Kapitel 16 Mampfi mampft Krümel

Lernziele:

In diesem Kapitel wird gezeigt, wie man ein Zustandsdiagramm als Planung eines Methodenrumpfs nutzen kann.
Wiederholt werden Objektkommunikation, Zustandsdiagramm und Mehrfachauswahl.

16.1 Krümel auf dem Startfeld von Mampfi verschwinden lassen

Eine bereits in Kapitel 15 formulierte Kritik am aktuellen Stand des Projekts war, dass auf dem Startfeld von Mampfi ein Krümel liegt (Abbildung 1). Ziel ist es den Krümel auf dem Startfeld von Mampfi verschwinden zu lassen.

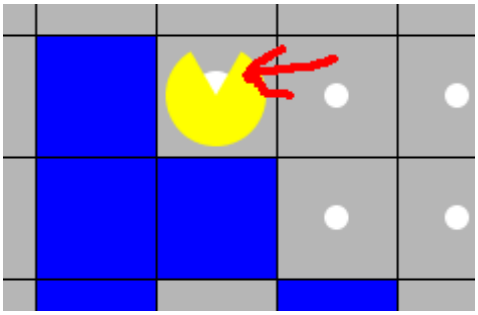


Abbildung 1: Auf dem Startfeld von Mampfi liegt ein Krümel.

Aufgabe 16.1

a) Welches Objekt soll dafür sorgen, dass der Krümel auf dem Startfeld verschwindet, d.h. welches Objekt gibt den Methodenaufruf dazu?
b) Welche Objekte müssen in die Objekt-kommunikation eingebunden werden? Wo gibt es schon passende Methoden, wo müssen noch Methoden ergänzt werden?

Bei der Erzeugung eines Labyrinths werden die Zellen erzeugt, die Gänge erstellt und die Powerkrümel verteilt. Es ist somit sinnvoll auch dort dafür zu sorgen, dass der Krümel vom Startfeld von Mampfi verschwindet. Folgendes ist dazu nötig:

a) Das Labyrinth muss die Startposition von Mampfi kennen.
b) Der Zelle, auf der Mampfi startet muss mitgeteilt werden dort den Krümel zu entfernen.
c) Die Klasse Zelle muss eine entsprechende Methode *KruemelEntfernen* anbieten.

zu c)
Diese Methode gibt es indirekt schon innerhalb der Methode IstMauerSetzen:

```
public void IstMauerSetzen(boolean istMauerNeu)
    {
        istMauer = istMauerNeu;
        zellenSymbol.FuellungSichtbarSetzen(istMauerNeu);
        if ((istMauer == true) )
        {
                if (kruemel != null)
          {
                    kruemel.KruemelSymbolEntfernen();
                    kruemel=null;
          }
        }
    }
```

Der gelb markierte hat die Funktion den Krümel zu entfernen und kann als eigene Methode ausgelagert werden:

```
 public void IstMauerSetzen(boolean istMauerNeu)
    {
        istMauer = istMauerNeu;
        zellenSymbol.FuellungSichtbarSetzen(istMauerNeu);
        if ((istMauer == true) )
        {
            KruemelEntfernen();
        }
    }


    public void KruemelEntfernen()
    {
        if (kruemel != null)
        {
            kruemel.KruemelSymbolEntfernen();
            kruemel=null;
        }
    }
```

zu a)
Man ergänzt die Attribute mampfiStartX und mampfiStartY in der Klasse LABYRINTH. Damit Mampfi selbst dieses Startfeld auch erfahren kann, müssen auch die entsprechenden Geben-Methoden mampfiStartXGeben und mampfiStartYGeben ergänzt werden (Abbildung 2).

zu c)
Im Konstruktor der Klasse LABYRINTH muss ein Aufruf der Methode *KruemelEntfernen* an das passende Zellenobjekt ergänzt werden. Dieser sorgt dafür , dass der Krümel verschwindet.

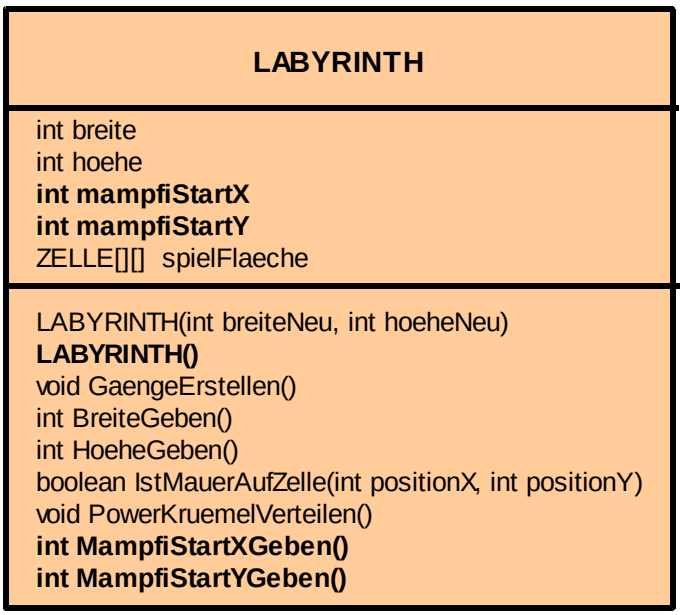


Abbildung 2: Die Klasse LABYRINTH ergänzt um zwei Attribute und zwei Methoden

Aufgabe 16.2

Setze die besprochenen Änderungen a), b) und c) um, d.h.

