Kapitel 9 Krümel im Labyrinth

Lernziele:

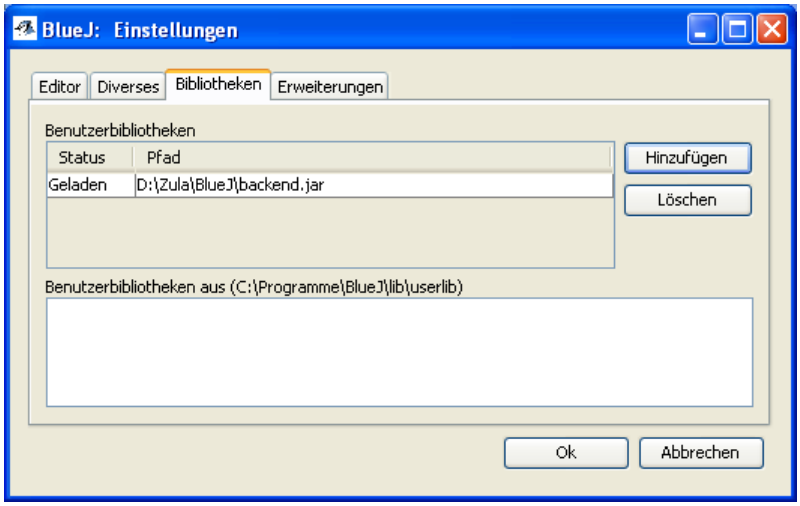
In diesem Kapitel kommen keine neuen Inhalte vor, sondern es werden alte erneut angewendet.

9.0 Vorbereitungen

a) Wieder ein neues backend (alle)

Da wir das Spiel bzw. das Konzept parallel zur Erprobung entwickeln, kommt es immer wieder zu Veränderungen im Backend. Aus diesem Grund sollt Ihr die zentral im Marktplatz abgelegte Bibliothek einbinden. Wir können dann dort Änderungen vornehmen, die für Euch dann automatisch für Euch verfügbar sind.
Bevor ihr das neue Backend ladet, muss das alte gelöscht werden.

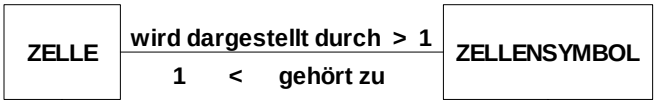
1. Öffnet in der BlueJ-Menüleiste das Menü *Werkzeuge* > *Einstellung*
2. Klickt auf den Reiter „Bibliotheken“ und dann auf den Button „Löschen“



3. Wählt nun über „Hinzufügen“ die Datei KM_backend.jar **direkt vom Marktplatz** aus und **nicht** aus dem Home-Verzeichnis. Schliesst das Menü mit „OK“
4. Jetzt BlueJ neu starten, damit die Klassensammlung von BlueJ geladen wird

b) Änderung des Konzepts der SYMBOL-Klassen (alle)

Bisher haben sich die Objekte der SYMBOL-Klassen automatisch an der passenden Position selbst gezeichnet. Dies war möglich, da die Beziehung beispielsweise zwischen einer Zelle und ihrem Symbol bidirektional war, das Zellensymbol konnte sich



über die Methode PositionXGeben() die Information für die Position holen.

