Kapitel 13 Krümel im Labyrinth

Lernziele:
Löschen von Referenzen
Vertiefungen zur Objekt-kommunikation.

13.0 Vorbereitungen

Speichert die Datei KRUEMELSYMBOL.java und fügt sie in Euer BlueJ-Projekt ein (Bearbeiten --> Klasse aus Datei hinzufügen).

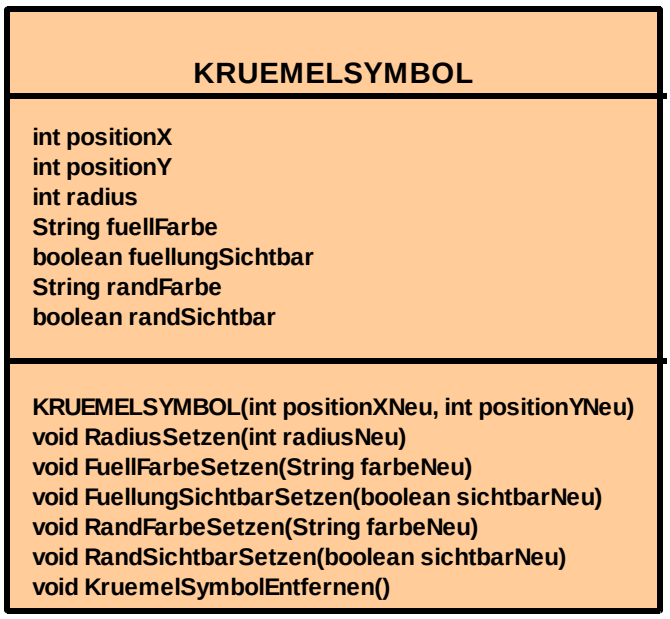


Abbildung 1:
Klasse KRUEMELSYMBOL zur Darstellung von Krümeln

13.1 Die Klasse KRUEMEL

Ziel dieses Kapitels ist es die Krümel im Spiel hinzuzufügen, die dann von Mampfi verspeist werden können.

Aufgabe 13.1

a) Überlege sinnvolle Attribute und Methoden für Klasse KRUEMEL. Beschränke dich dabei auf Wesentliches, Erweiterungen sind später noch möglich. Stelle dein Ergebnis als Klassenkarte dar.
b) Zu welchen anderen Klassen haben Objekte der Klasse KRUEMEL Beziehungen. Zeichne als Antwort ein Klassendiagramm mit (beschrifteten!) Beziehungen, ohne Attribute und Methoden.
c) Ergänze in der Klassenkarte von KRUEMEL die Datentypen, um ein erweitertes Klassendiagramm zu erhalten.
d) Wie werden die Beziehungen aus b) umgesetzt? Ergänze das erweiterte Klassendiagramm von KRUEMEL. Bei welcher Klasse muss wegen der Beziehungen ebenfalls eine Ergänzung vorgenommen werden?

Hinweis: Solltest du bei der Lösung zu dieser Aufgabe nicht sicher sein, findest du im Anhang verschiedene Fragen und Tipps.

Computerspiele sind komplexe Programme. I. A. ist es deshalb sinnvoll, nicht die gesamte Spiellogik in einem Schritt umzusetzen, sondern Einschränkungen zu machen, um zunächst die Komplexität gering zu halten. Schrittweise können dann die Einschränkungen aufgehoben werden.

Aufgabe 13.2

Benenne ausgehend vom Entwurf in Aufgabe 13.1. Schritte, die einfach umsetzbar sind und solche, die komplexer sind.
Finde Einschränkungen, die nicht hinderlich sind, "das Füllen des Labyrinths mit Krümeln" zu testen. Die Einschränkungen sollen die Komplexität reduzieren und damit das Programmieren und Testen erleichtern.

© brichzin@tcs.ifi.lmu.de, 2007

Leicht in Java umsetzen lässt sich die Klasse KRUEMEL inklusiv der Beziehung zur Klasse KRUEMELSYMBOL. Dieser Teilschritt lässt sich auch gut testen, z.B.

- Wird beim Erzeugen eines Krümels auch ein Symbol an der passenden Position erzeugt?
- Wird bei einem Aufruf der Methode *MachtUnverwundbarSetzen* der entsprechende Attributwert richtig gesetzt und das Krümelsymbol angepasst?

KRUEMEL
int positionX int positionY boolean machtUnverwundbar KRUEMELSYMBOL kSymbol
KRUEMEL(int positionXNeu, int positionYNeu) void MachtUnverwundbarSetzen(boolean machtUnverwundbarNeu) boolean MachtUnverwundbarGeben()

Abbildung 2: Klasse KRUEMEL

Komplexer wird es, wenn man das gesamte Labyrinth betrachtet. Die neuen Klassen KRUEMEL und KRUEMELSYMBOL werden über Beziehungen in die bestehende Struktur integriert (Abbildung 3).

Aufgabe 13.3

Wie viele Objekte werden beim Erzeugen eines Labyrinths mit den Maßen 10 x 10 erzeugt, wenn auf jeder Zelle ein Krümel liegt (Abbildung 4)?

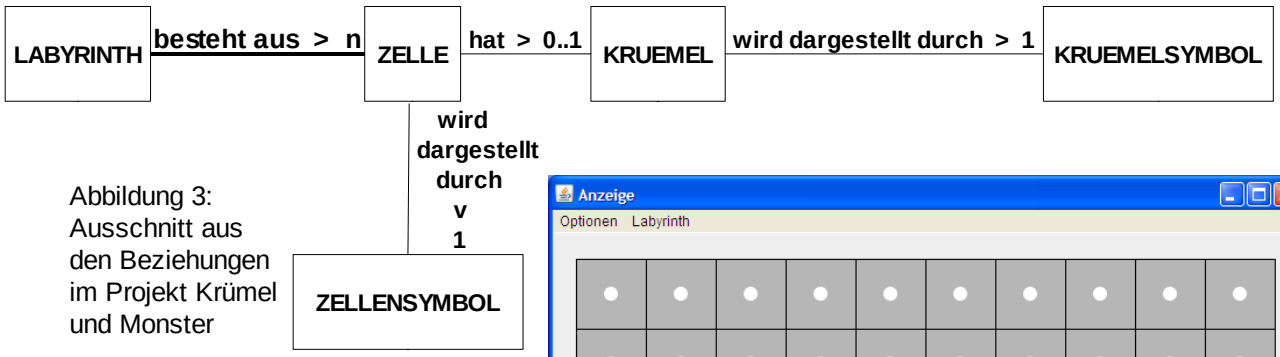


Abbildung 3: Ausschnitt aus den Beziehungen im Projekt Krümel und Monster

Beim Erzeugen eines Labyrinths wie in Abbildung 4 müssen neben dem Labyrinth automatisch 400 weitere Objekte erzeugt werden. Um die Komplexität im Rahmen zu halten, wurden folgende Einschränkungen gemacht.