Ergänze in der Klasse LABYRINTH Attribute für die Startposition von Mampfi und die zugehörigen Geben-Methoden.

Ergänze in der Klasse ZELLE die Methode *KruemelEntfernen* und ergänze den Konstruktor der Klasse LABYRINTH so, dass der Krümel auf dem Startfeld von Mampfi entfernt wird.

Solltest du im letzten Punkt unsicher sein, findest du dazu im Anhang noch ein Sequenzdiagramm.

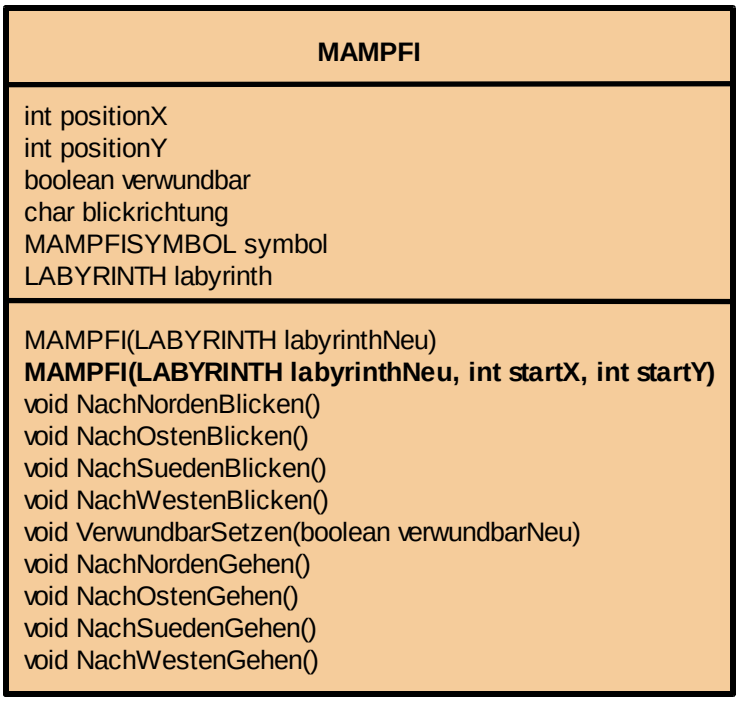
Die gerade ausgeführten Änderungen haben auch Auswirkungen auf die Klassen MAMPFI und SPIELSTEUERUNG.

Aufgabe 16.3

Überlege, welche Änderungen in den Klassen MAMPFI und SPIELSTEUEURUNG nötig sind, um die Veränderungen in der Klasse LABYRINTH sinnvoll einzubetten. Zeichne bei Bedarf ein Sequenzdiagramm zur Planung der Objektkommunikation.

d) Mampfi soll beim Erzeugen in dem Feld starten, dessen Position in den neuen Attributen der Klasse LABYRINTH abgespeichert wurden. Dazu ist ein neuer Konstruktor in der Klasse MAMPFI nötig, der als Eingabeparameter auch die Startpositionen von Mampfi hat. (Abbildung 3)

Abbildung 3: Die Klasse MAMPFI ergänzt um einen weiteren Konstruktor.



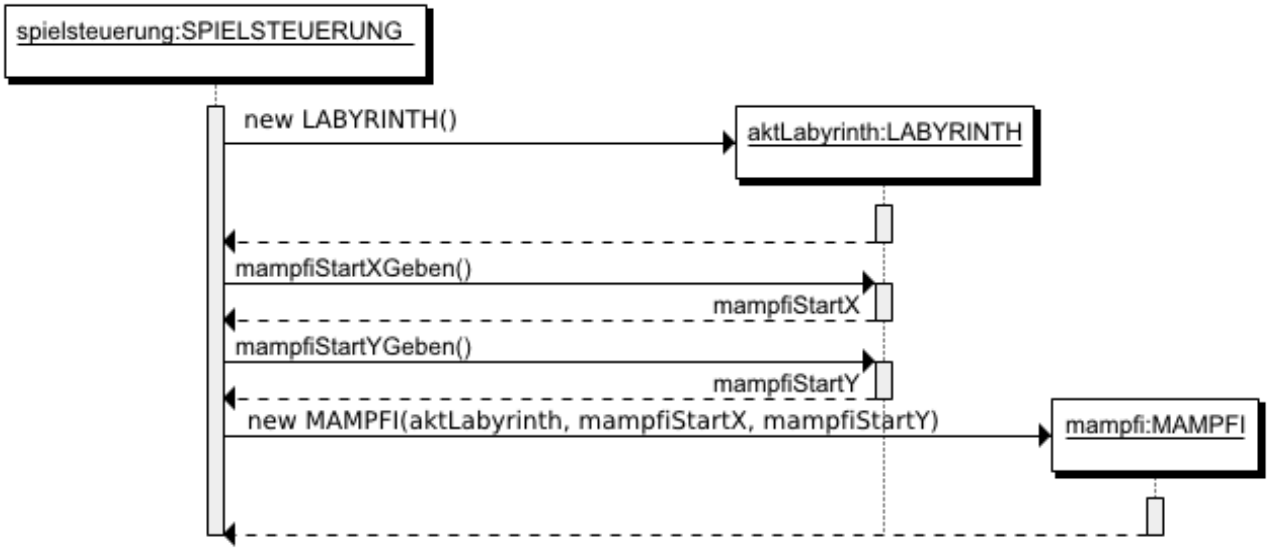


Abbildung 4: Objektkommunikation, um Mampfi beim Erzeugen auf die passende Zelle zu positionieren