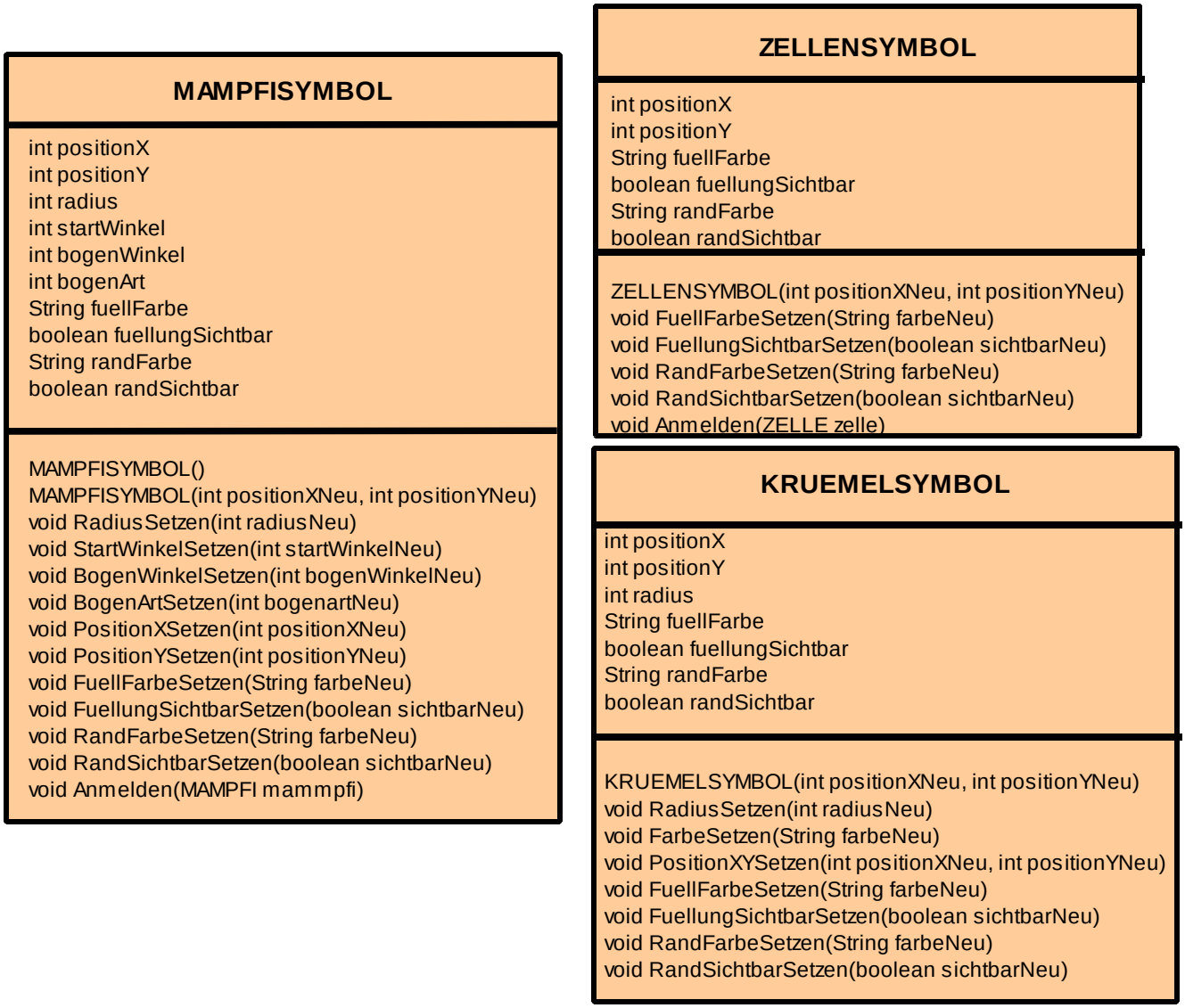
Neuerdings muss jede Information zur Darstellung eines Objekts unseres Spiels Krümel & Monster, also nicht nur die Farbe etc. an sein Symbol durch einen Methodenaufruf weitergeben.
Beispielsweise teilt eine Zelle über den Methodenaufruf

```
form = new ZELLENSYMBOL(5, 4);
```

ihrer Darstellung mit, dass es an der Position (5, 4) gezeichnet werden soll. II[1].

Neuerdings muss für besondere Leistungen des Backends, wie z. B. das Setzen einer Mauer per Mausklick, das entsprechende Objekt an dem Backend angemeldet werden. Dafür wird in der passenden SYMBOL-Klasse eine Methode *anmelden* angeboten.

Im folgenden sind die Klassendiagramme der neuen SYMBOl-Klassen abgebildet:



c) Die SYMBOL-Klassen ersetzen **(nur für diejenigen, welche die eigenen Klassen weiter verwenden wollen)**

Wegen die Änderungen im Backend müssen auch die Klassen MAMPFISYMBOL und ZELLENSYMBOL erneuert werden.
Löscht dazu die alten Klassen und fügt die neuen über das Menü *Bearbeiten> Klasse aus Datei hinzufügen* ein.
Für das heutige Kapitel musst Du zusätzlich noch die Klasse KRUEMELSYMBOL einfügen.