- Im Labyrinth sind keine Mauern.
- Direkt nach dem Erzeugen des Labyrinths liegt auf jeder Zelle ein Krümel.
- In der Ausgangssituation gibt es im Labyrinth nur einfache Krümel.

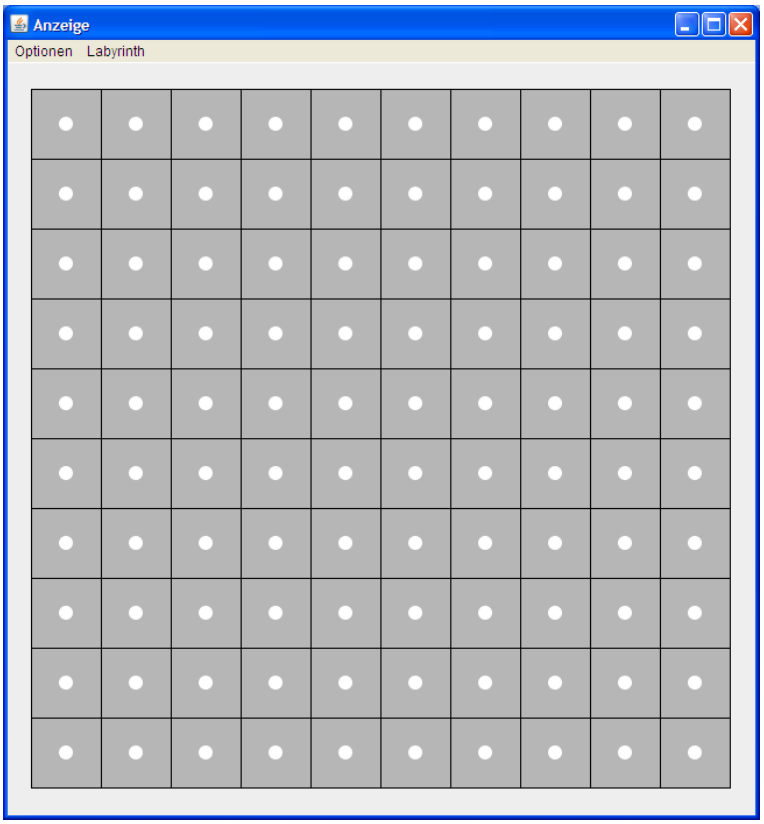


Abbildung 3: Ein Labyrinth mit Krümeln

© brichzin@tcs.ifi.lmu.de, 2007

Aufgabe 13.4
Setze den Entwurf aus Aufgabe 13.1.**schrittweise** in Java um.
Berücksichtige dabei die auf der letzten Seite vorgeschlagenen Einschränkungen.
Teste nach jedem Teilschritt und dokumentiere (über Javadoc-Kommentare) alle Neuerungen.
Hinweise:
Aus Sicht des bisher Erlernten kannst du diese Aufgaben völlig alleine lösen. Vergiss nicht nach Änderungen ausführlich zu testen, ob dein Ziel erreicht wurde.
Solltest du dich unsicher fühlen, findest du im Anhang verschiedene Fragen und Tipps, die dir helfen sollen die nächsten Schritte zu finden.

13.2 Krümel entfernen

Aufgabe 13.5
Erzeuge ein (mit Krümeln gefülltes) Labyrinth wie in Abbildung 3. Rufe die Methode *GaengeErstellen* des Labyrinths auf.
Womit kannst du nicht zufrieden sein?

Krümel auf Mauern machen keinen Sinn. Hier muss die bisherige Lösung verbessert werden.

Aufgabe 13.6
Ausgangspunkt ist ein mit Krümeln gefülltes Labyrinth. Wird nun eine Zelle zur Mauer, so muss, soweit vorhanden, der Krümel auf der Zelle entfernt werden.
- Wie kann ein Krümel entfernt werden?
- In welcher Methode muss ein Methodenaufruf zum Entfernen des Krümels ergänzt werden.
- Welche Objekte sind beim Entfernen eines Krümels beteiligt? Wie müssen sie miteinander kommunizieren?

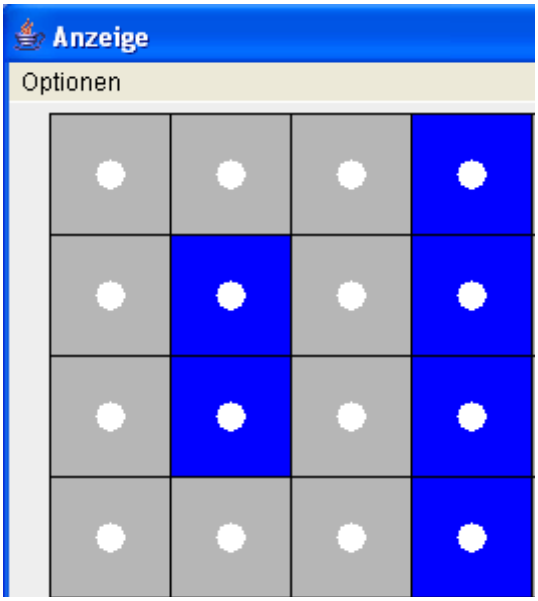


Abbildung 4: Krümel auf Mauern

© brichzin@tcs.ifi.lmu.de, 2007

Folgende Objekte sind beim Entfernen eines Krümels beteiligt:
- Ein Objekt der Klasse ZELLE:
  Entsprechend der Situation in Aufgabe 13.5 muss ein Krümel entfernt werden, wenn bei einer Zelle die Methode *IstMauerSetzen* mit dem Eingabewert `true` aufgerufen wird.
- Das zugehörige Objekt der Klasse KRUEMEL:
  Der Krümel, der auf der Zelle sitzt, soll eben gelöscht werden.
- Das zugehörige Objekt der Klasse KRUEMELSYMBOL:
  Es ist wichtig, dass in der Anzeige auch das Symbolobjekt verschwindet.