e) Das Spielsteuerungsobjekt erzeugt Mampfi. Um ihn auf der richtigen Position (d.h. entsprechend der in dem Labyrinth gespeicherten Startwerte) zu platzieren, muss die Spielsteuerung zuerst die Startposition vom Labyrinth abfragen und dann beim Erzeugen eingeben. Die Objektkommunikation veranschaulicht das Sequenzdiagramm in Abbildung 13.

Hinweis:

Beachte, dass in einem Sequenzdiagramm zu Beginn der Kommunikation noch nicht alle beteiligten Kommunikationspartner schon existieren müssen. In Abbildung 13 wird durch den ersten Methodenaufruf das Labyrinth und durch die vierten Methodenaufruf Mampfi erzeugt. Entsprechend beginnen die Lebenslinien erst an diesen Stellen. Erzeugt werden die Objekte wie gewohnt über den new Operator gefolgt vom Klassennamen. Entsprechend ist die Beschriftung der Nachrichten.

Aufgabe 16.4

a) Ergänze in der Klasse MAMPFI einen Konstruktor, der Eingabewerte hat um die Startpositionen zu setzen. (s. o. Punkt d)

b) Ergänze die Anweisungen im Konstruktor der Klasse SPIELSTEUERUNG so, dass Mampfi auf der Zelle startet, deren Positionen in der Klasse LABYRINTH gespeichert sind. (s. o. Punkt e)

16.2 Zustandsdiagramme als perfekte Planung für Programmabläufe

In Kapitel 15 wurde analysiert, was alles passieren kann/muss, wenn sich Mampfi bewegt. Das Ergebnis als Zustandsdiagramm dargestellt ist nochmals in Abbildung 5 zu sehen.