1 Der Konstruktor der Klasse ZELLENSYMBOL hat folgenden Methodenkopf:
ZELLENSYMBOL(int positionXNeu, int positionYNeu)

d) Änderungen in ZELLE **(nur für diejenigen, welche die eigenen Klassen weiter verwenden wollen)**

Die unter 2. beschriebenen Änderungen haben folgende Änderungen zur Folge:

- Der Kontruktor der Klasse ZELLENSYMBOL hat sich geändert. Deshalb muss die Zeile
  `form = new ZELLENSYMBOL(this);`
  geändert werden.
  zusätzlich ist direkt im Anschluss ein Methodenaufruf nötig, um blau als Farbe für die Mauern zu setzen. In der folgenden Abbildung sind die Änderungen farbig hervorgehoben.

```
ZELLE(int positionXNeu, int positionYNeu)
{
    positionX = positionXNeu;
    positionY = positionYNeu;
    istMauer = false;

    form = new ZELLENSYMBOL(positionX, positionY);
    form.FuellFarbeSetzen("blau");
```

- Die Methode *IstMauerSetzen* muss wie folgt geändert werden

```
void IstMauerSetzen(boolean istMauerNeu)
{
    istMauer = istMauerNeu;
    form.FuellungSichtbarSetzen(istMauerNeu);

}
```

  (Ist der Eingabewert dieser Methode true, wird nicht nur der Wert des Attributs istMauer passend gesetzt, sondern auch über den Methodenaufruf form.AusfuellenSetzen(true);
  Die Füllfarbe blau für Mauern wurde schon beim Konstruktor festgelegt.
- Jede Zelle muss am Backend angemeldet werden. Dies geschieht durch folgende Zeile im Konstruktor:
  `form.anmelden(this);`

e) Änderungen in MAMPFI **(nur für diejenigen, welche die eigenen Klassen weiter verwenden wollen)**

- Der Kontruktor der Klasse MAMPFISYMBOL hat sich geändert. Deshalb muss die Zeile
  `form = new MAMPFISYMBOL(this);`
  wie folgt verändert werden:
  `form = new MAMPFISYMBOL();`
- Auch Mampfi muss nun immer seine Änderungen dem Symbol mitteilen. Dies hat er bisher nur getan als er die Blickrichtung oder seine Farbe verändert hat. Nun

muss er auch, wenn er sich bewegt, die x- und y-Position dem Mampfisymbol übergeben. Deshalb muss jeweils am Ende der Methoden *NachNordenGehen und NachSuedenGehen* der Methodenaufruf *form.PositionYSetzen* ergänzt werden,
am Ende der Methoden *NachWestenGehen und NachOstenGehen* der Methodenaufruf *form.PositionXSetzen* .

- Alle Methoden *FarbeSetzen* müssen in *FuellFarbeSetzen* umbenannt werden. (Arbeite mit Suchen und Ersetzen bzw. lass dir vom Compiler die Fehler anzeigen)

.
Gibt es noch andere Methoden in denen Du die x- oder y-Position veränderst. Musst Du entsprechendende Änderungen durchführen.

9.1 Die Klasse KRUEMEL

Aufgabe 9.1
Entwirf eine Klasse KRUEMEL als Klassendiagramm auf Papier mit Bleistift. Welche weitere Klasse ist zur Darstellung von Krümeln nötig? Wie äußert sich dies im (erweiterten) Klassendiagramm?

Aufgabe 9.2
Setze den Entwurf aus Aufgabe 9.1. in Java um. Teste nach jedem Teilschritt.

Aufgabe 9.3
Fülle das Labyrinth mit Krümeln. In diesem ersten Entwurf sollen Mauern noch nicht vorkommen. Es ist im ersten Entwurf auch noch einfacher sich auf einer Sorte von Krümeln zu beschränken.

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 Folgenden verschiedene Fragen und Tipps, die dir helfen sollen, die nächsten Schritte zu finden.