Es gibt nun verschiedene Ansätze zur Lösung.
a) Man sorgt nur dafür, dass das Krümelsymbol in der Anzeige verschwindet.
b) Man löscht den Krümel und sein Symbol.

zu a)
Das Verschwinden des Symbols kann man erreichen, in dem man den Wert des Attributs fuellungSichtbar auf false setzt, oder den Radius auf den Wert 0. Die Objektkommunikation für diesen Lösungsansatz zeigt folgendes Sequenzdiagramm:

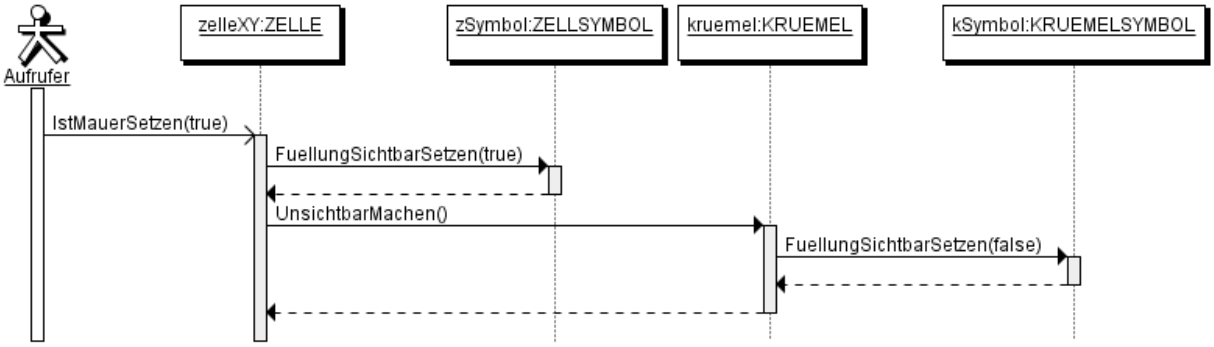


Abbildung 5: Objektkommunikation, die dafür sorgt, dass beim Erstellen einer Mauer der Krümel auf der Zelle verschwindet.

Beachte folgendes bei dieser Lösung:
Zu Beginn wird die Methode *IstMauerSetzen* der betreffenden Zelle aufgerufen. Dadurch wird die Zelle zum aktiven Objekt. Da es jedoch keine Referenz von der Zelle zum Krümelsymbol gibt, muss die Aufforderung den Wert des Attributs fuellungSichtbar auf `true` zu setzen über die "Zwischenstation" Krümel kommuniziert werden. Dazu muss in der Klasse KRUEMEL eine neue Methode *UnsichtbarMachen* geschrieben werden, deren einzige Aufgabe ist, die Methode *FuellungSichtbarSetzen* mit dem Eingabewert `true` aufzurufen.

Diese Lösungsmöglichkeit ist sicher zufriedenstellend für einen Spieler von Krümel & Monster. Aus Sicht der Objektmodellierung ist diese Lösung nicht schön, denn die Krümel(-objekte) existieren weiter, obwohl sie nicht mehr verwendet werden und nicht mehr sichtbar sind. Bei großen Softwareprojekten ist es sehr wichtig nicht mehr verwendete Objekte zu löschen, da es sonst Nachteile im Leistungsverhalten (engl. performane) der Software hinsichtlich Laufzeit und Speicher geben kann.

zu b)
In Java gibt es keine Methode um Objekte zu löschen. Jedoch sorgt ein Programm, die sogenannte automatische Speicherbereinigung (engl. garbage collection) dafür, dass alle Objekte, die nicht verwendet werden gelöscht werden. Dabei ist ein nicht verwendetes Objekt ein Objekt, auf das keine Referenz zeigt.

 © brichzin@tcs.ifi.lmu.de, 2007

Daraus folgt, dass man Objekte in Java löscht, indem man dafür sorgt, dass keine Referenzen auf diese Objekte zeigen, d.h. indem man alle Referenzen auf diese Objekte löscht. Abbildung 6 zeigt die Referenzen der beteiligten Objekte.
Hinweis:
Neben den von uns angelegten Referenzen gibt es noch Referenzen vom Backend auf die Symbolobjekte. Nur so kann das Backend (als Bibliothek in BlueJ integriert) seine Aufgabe, die Symbolobjekte zu zeichnen.

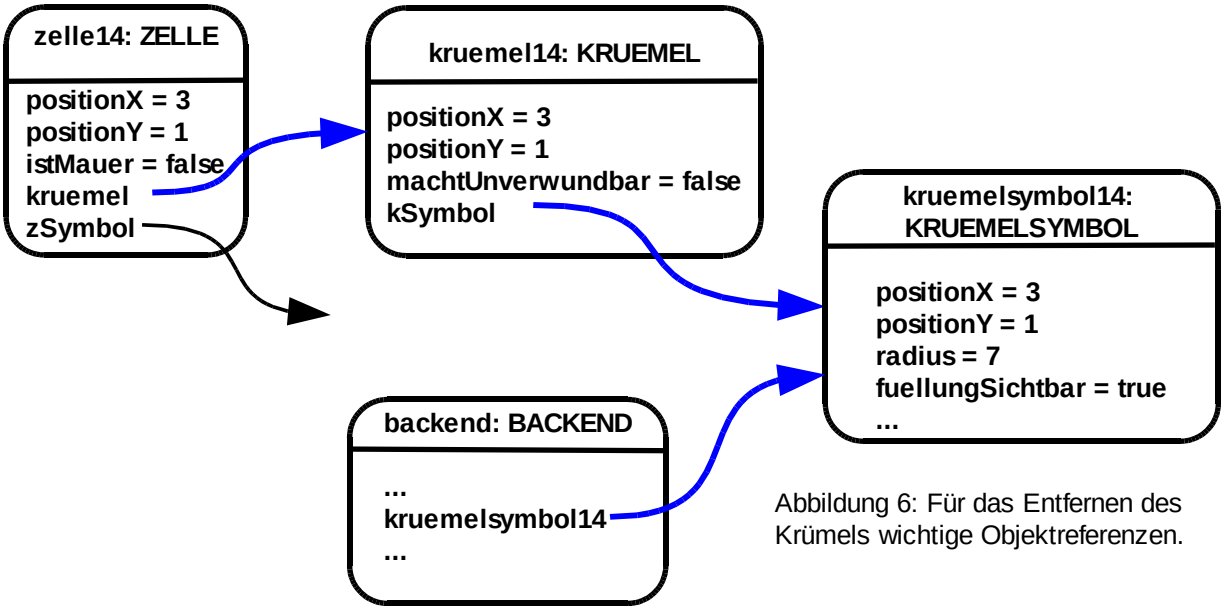


Abbildung 6: Für das Entfernen des Krümels wichtige Objektreferenzen.

