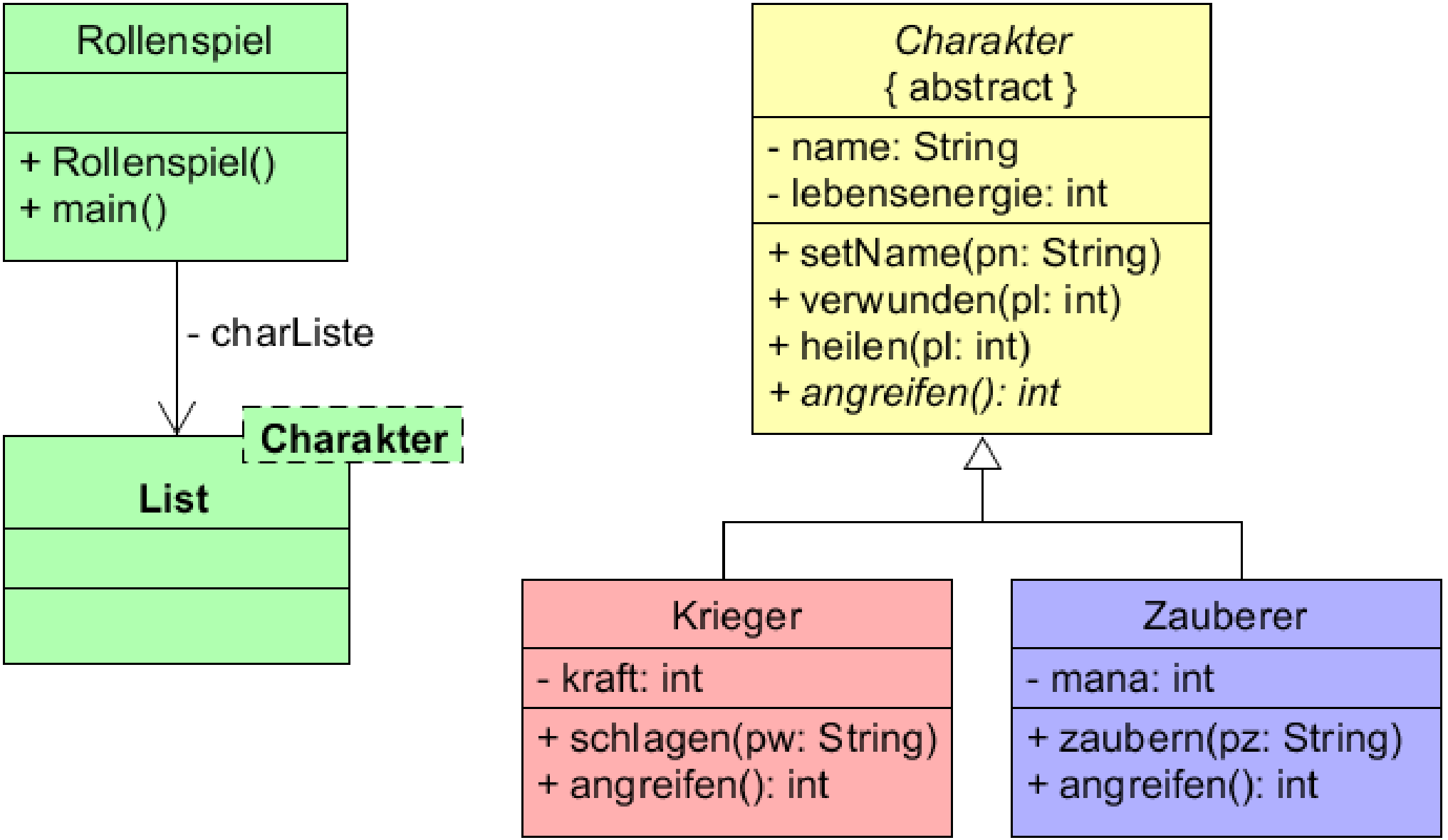
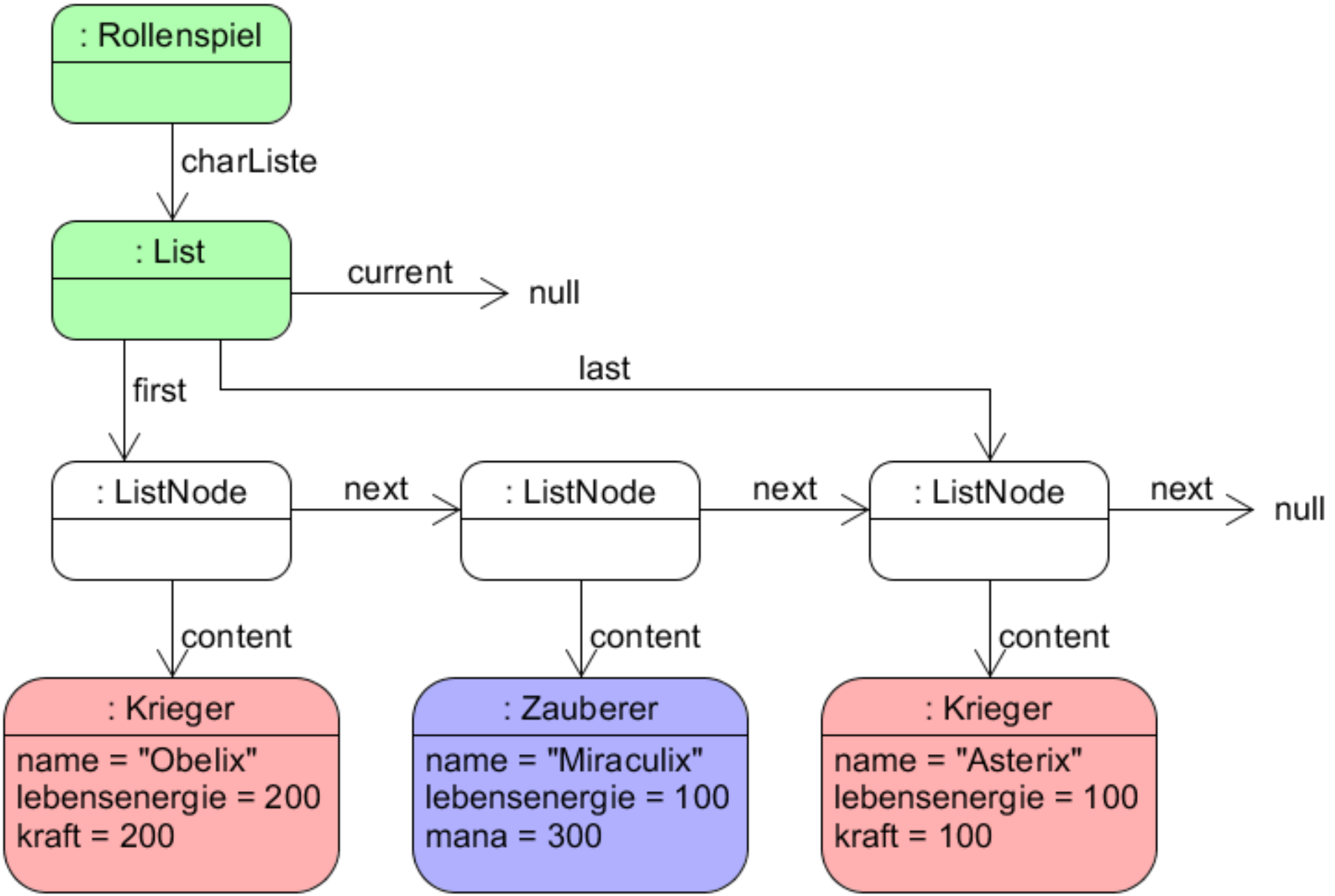
Methoden von Unterklassen ausführen mit Typecast