Tipps zu Aufgabe 9.1:
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 ja jederzeit nachträglich optimieren und ergänzen:

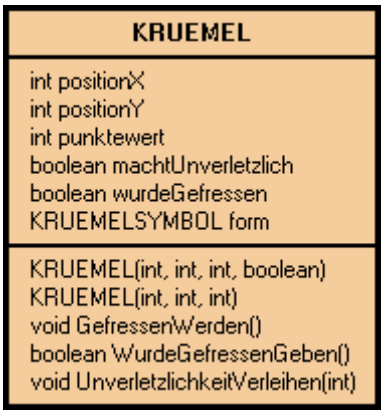
- Wo sind Krümel zu finden?
- Welchen Vorteil hat Mampfi (bzw. der Spieler der Mampfi leitet), wenn ein Krümel gefressen wird?
- Sind alle Krümel gleich?
- Wie kann man Krümel im Labyrinth sichtbar machen?
- Was passiert, aus unserer Sicht als Spieleentwickler, mit dem Krümel, wenn er gefressen wird.

Tipps zu Aufgabe 9.2:
Folgende Anleitung sollen dir bei der Entwicklung helfen

- Setze in der Java-Klasse zunächst nur die Attribute und den Konstruktor um.
- Überlege dir beim Konstruktor sinnvolle Eingabewerte!. Die Eingabewerte werden später helfen, über Wiederholungsanweisungen das Labyrinth zu füllen (--> Aufgabe 9.3)
- 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 widerum an Hand eines konkreten Objekts mit Hilfe des Objektinspektors und der Anzeige, ob Methodenaufrufe die gewünschte „Ergebnisse“ liefern.

Tipp zu Aufgabe 9.3

- Aus Modellierungssicht ist die Beziehung „Ein Objekt der Klasse ZELLE enthält ein (oder kein) Objekt der Klasse KRUEMEL“ sehr sinnvoll.



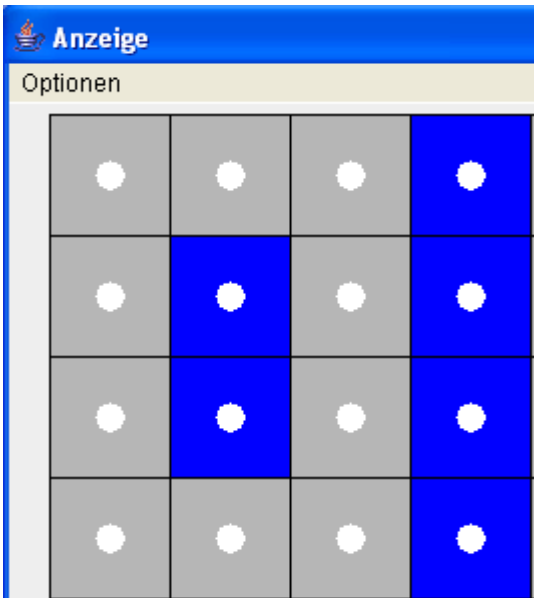
Noch mehr Tipps zu Aufgabe 9.1:
Ein mögliches Ergebnis zu dieser Aufgabe ist der Abbildung rechts zu entnehmen. Beachte dabei, dass das Referenzattribut form eine Beziehung zwischen den Klassen KRUEMEL und KRUEMELSYMBOL umsetzt.
Hinweis:
Es gibt verschiedene richtige Lösungen! Welche Methoden bzw. Eingangsparameter bei Konstruktoren wichtig sind, wird teilweise erst im späteren Verlauf deutlich. Einer der beiden Konstruktoren im gezeigten Klassendiagramm hat den Methodenkopf
KRUEMEL(int positionXNeu, int positionYNeu, int punktewertNeu, boolean machtUnverletzlichNeu)

Noch mehr Tipps zu Aufgabe 9.3:
Es gibt zwei mögliche Lösungsansätze:
Einträge sind entweder (hauptsächlich) im Konstruktor der Klasse LABYRINTH nötig
oder (hauptsächlich) im Konstruktor der Klasse ZELLE.