Um einen Krümel und sein Symbol zu löschen, müssen nun alle Referenzen, die auf diese beiden Objekte zeigen gelöscht werden.
Das Löschen einer Referenz erfolgt dadurch, dass man dem Referenzattribut den "Wert" null zuweist. Wird beispielsweise beim Objekt kruemel14 die Anweisung

```
kSymbol = null;
```

ausgeführt, dann verändert sich der Objektzustand in Abbildung 6 wie folgt:

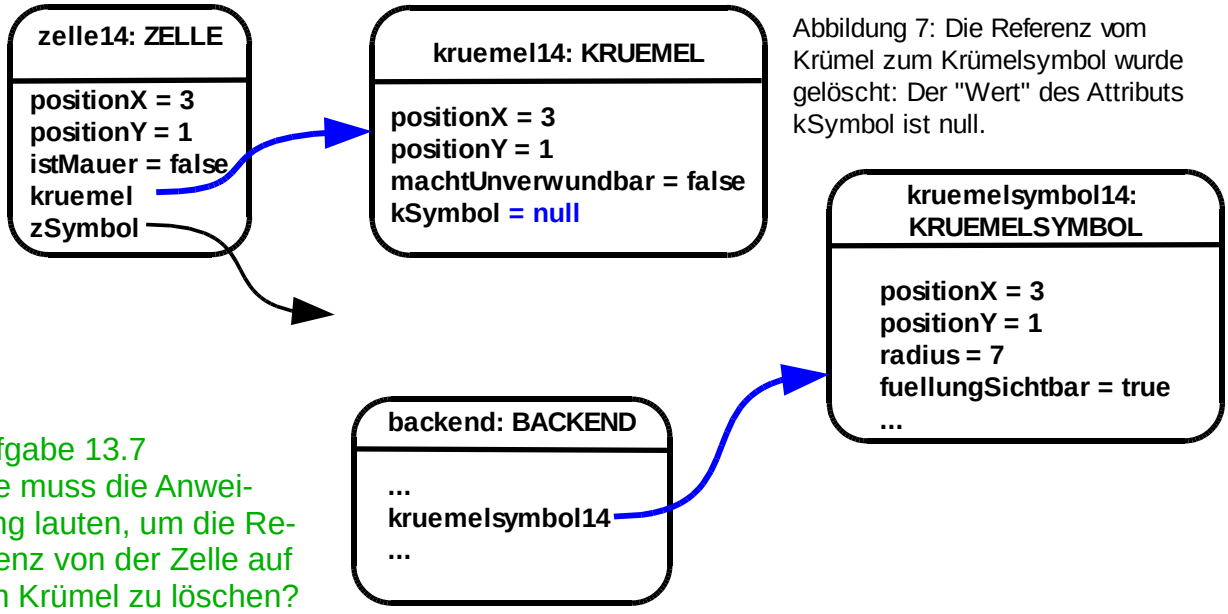


Abbildung 7: Die Referenz vom Krümel zum Krümelsymbol wurde gelöscht: Der "Wert" des Attributs kSymbol ist null.

Aufgabe 13.7
Wie muss die Anweisung lauten, um die Referenz von der Zelle auf den Krümel zu löschen?
Warum lässt sich nicht auf die gleiche Art und Weise die Referenz vom Backend auf das Krümelsymbol entfernen?

© brichzin@tcs.ifi.lmu.de, 2007

Da wir den Quelltext des Backends nicht verändern können, ist es für uns auch nicht möglich dort den Wert eines Referenzattributs auf null zu setzen. Das Krümelsymbol bietet jedoch die Methode *KruemelSymbolEntfernen* an (Abbildung 1), mit der die Referenz zwischen dem Backend und dem Krümelsymbol gelöscht werden kann.

13.3 Miteinander „reden“ ist wichtig

Aufgabe 13.8

In welcher Reihenfolge müssen die Referenzen (blaue Pfeile in Abbildung 6) gelöscht werden?

Beachte bei der Antwort, dass ähnlich zu dem Sequenzdiagramm in Abbildung 5 die Objektkommunikation nach dem Aufruf der Methode *IstMauerSetzen(true)* vom Zellenobjekt ausgeht.

Wie sieht die Objektkommunikation zwischen den Objekten aus?

Wenn wie von Abbildung 6 auf Abbildung 7 als erstes die Referenz vom Krümel auf das Krümelsymbol gelöscht wird, dann wäre es nicht mehr möglich, einen Methodenaufruf *KruemelSymbolEntfernen* an das Objekt kSymbol zu schicken. Eine der beiden Voraussetzung für eine funktionierende Objektkommunikation (Eine Referenz vom Sender zu Empfänger muss existieren, vgl. Kapitel 11.2) ist nicht mehr erfüllt.

Hinweis:
Führt man für die Situation in Abbildung 7 den Methodenaufruf kSymbol.*KruemelSymbolEntfernen* aus, so erhält man die Fehlermeldung "NullPointer Exception". Frei übersetzt bedeutet dies: Fehler (exception) wegen einer Referenz (pointer) die nichts (null) referenziert.