Beispiel: Liste von Charakter-Objekten



Liste enthält Krieger- und Zauberer-Objekte



Alle Zauberer sollen zaubern !

```
public class Rollenspiel
{
    private List<Charakter> charListe;
    ...
    public void alleZaubern()
    {
        Charakter akt;
        charListe.toFirst();
        while (charListe.hasAccess())
        {
            akt = charListe.getContent();
            akt.zaubern("Feuerball");
        }
```

Alle Zauberer sollen zaubern !

```
...
Charakter akt;
...
akt = charListe.getContent();
akt.zaubern("Feuerball");
...
```

Problem 1: Liste enthält auch Krieger-Objekte

→ Krieger können nicht zaubern!

Problem 2: akt ist Referenz auf Charakter-Objekt
Methode zaubern() gehört zur Klasse Zauberer

→ Aufruf nicht möglich (Compiler: Fehlermeldung)

Alle Zauberer sollen zaubern !

Problem 1: Liste enthält auch Krieger-Objekte

Lösung: Jedes Objekt „fragen“:
Bist du ein Zauberer- oder ein Krieger-Objekt?

Problem 2: akt ist Referenz auf Charakter-Objekt

Lösung: „Typecast“:
Referenz von Charakter in Zauberer umwandeln

1. Klasseninformation abfragen

```
public void alleZaubern()
{
    Charakter akt;
    charListe.toFirst();
    while (charListe.hasAccess())
    {
        akt = charListe.getContent();
        if (akt.getClass() == Zauberer.class)
        {
            ...
        }
```

1. Klasseninformation abfragen

if (akt.getClass() == Zauberer.class)
...

Für **Objekte** JEDER Klasse:
mithilfe von getClass() die Klasse erfragen

akt ist Charakter-Referenz, zeigt aber manchmal auf Zauberer-Objekte
→ getClass() gibt dann die Klasse Zauberer zurück

JEDE **Klasse** hat das Attribut „class"
→ für den Vergleich mit getClass()

2. Typecast

```
public void alleZaubern()
{
    Charakter akt;
    charListe.toFirst();
    while (charListe.hasAccess())
    {
        akt = charListe.getContent();
        if (akt.getClass() == Zauberer.class)
        {
            Zauberer z;
            z = (Zauberer) akt;
            z.zaubern("Feuerball");
        }
        ...
```

2. Typecast

Charakter akt;
...
Zauberer z;
z = (Zauberer) akt;
...

Wir wissen:
Charakter-Referenz akt zeigt auf Zauberer-Objekt

Der Compiler weiß das nicht!
Zauberer-Referenz z kann auf das Objekt zeigen,
aber man muss dem Compiler versichern,
dass es ein Zauberer-Objekt ist.

2. Typecast

Charakter akt;
...
Zauberer z;
z = (Zauberer) akt;
z.zaubern("Feuerball");
...

Mit der Referenz z kann man dann Methoden der Klasse Zauberer aufrufen.

Vorsicht: Programmabsturz !

```
Charakter akt;
...
Zauberer z;
z = (Zauberer) akt;
z.zaubern("Feuerball");
...
```

Man sollte sicher sein, dass akt in diesem Moment wirklich auf ein Zauberer-Objekt zeigt.

Sonst stürzt das Programm hier ab!

Zusammenfassung

getClass() gibt die Klasse eines Objekts zurück.
Die Rückgabe vergleicht man mit <Klasse>.**class**.

Beispiel: if (akt.getClass() == Zauberer.class) ...

Eine Referenz einer Oberklasse kann man mit **Typecast** in eine Referenz einer Unterklasse umwandeln.
Man schreibt die Klasse, in die „gecastet" wird, in Klammern.

Beispiel:
```
Charakter c = charListe.getContent();
Zauberer z = (Zauberer) c;
```

Kürzer: z = (Zauberer)(charListe.getContent());

Autor / Quellen

Autor:

- Christian Pothmann (cpothmann.de)
  Freigegeben unter CC BY-NC-SA 4.0, Juni 2021