9.2 Miteinander „reden“ ist wichtig

Aufgabe 9.4
- Erzeuge ein (mit Krümeln gefülltes) Objekt der Klasse Labyrinth. Klicke auf eine beliebige Zelle in der Anzeige oder rufe auf andere Art die Methode IstMauerSetzen mit dem Eingabewert true auf.
  Womit kannst du nicht zufrieden sein

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



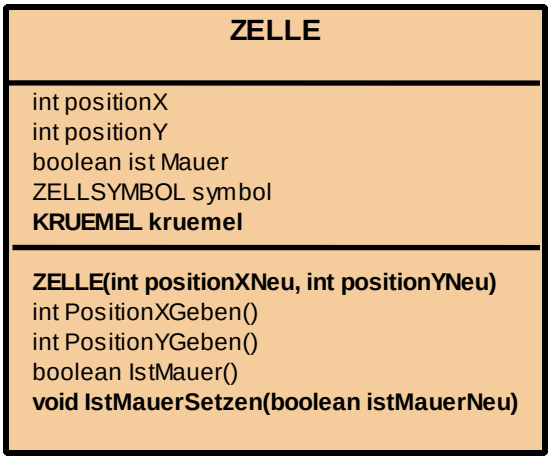
Aufgabe 9.5
Überlege welche Objekte für die eben angesprochene Verbesserung beteiligt werden müssen und wie sie miteinander kommunizieren müssen.

Aufgabe 9.6
Ändere deine betroffenen Java-Klassen so, dass nie auf einer Mauer ein Krümel erscheint.

Tipps zu Aufgabe 9.5 / 9.6

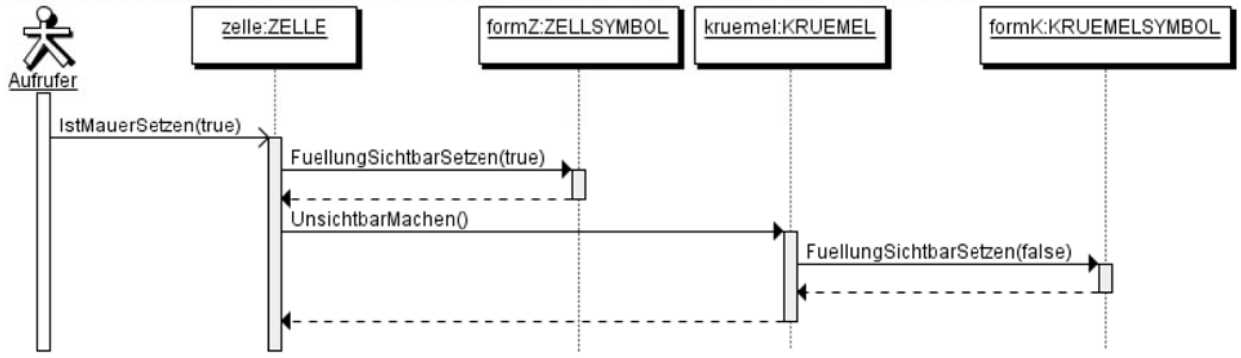
Es gibt zwei Varianten die Krümel in das Spiel zu integrieren. Entweder über die Klasse LABYRINTH oder über die Klasse ZELLE. Hier wird die Variante über die Klasse ZELLE skizziert:

Die Beziehung „Ein Objekt der Klasse ZELLE enthält ein Objekt der Klasse KRUEMEL“ wird über ein Referenzattribut in der Klasse ZELLE vom Typ KRUEMEL realisiert (siehe Markierung im Klassendiagramm rechts). Dies hat auch Auswirkungen auf den Konstruktor der Klasse ZELLE, da dort der Methodenaufruf zur Erzeugung des Krümelobjekts ergänzt werden muss (Vergleiche die Umsetzung von Objektbeziehungen in den früheren Kapiteln z. B. Kapitel 3).



