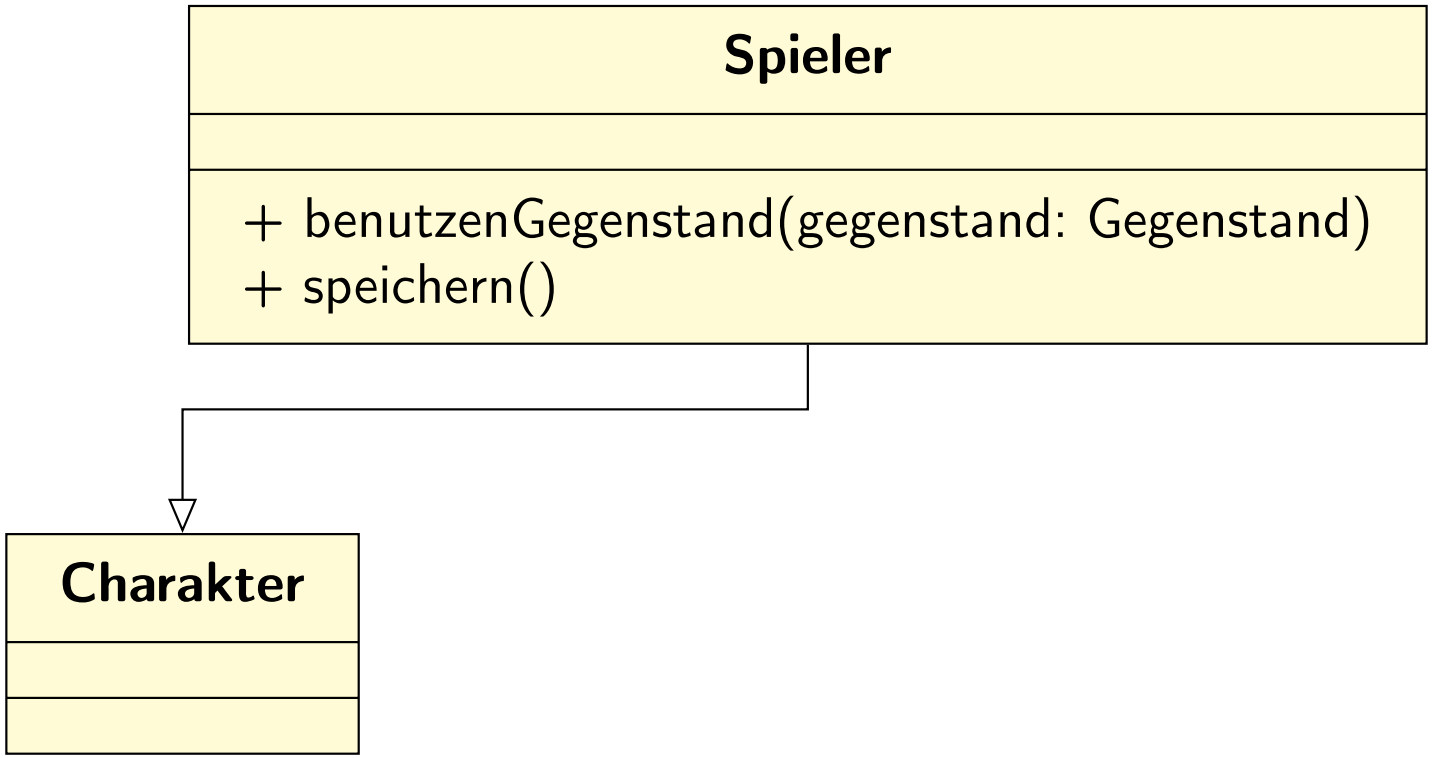


Vererbung

Definition

- **Vererbung**: Ein Konzept, das es erlaubt, Eigenschaften und Verhalten einer bestehenden Klasse zu erben und zu erweitern.
- Die abgeleitete Klasse erbt die Attribute und Methoden der Basisklasse und kann eigene hinzufügen.
- Dies ermöglicht die Wiederverwendung von Code und die Strukturierung von Klassen.

Vererbung



Python-Implementierung:

- Definiere eine Klasse, die von einer anderen Klasse erbt.
- Füge spezifische Attribute und Methoden hinzu.

Vererbung (Fortsetzung)

Beispiel: Klasse Spieler

- Erbt von Klasse Charakter
- Spezifische Methoden: BenutzenGegenstand, Speichern

Python-Implementierung:

```
1 class Spieler(Charakter):
2     def benutzen_gegenstand(self, gegenstand):
3         # Implementierung der Benutzung eines Gegenstands
4         pass
5
6     def speichern(self):
7         # Implementierung des Speicherns des Spielstands
8         pass
9
```

Beispiel: Implementierung in Python

Python-Klassen

- Python bietet eine einfache Syntax für die Definition von Klassen und Vererbung.
- Attribute und Methoden werden ähnlich wie in der UML-Syntax definiert.

- Zeigen Sie die Python-Implementierung für die Charakter- und Spielerklassen.
- Erläutern Sie die Syntax und die Verwendung von Konstruktoren, Attributen und Methoden.