Somit ergibt sich als einzig mögliche Reihenfolge:

1) Löschen der Referenz vom Backend auf das Krümelsymbol über den Methodenaufruf kSymbol.*KruemelSymbolEntfernen*
2) Löschen der Referenz vom Krümel auf sein Symbol über die Zuweisung kSymbol = null
3) Löschen der Referenz von der Zelle auf den Krümel über die Zuweisung kruemel = null

Aufgabe 13.9

Im Quelltext welcher Klasse müssen folgende Ergänzungen vorgenommen werden?

a) der Methodenaufruf kSymbol.KruemelSymbolEntfernen entsprechend Punkt 1)
b) die Zuweisung kSymbol = null entsprechend Punkt 2)
c) die Zuweisung kruemel = null entsprechend Punkt 3)

Die Objektkommunikation zum Löschen des Krümels nach dem Aufruf der Methode *IstMauerSetzen(true)* ist in Abbildung 8 dargestellt.

© brichzin@tcs.ifi.lmu.de, 2007

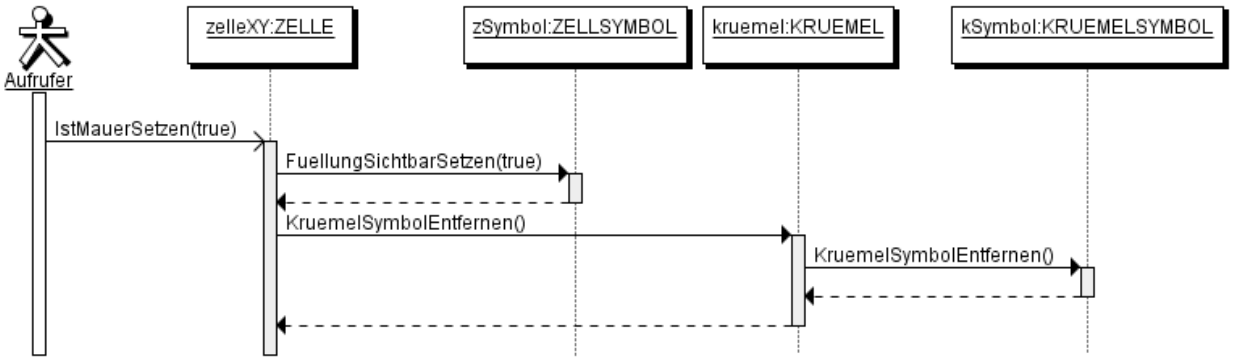


Abbildung 8: Objektkommunikation, die dafür sorgt, dass beim Erstellen einer Mauer der Krümel auf der Zelle entfernt wird. Nicht sichtbar sind im Sequenzdiagramm die Zuweisungen, die für das Löschen der Referenzen sorgen.

Hinweise:

- Die Zuweisungen auf null sind im Sequenzdiagramm nicht enthalten, da dieser Diagrammtyp nur Methodenaufrufe und keine Zuweisungen visualisiert.
- Zu Beginn wird die Methode *IstMauerSetzen* der betreffenden Zelle aufgerufen. Dadurch wird die Zelle zum aktiven Objekt. Da es jedoch keine Referenz von der Zelle zum Krümelsymbol gibt, muss der Aufruf der Methode *KruemelSymbolEntfernen* des Krümelsymbolobjekts über die "Zwischenstation" Krümel kommuniziert werden. Dazu muss in der Klasse KRUEMEL eine neue Methode geschrieben werden, deren einzige Aufgabe ist, die Methode *KruemelSymbolEntfernen* aufzurufen. Da auch diese neue Methode die Aufgabe hat das Krümelsymbol zu entfernen, wird sie ebenfalls ebenfalls *KruemelSymbolEntfernen* genannt. Die Methode *KruemelSymbolEntfernen* gibt es somit in der Klasse KRUEMEL und KRUEMELSYMBOL, beide mit dem gleichen Ziel, aber mit unterschiedlichen Methodenrümpfen.

Aufgabe 13.10
Setze die besprochenen Änderungen in den Klassen ZELLE und KRUEMEL um. Teste danach, indem du ein Objekt der Klasse Labyrinth erzeugst und dann die Methode *GaengeErstellen* aufrufst. Danach darf auf keiner Mauer ein Krümel liegen (z.B. Abbildung 9).
Bei Bedarf findest du Hilfe im Anhang.
Vergiss nicht, alle Änderungen (über Javadoc-Kommentare) zu dokumentieren.