Tipps zu Aufgabe 9.5

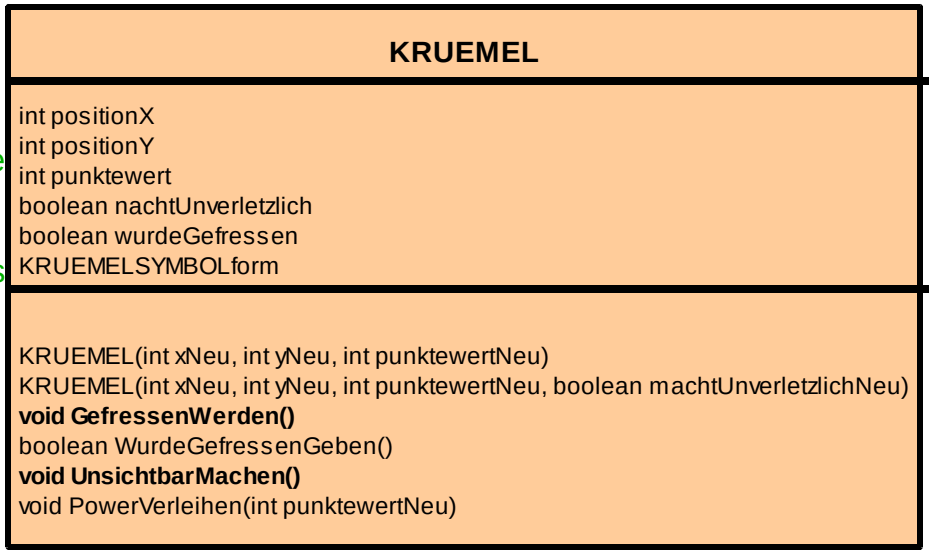
Folgendes Sequenzdiagramm zeigt die Kommunikation der Objekte:



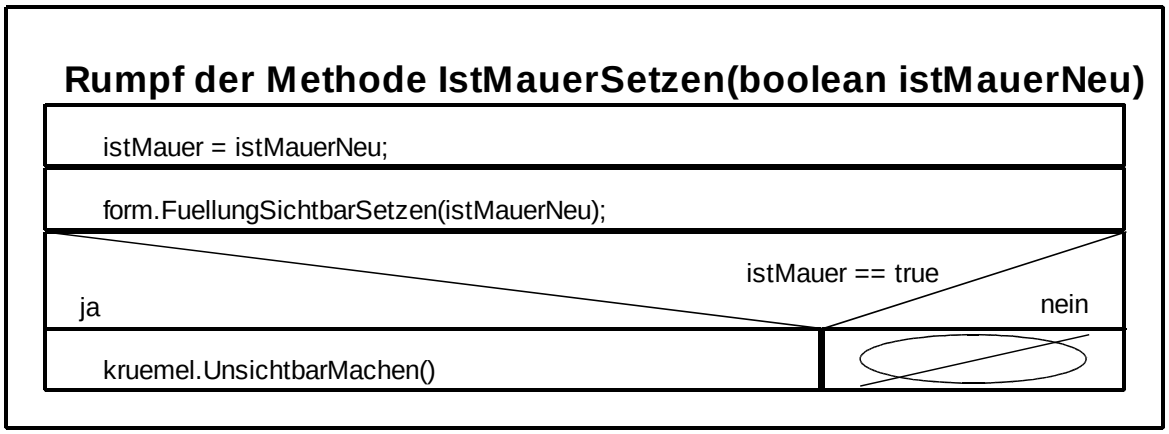
Beachte, dass kein Zellen-Objekt ein Objekt der Klasse KRUEMELSYMBOL kennt, sondern jeweils nur ein Objekt der Klasse KRUEMEL. Um das Verschwinden des Krümelsymbols in der Anzeige zu erreichen, muss auch das passende Objekt der Klasse KRUEMELSYMBOL in die Kommunikation einbezogen werden. Dies ist indirekt über das passende Krümelobjekt möglich: Wird in der Klasse KRUEMEL eine neue Methode, beispielsweise mit dem Namen *UnsichtbarMachen*, hinzugefügt, kann. innerhalb dieser Methode die Methode *FuellungSichtbarSetzen* des zugehörigen Krümelsymbols aufgerufen, weil jeder kruemel seine form kennt.