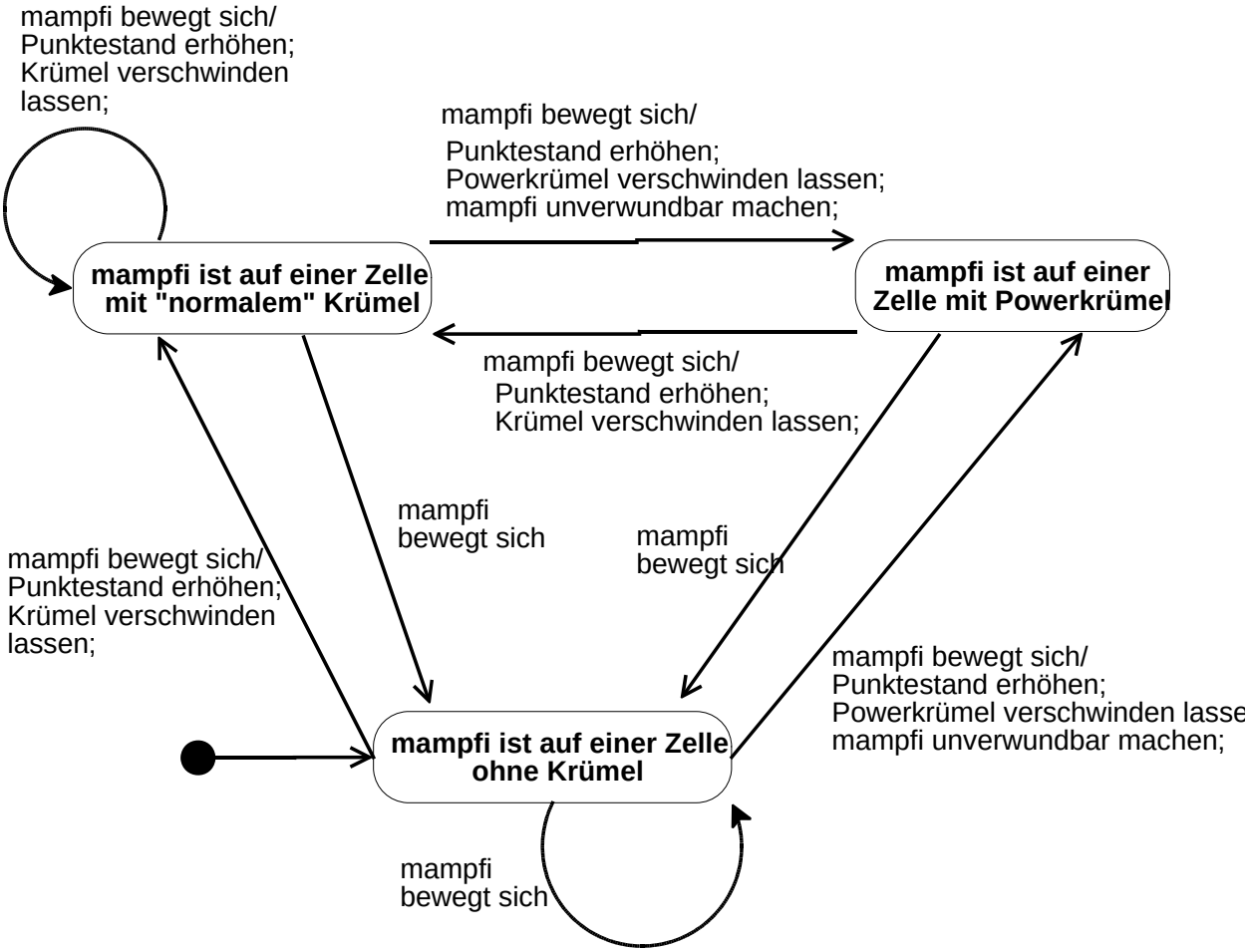


Abbildung 5: Zustandsdiagramm zur Planung des Spielablaufs

Weiterhin wurde entschieden, dass ein Objekt der Klasse Spielsteuerung den Spielablauf kontrollieren soll. Aus dem Zustandsdiagramm ist zu sehen, dass der Ablauf davon abhängt, ob auf der Zelle, auf der sich Mampfi befindet ein normaler Krümel, ein Powerkrümel oder kein Krümel liegt. Hat die Spielsteuerung diese Information, so lassen sich die notwendigen Methodenaufrufe aus Abbildung 5 leicht ableiten.

Aufgabe 16.5

Wie kann die Spielsteuerung die Information erhalten, in welchem der drei Zustände aus Abbildung 5 sich die Spielsituation befindet?

Zeichne als Planung ein Sequenzdiagramm und Kennzeichne die Methoden, die in den betreffenden Klassen noch nicht existieren farbig.

16.3 Ist ein Krümel für mampfi da?

Ob mampfi einen Krümel auf einer gerade erreichten Zelle findet, weiß natürlich die Zelle selbst am besten. Um gebündelt diese Information nach außen weiterzugeben, muss die Klasse ZELLE um eine Methode *KruemelBelegungAnzeigen* erweitert werden. Über ein Zeichen (char) als Rückgabewert der Methode wird die gesuchte Information ausgegeben:

'L' :Die Zelle ist leer.
'P': Auf der Zelle liegt ein Powerkruemel, der mampfi unverletzlich macht.
'N': Auf der Zelle liegt ein normaler Krümel.

Die Information kennt die Zelle jedoch nur selbst, wenn die Zelle keinen Krümel hat. In diesem Fall ist im Referenzattribut null gespeichert.
Sollte es einen Krümel geben, muss die Zelle den Wert des Attributs machtUnverletzlich des Krümels abfragen. Ist die Antwort true, dann liegt ein Powerkrümel auf der Zelle, sonst ein normaler.
Die vollständige Objektkommunikation zeigen für die drei beschriebenen Fälle die Abbildungen 6, 7 und 8:

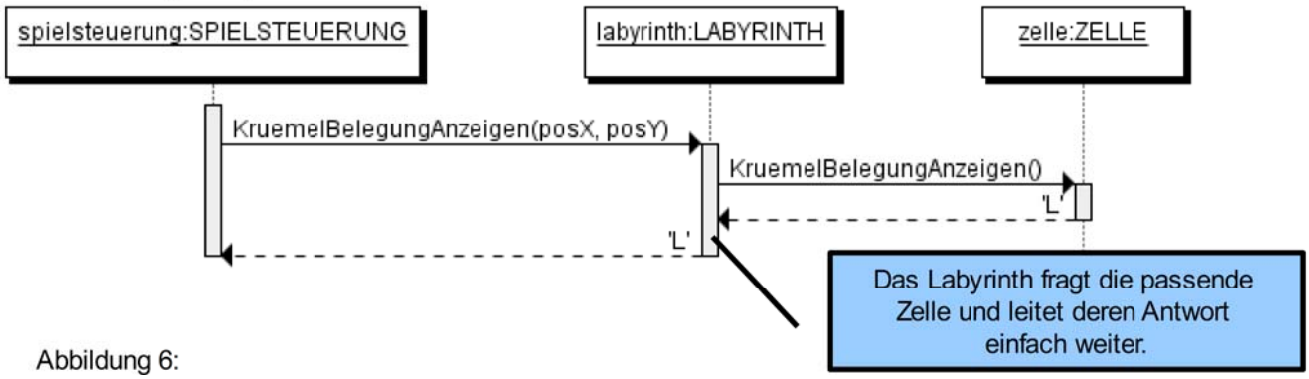


Abbildung 6:
Objektkommunikation beim Aufruf der Methode *KruemelBelegungAnzeigen*, falls es keinen gibt.

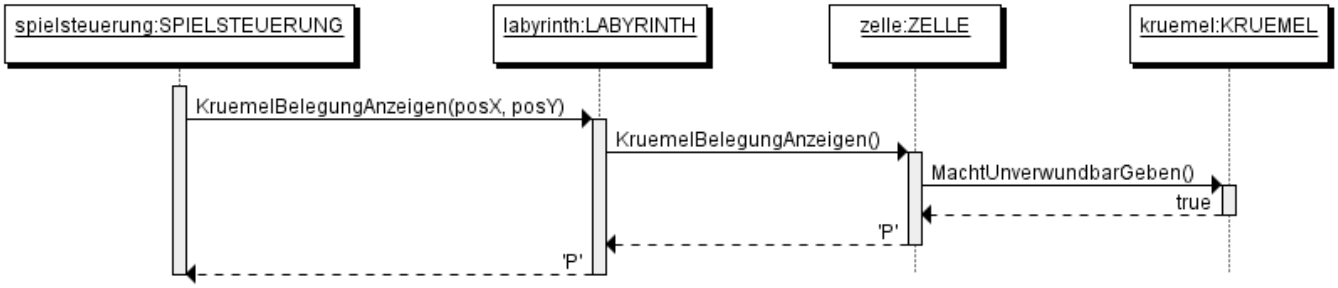


Abbildung 7: Objektkommunikation beim Aufruf der Methode *KruemelBelegungAnzeigen*, falls ein Powerkrümel auf der Zelle liegt.