© brichzin@tcs.ifi.lmu.de, 2007

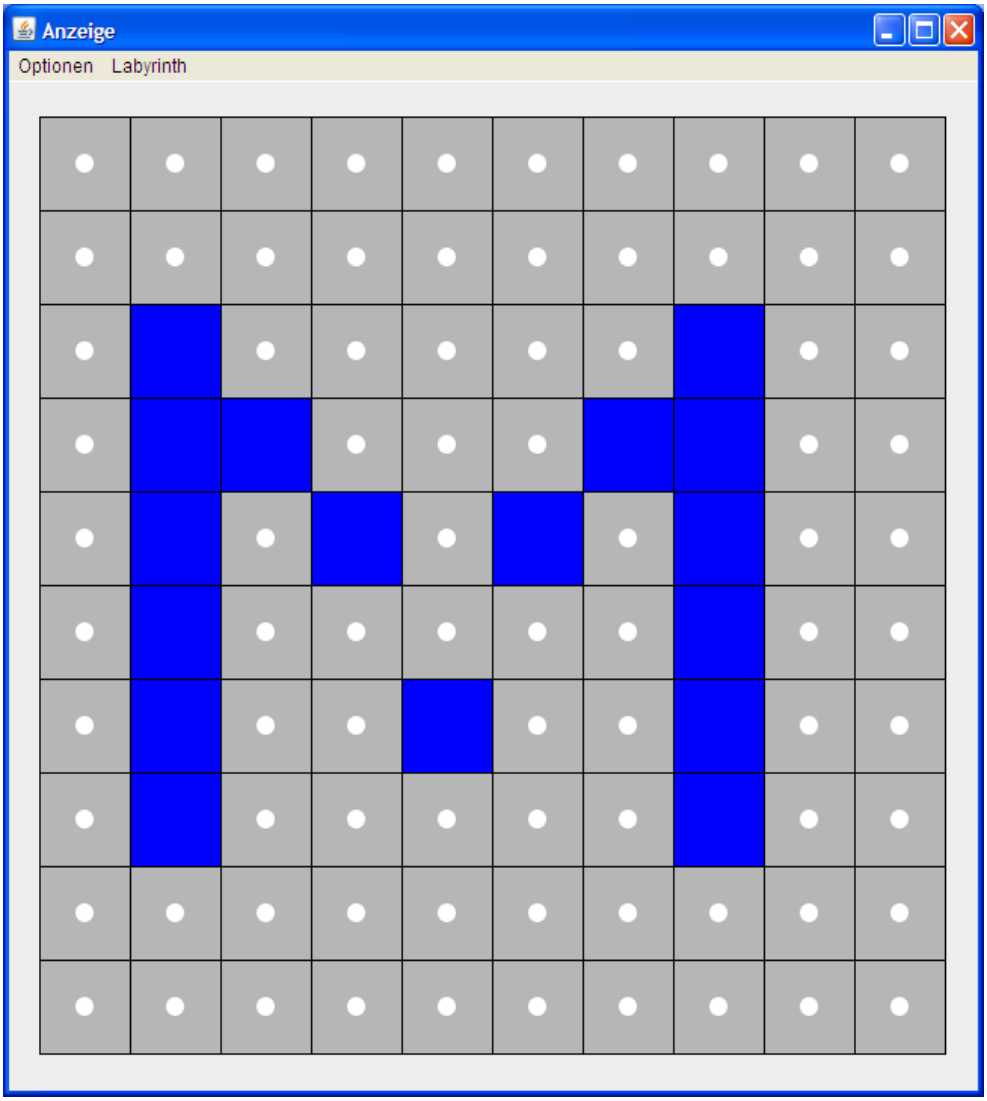


Abbildung 9: Labyrinth mit Mauern und Krümeln

13.4 Zusammenfassung

Aufgabe 13.11
In diesem Kapitel wurde neu erklärt, wie man Referenzen löschen kann und erläutert, dass dadurch mit Hilfe der garbage collection Objekte gelöscht werden. Fasse dies zusammen.

Zum wiederholten Mal wird die Objektkommunikation vertieft. U. a. folgende Punkte sollten dir vertraut sein:
- Zwei Voraussetzungen für eine funktionierende Objektkommunikationen
- Erklärung einer Objektkommunikation über mehrere "Stationenen" an einem selbst gewählten Beispiel
- Veranschaulichung der Objektkommunikation durch ein Sequenzdiagramm

Aufgabe 13.12
Lies den Anhang B.

© brichzin@tcs.ifi.lmu.de, 2007

Anhang A: Tipps zu den Aufgaben

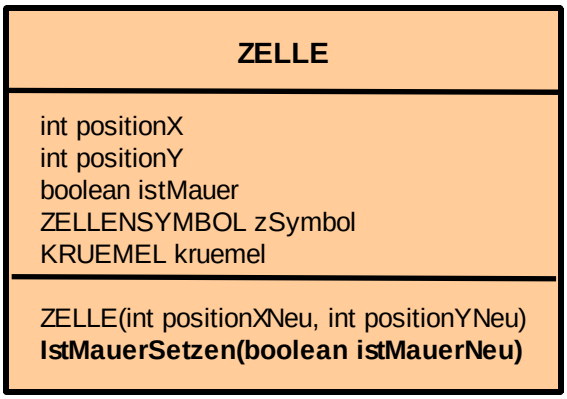
Tipps zu Aufgabe 13.1:
Folgende Beschreibungen sollen dir helfen wichtige Attribute und Methoden der Klasse KRUEMEL herauszufinden. Es ist kein Problem, wenn die Klasse im ersten Anlauf noch nicht perfekt ist, du kannst jederzeit nachträglich optimieren und ergänzen:
- Wo sind Krümel zu finden?
- Es gibt unterschiedliche Krümelvarianten. Wodurch unterscheiden sie sich?
- Wie kann man Krümel im Anzeigefenster sichtbar machen?
- Wie wird die Situation im Spiel besser modelliert?
  "Das Labyrinth hat viele Krümel." oder "Jede Zelle hat einen oder keinen Krümel."

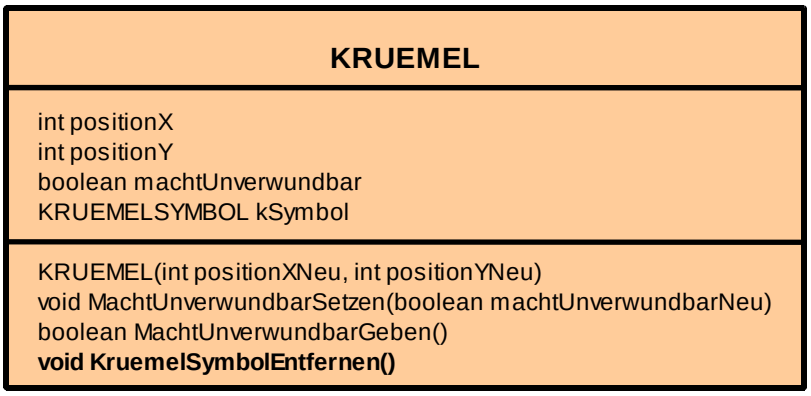
Tipps zu Aufgabe 13.4:
Folgende Anleitung sollen dir bei der Entwicklung helfen
- Setze in Java in der Klasse KRUEMEL zunächst nur die Attribute und den Konstruktor um.
- Überlege dir beim Konstruktor der Klasse KRUEMEL sinnvolle Eingabewerte!. Die Eingabewerte sind nötig, um das Krümelsymbol richtig zu platzieren.
- Erzeuge ein Objekt der Klasse KRUEMEL. Öffne den passenden Objektinspektor und überprüfe die Attributwerte. Kontrolliere, ob das Symbol des Krümels korrekt in der Anzeige erscheint!
- Setze nun die Methoden um.
- Überprüfe wiederum an Hand eines konkreten Objekts mit Hilfe des Objektinspektors und der Anzeige, ob Methodenaufrufe die gewünschten „Ergebnisse“ liefern.
- Setze nun die Beziehungen aus Aufgabe 13.1d) um, die die KRUEMEL ins Spiel integrieren.

© brichzin@tcs.ifi.lmu.de, 2007

Tipps zu Aufgabe 13.10:
Folgende Anleitung sollen dir bei der Entwicklung helfen
Überblick

a) Geändert werden muss die Methode *IstMauerSetzen* der Klasse ZELLE.
b) Ergänzt werden muss die Methode *KruemelSymbolEntfernen* in der Klasse KRUEMEL (siehe Fettdruck in den Klassendiagrammen).