Tipps zu Aufgabe 9.6

a)
Ergänzt werden muss, wie gerade beschrieben, in der Klasse KRUEMEL eine Methode *unsichtbarMachen*. Diese sorgt dafür, dass der Krümel in der Anzeige nicht mehr sichtbar ist. Diese Methode soll auch verwendet werden, um die Methode *GefressenWerden* zu optimieren.



b)
Ergänzt werden muss dann in der Methode *IstMauerSetzen* (der Klasse ZELLE) ein Methodenaufruf, der das Verschwinden des Krümels bewirkt, falls der Eingabewert true ist. Diese Methode verwendet die Methode *unsichtbarMachen* der Klasse KRUEMEL. Folgendes Struktogramm kann eine Hilfe für die Umsetzung der Methode *IstMauerSetzen* sein:

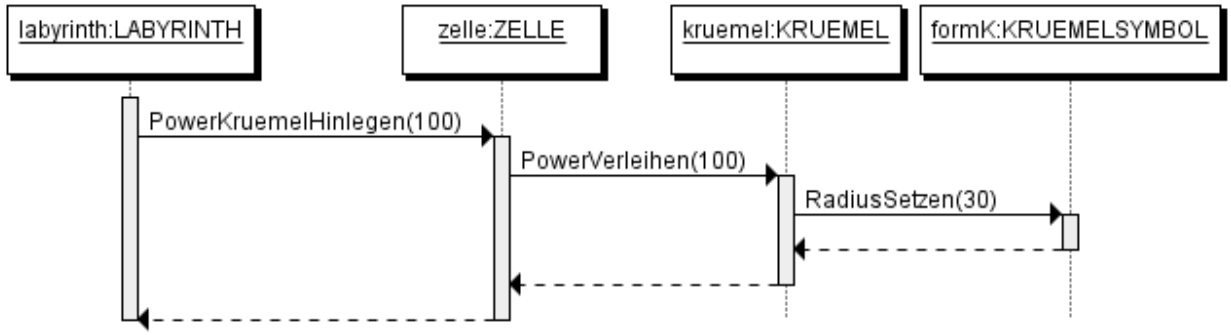


9.3 Wo sind die Krümel die unverletzlich machen?

Bisher gibt es noch keine Krümel die unverletzlich machen. Vier Stück soll es geben. Je nachdem wie du das Spiel gestalten möchtest, können sie sich immer am gleichen Platz befinden oder in unterschiedlichen Level auch an unterschiedlichen Stellen zu finden sein.

Aufgabe 9.7
Überlege, wo es sinnvoll wäre den Ort der besonderen Krümel zu speichern. Welche Klassen müssen verändert werden, um diese Information zu verarbeiten? An welchen Stellen genau, müssen Veränderungen vorgenommen werden?

Mit Sicherheit ist die Klasse LABYRINTH richtig, die Information über den Ort der besonderen Krümel zu speichern. In einer einfachen Variante könnte man vier Koordinaten festlegen (z.B. (1/1), (1/hoehe-1), (breite-1/1) und (breite-1/hoehe-1)) und im Konstruktor der Klasse LABYRINTH vier Methodenaufrufe ergänzen. Jeder der Methodenaufrufe muss dafür sorgen, dass der „normale“ Krümel zum Powerkrümel wird und dies entsprechend auch in der Anzeige dargestellt wird. Eine mögliche Objektkommunikation dazu wird in folgendem Seqeunzdiagramm dargestellt:



Hinweis: Beachte, dass das Labyrinth das Objekt der Klasse KRUEMEL nicht kennt, sondern nur Objekte der Klasse ZELLE.
Ähnlich wie in Kapitel 9.2 muss eine Methode ergänzt werden, das eine Nachricht „weiterleiten“.
So muss in der Klasse ZELLE eine Methode beispielsweise mit dem Namen *PowerKruemelHinlegen* ergänzt werden. Diese ruft dann die Methode *PowerVerleihen* des ihr bekannten Krümelobjekts mit dem Eingabewert 100 auf. Der Eingabewert 100 legt fest, wie viele Punkte für den verspeisten Krümel gutgeschrieben werden.

ZELLE
int positionX int positionY boolean ist Mauer ZELLSYMBOL symbol KRUEMEL kruemel
ZELLE(int positionXNeu, int positionYNeu) int PositionXGeben() int PositionYGeben() boolean IstMauer() void IstMauerSetzen(boolean istMauerNeu) **void PowerKruemelHinlegen(int punktewertNeu)**

Aufgabe 9.8
Beim Erzeugen eines Labyrinths sollen vier Krümel enthalten sind, die unverletzlich machen. Ergänze die betroffenen Klassen entsprechend den obigen Überlegungen.