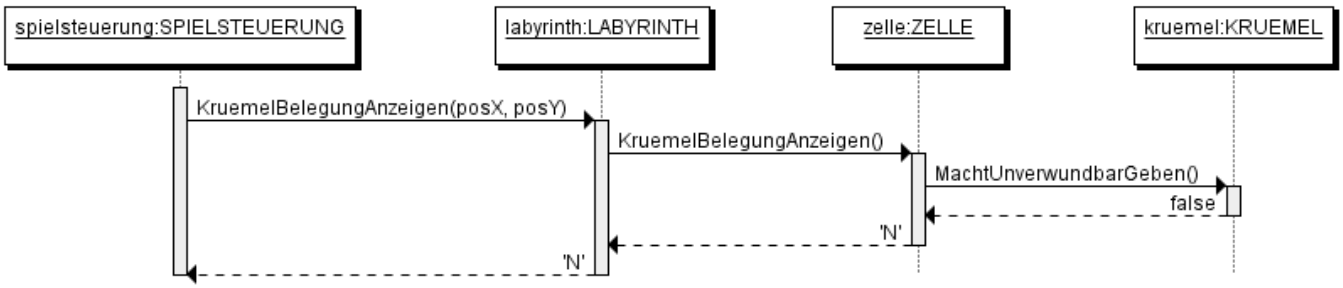


Abbildung 8: Objektkommuikation beim Aufruf der Methode *KruemelBelegungAnzeigen*, falls ein normaler Krümel auf der Zelle liegt.

Aufgabe 16.6

Warum benötigt die Methode *KruemelBelegungAnzeigen* der Klasse LABYRINTH Eingabewerte, nicht jedoch die gleich lautende Methode der Klasse ZELLE?

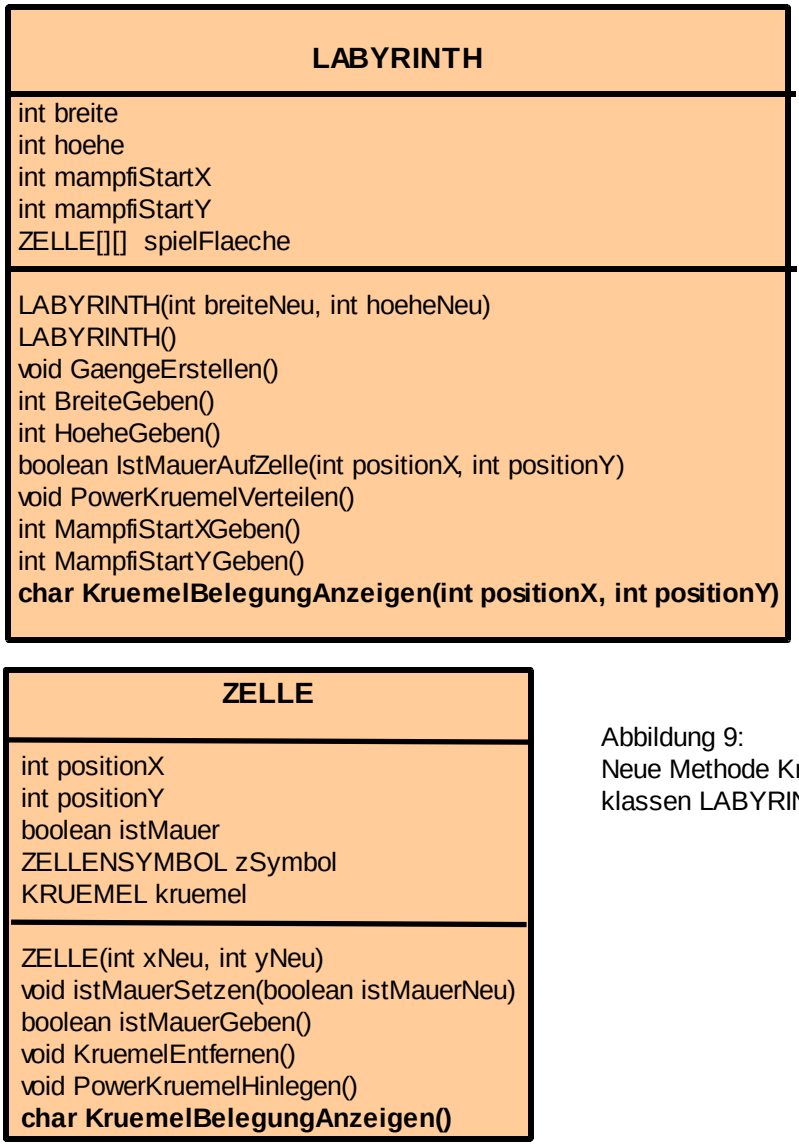


Abbildung 9:
Neue Methode KruemelBelegungAnzeigen in den klassen LABYRINTH und ZELLE

Aufgabe 16.7

Ergänze in den Klassen LABYRINTH und ZELLE die die Methode *KruemelbelegungAnzeigen* (und falls nötig in der Klasse KRUEMEL die Methode *MachtUnverwundbarGeben)*

Hinweise:

- Solltest du dich unsicher fühlen findest du Tipps im Anhang. Versuche es jedoch zuerst alleine!
- Selbstverständlich sollte die Methoden auch getestet werden. Da das Testen nach dem nächsten Kapitel jedoch unvergleichlich einfacher ist, kann an dieser Stelle ausnahmsweise darauf verzichtet werden.