Erläuterungen zu

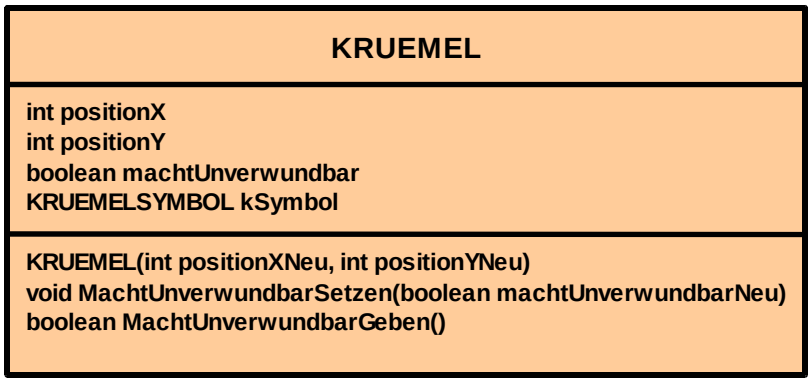
b) In der Methode *KruemelSymbolEntfernen* in der Klasse KRUEMEL muss die Referenz zwischen dem Backend und dem Krümelsymbol gelöscht werden (Punkt 1 auf Seite 6)
und die Referenz zwischen dem Krümel und seinem Symbol (Punkt 2 auf Seite 6)

a) In der Methode *IstMauerSetzen* der Klasse ZELLE muss dafür gesorgt werden, dass falls bei einem Eingabewert `true`, die Methode *KruemelSymbolEntfernen* des Krümels aufgerufen werden muss. Jedoch muss durch eine bedingte Anweisung garantiert werden, dass der Methodenaufruf nur erfolgt, wenn es noch einen Krümel gibt.
Was würde passieren, wenn eine Zelle ohne einen Krümel den Aufruf krumel.*KruemelSymbolEntfernen* durchführen würde.

© brichzin@tcs.ifi.lmu.de, 2007

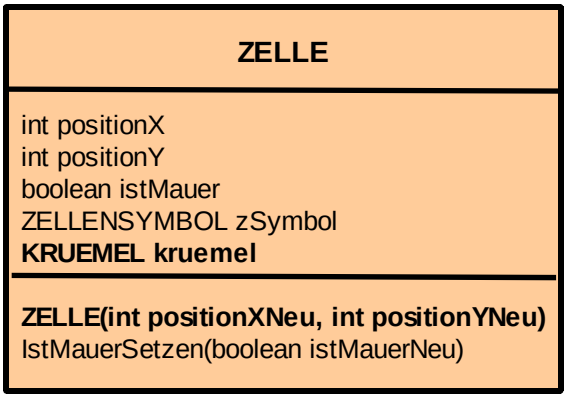
Noch mehr Tipps zu Aufgabe 13.1:

Mögliche Attribute und Methoden der Klasse KRUEMEL sind in dem erweiterten Klassendiagramm abgebildet:



Jeder Krümel wird dargestellt durch ein Krümelsymbol. Jeder Krümel ist genau einer Zelle zugeordnet. Nicht auf jeder Zelle muss ein Krümel sein. Somit hat eine Zelle einen oder keinen Krümel:

Die Beziehungen müssen in Form von Referenzattributen umgesetzt werden:
In der Klasse KRUEMEL gibt es ein Referenzattribut vom Typ KRUEMELSYMBOL, in der Klasse ZELLE vom Typ KRUEMEL.



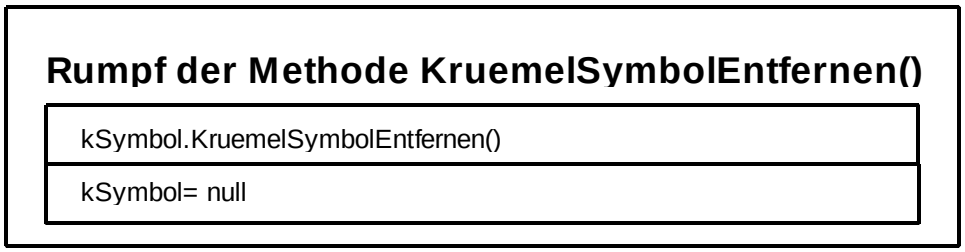
Noch mehr Tipps zu Aufgabe 13.4:

Denke daran, dass in der Klasse ZELLE nicht nur das Referenzattribut ergänzt , sondern auch der Konstruktor entsprechend angepasst werden muss!
(Vergleich z.B. die Vorgehensweise im Konstruktor der Klasse MAMPFI hinsichtlich des MAMPFISYMBOLS)

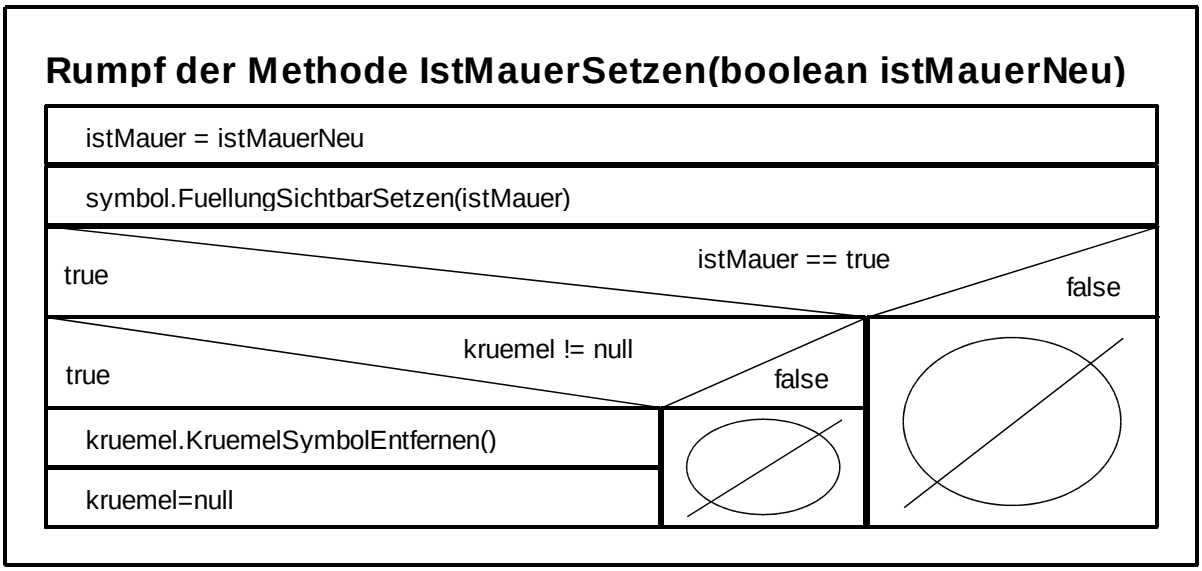
© brichzin@tcs.ifi.lmu.de, 2007

Noch mehr Tipps zu Aufgabe 13.10:

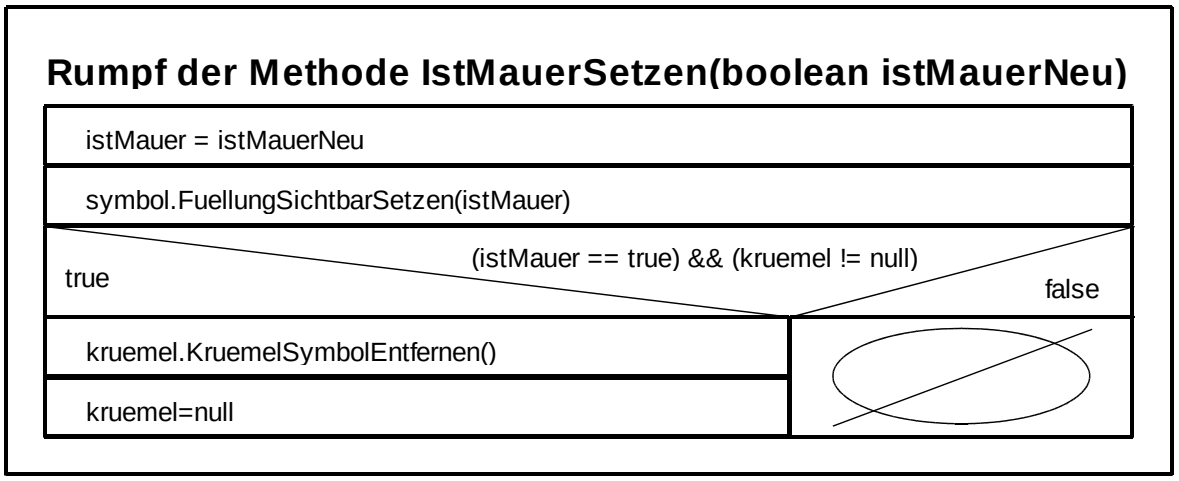
Struktogramm der in der Klasse KRUEMEL ergänzten Methode *KruemelSymbolEntfernen*



Struktogramm der in der Klasse ZELLE veränderten Methode *IstMauerSetzen*



Eine kürzere Formulierung der Methode *IstMauerSetzen* ergibt sich mit dem UND Operator "&&" .

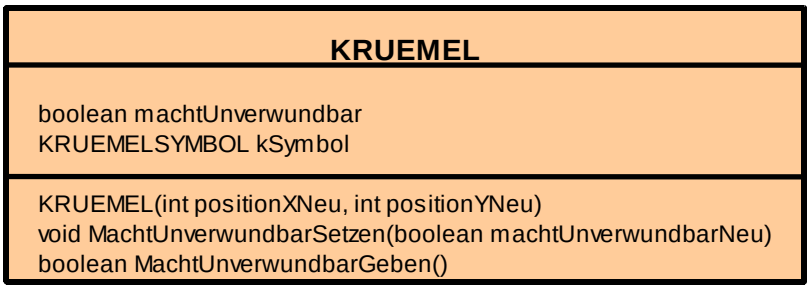


© brichzin@tcs.ifi.lmu.de, 2007

Anhang B: Alternative Umsetzung der Klasse KRUEMEL (und ZELLE)

Ist das Speichern der Position eines Krümels in entsprechenden Attributen der Klasse KRUEMEL nötig? Ist nicht die Position des Krümels durch die Position der Zelle festgelegt? Warum werden die Informationen in der Zelle und im Krümelsymbol redundant gespeichert? Dort gibt es ebenfalls die Attribute positionX und positionY.

Aus Modellierungssicht ist die Position eine wichtige Eigenschaft des Krümels. Aus diesem Grund ist die in diesem Kapitel vorgestellte Lösung sicherlich vertretbar. Der Vorwurf einer doppelten Speicherung der Daten ist aber auch gerechtfertigt. Insbesondere wird die Krümelposition ein einziges Mal festgelegt (beim Erzeugen), und dann nie wieder geändert. In der Klasse KRUEMEL werden die Positionskoordinaten nur dazu verwendet, im Konstruktor das Krümelsymbol richtig zu positionieren. Aus diesem Grunde ist es denkbar, in der Klasse KRUEMEL die Attribute positionX und positionY wegzulassen und im Konstruktor die Eingabewerte an das Kruemelsymbol weiterzureichen (Siehe Klassendiagramm bzw. Quelltext des Konstruktors)




```
KRUEMEL(int positionXNeu, int positionYNeu)
{
    machtUnverwundbar = false;

    kSymbol = new KRUEMELSYMBOL(positionXNeu, positionYNeu);
    kSymbol.RadiusSetzen(6);   // Groesse angepasst auf 10x10 Labyrinth
}
```

Die eben durchgeführten Überlegungen lassen sich auch auf die Klasse ZELLE übertragen. Auch hier kann man auf die Positionsattribute verzichten. Wiederum sorgt der Konstruktor für ein einmaliges Setzen der richtigen Positionen.

```
// Konstruktor für Objekte der Klasse ZELLE
ZELLE(int positionXNeu, int positionYNeu)
{
    istMauer = false;

    zSymbol = new ZELLENSYMBOL(positionXNeu, positionYNeu);
    zSymbol.FuellFarbeSetzen("blau");
    zSymbol.FuellungSichtbarSetzen(false);

     kruemel = new KRUEMEL(positionXNeu, positionYNeu);
}
```

Aufgabe B. 1
Setze diese alternative Lösung um. Teste und dokumentiere sie.

© brichzin@tcs.ifi.lmu.de, 2007