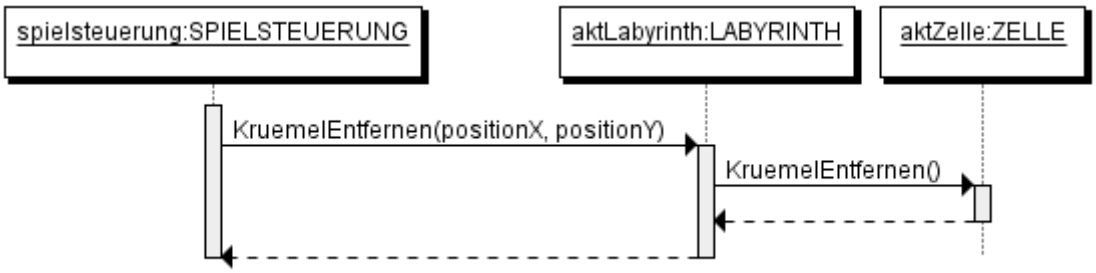
16.4 Abhängig von der Krümelbelegung den Spielzug auswerten

In dem Zustandsdiagramm aus Abbildung 5 kann man einfach ablesen, was abhängig von der Krümelbelegung der Zelle zu tun ist. Als ausgelöste Aktionen sind aufgetreten:
a) den Punktestand erhöhen
b) den Krümel verschwinden lassen
c) Mampfi unverwundbar machen

Aufgabe 16.8
Welche Methoden müssen in den Fällen a),b) und c) aufgerufen werden´?
Welche davon kann die Spielsteuerung direkt aufrufen?
Wo müssen Ergänzungen durchgeführt werden, damit die Spielsteuerung auch die anderen Aktionen auslösen kann?

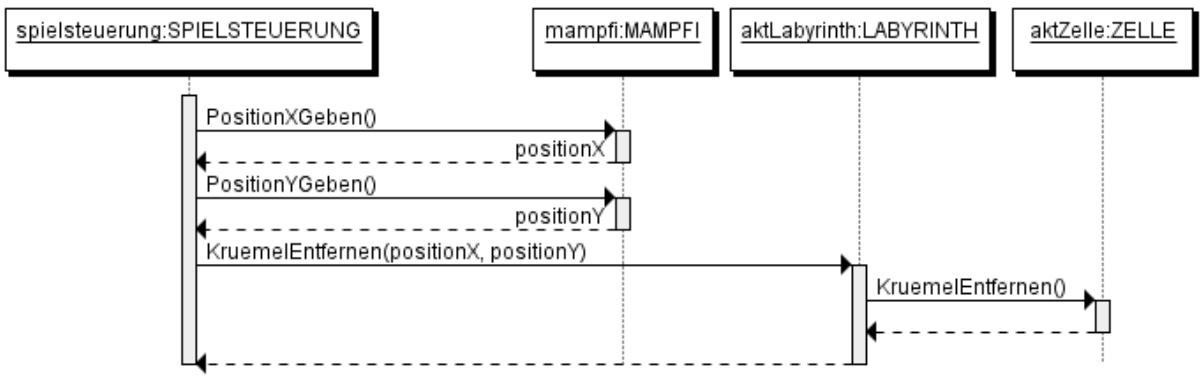
zu c)
Da die Spielsteuerung eine Referenz auf mampfi hat, ist diese Aktion mit dem Methodenaufruf *mampfi.VerwundbarSetzen(false)* leicht erledigt.

zu b)
Die Spielsteuerung hat keine direkte Referenz zu Zellen, sondern kann nur mit Hilfe des Labyrinths als Zwischenstation mit ihnen kommunizieren. In der Klasse LABYRINTH muss eine Methode *KruemelEntfernen* ergänzt werden, die diese Kommunikation ermöglicht. Diese Methode muss als Eingabewerte die Positionsangaben der gewünschten Zelle haben. Das folgende Sequenzdiagramm veranschaulicht die Objektkommunikation.



Aufgabe 16.9
Woher erhält die Spielsteuerung die passende Position?

Die Spielsteuerung kann Mampfi nach seiner aktuellen Position fragen.



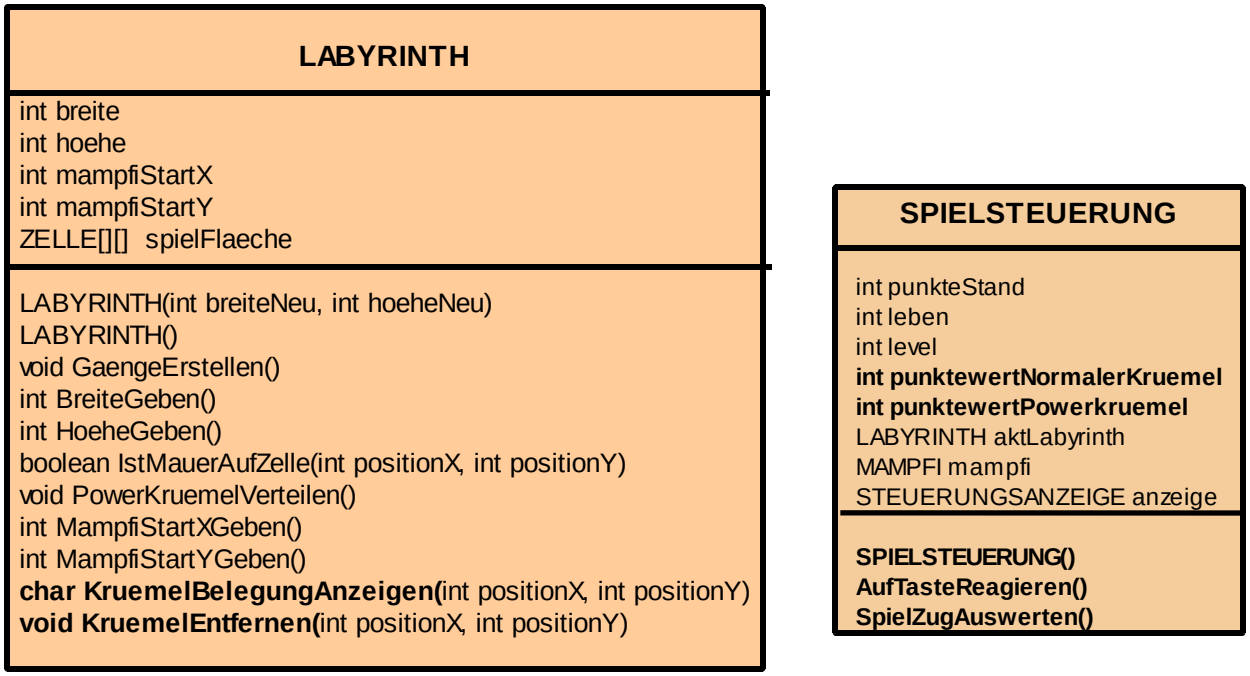
zu a)
Der Punktestand kann von der Spielsteuerung erhöht werden. Es muss jedoch festgelegt werden, wie viel Punkte ein normaler Krümel bzw. ein Power Wert ist. Dies kann über zwei neue Attribute der Spielsteuerung punktewertNormalerKruemel und punktewertPowerkruemel gemacht werden. Der Wert dieser Attribute muss im Konstruktor der Klasse SPIELSTEUERUNG festgelegt werden.

Alle Bestandteile des Zustandsdiagramms aus Abbildung 5 sind nun besprochen: Das bestimmen des Zustands (Kapitel 16.3) und die ausgelösten Aktionen (Kapitel 16.4).
Es ist sinnvoll dies in einer neuen Methode umzusetzen. *SpielzugAuswerten* ist ein geeigneter Name.

Aufgabe 16.10

a) Ergänze die besprochenen Änderungen in deinem Projekt. Als Hilfe sind in den erweiterten Klassendiagrammen auf der nächsten Seite die Neuerungen fett markiert. Du solltest dir solche ein Klassendiagramme selbst zur Planung anfertigen.
   Hinweise
   - Bei Bedarf kannst du weitere Tipps im Anhang aufrufen.
   - Verwende bei der Methode *SpielzugAuswerten* die Mehrfachauswahl!

b) Das Testen wird leichter, wenn du die Tastatursteuerung aus Kapitel 15 nutzt. Ergänze die Methode AufTasteReagieren der Klasse SPIELSTEUERUNG so, dass beim Drücken der Taste 'Z' der Spielzug ausgewertet wird und beim Drücken der Taste 'V' Mampfi wieder verwundbar wird.



16.5 Zusammenfassung

Aufgabe 16.11
Nachdem Zustandsdiagramme in Kapitel 15 vorgestellt wurden, halfen sie in diesem Kapitel zur Planung der Methode SpielzugAuswerten.

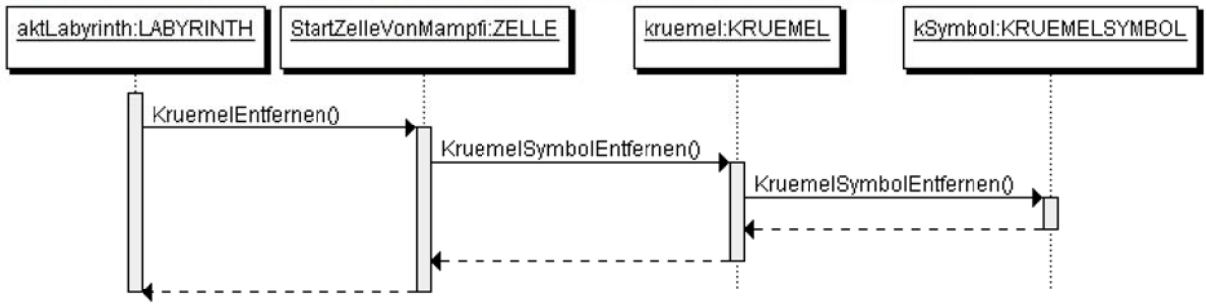
a) Vergleiche den Quelltext der Methode *SpielzugAuswerten* mit dem Zustandsdiagramm aus Abbildung 5 und stelle die Bezüge her.
b) Bisher wurde zur Veranschaulichung von Methodenrümpfen Struktogramme eingesetzt. Vergleiche die Einsatzmöglichkeit von Struktogrammen mit denen von Zustandsdiagrammen.

Aufgabe 16.12 **Zweispielermodus**(Fortsetzung Aufgabe 15.14)
Mache die Ergänzungen in diesem Kapitel auch im Zweispielermodus möglich.

Anhang A: Tipps zu den Aufgaben

Tipps zu Aufgabe 16.2:

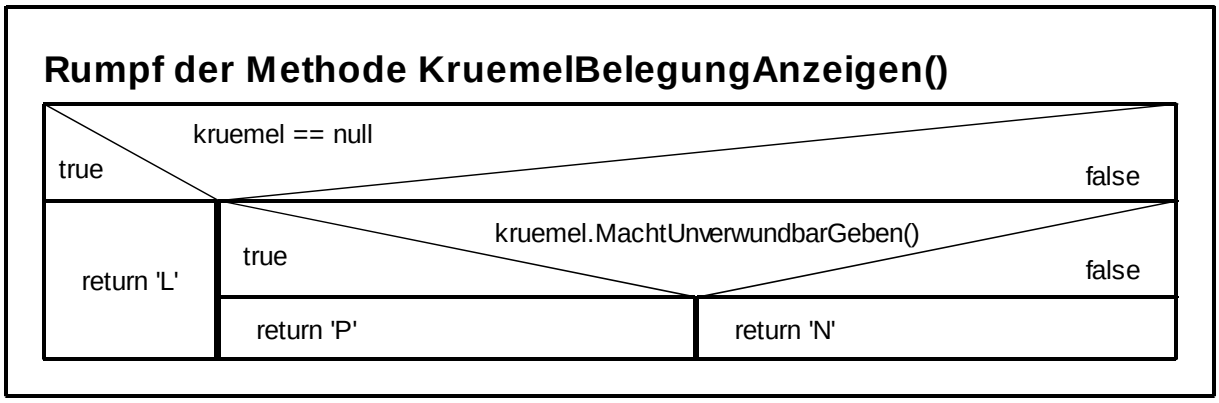
Objektkommunikation beim entfernen eines Krümels von der Zelle:



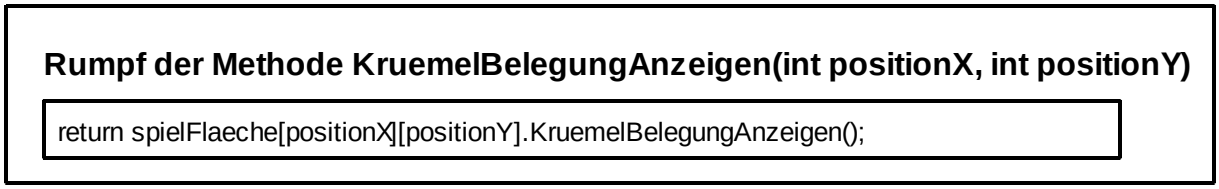
Hinweis:
Neu verfasst werden muss nur die Methode *KruemelEntfernen*. Die beiden Methoden *KruemelSymbolEntfernen* sind bereits fertig implementiert.

Tipps zu Aufgabe 16.7:

Das Struktogramm zur Methode *KruemelBelegungAnzeigen* der Klasse **ZELLE** hat folgende Form:



Das Struktogramm zur Methode *KruemelBelegungAnzeigen* der Klasse **LABYRINTH** hat folgende Form:



Tipps zu Aufgabe 16.10:

- Soweit noch nicht vorhanden müssen in der Klasse MAMPFI die Methoden *PositionXGeben* und *PositionYGeben* ergänzt werden.
- Struktogramm zur Methode *KruemelEntfernen* der Klasse LABYRINTH

Rumpf der Methode KruemelEntfernen(int positionX, int positionY)

```
spielFlaeche[positionX][positionY].KruemelEntfernen();
```

- Zur Methode *SpielAuswerten:*
  + Die aktuelle Position von Mampfi kann man innnerhalb der Methode *SpielAuswerten* als lokale Attribute zwischenspeichern. Lokale Attribute müssen auch deklariert werden. Dies geschieht innerhalb des Methodenrumpfs:

```
public void SpielzugAuswerten()
{
     //Deklaration
     int positionX;
     int positionY;
     //Initialisierung
     positionX = mampfi.PositionXGeben();
     positionY = mampfi.PositionYGeben();

     //...
}
```

  + Das Auswahlkriterium der Mehrfachauswahl in der Methode *SpielAuswerten* ist der Methodenaufruf *aktLabyrinth.KruemelBelegungAnzeigen(positionX, positionY)., denn e*r liefert die Antwort, in welchem der Zustände 'L', 'P' oder 'N' man sich befindet.
  + Achtung: Nicht vergessen die beiden neuen Attribute der Klasse SPIELSTEUERUNG auch im Konstruktor zu initialisieren.