Kapitel 11 Objektkommunikation mit Antworten

Lernziele:
Methoden mit Rückgabewerten, Wiederholung Objektkommunikation

11.1 Grenzen des Labyrinths

In der Klasse MAMPFI gibt es die Methode *NachOstenGehen*. In Abbildung 1 ist das Struktogramm des Methodenrumpfs zu sehen. Die Methode ist noch nicht optimal formuliert, da sie nur für Spielflächen der Größe 10x10 korrekt arbeitet. Hätte beispielsweise ein Labyrinth die Breite 20, so wäre es für Mampfi unmöglich in die rechte Hälfte zu gelangen.

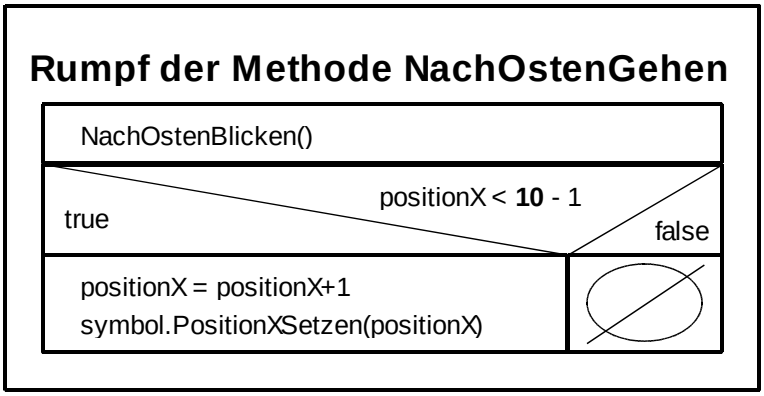

Abbildung 1: Struktogramm der Methode *NachOstenGehen*

Aufgabe 11.1
Welche Stelle im Methodenrumpf ist für die Einschränkung verantwortlich?
Wie kann man die Einschränkung aufheben?
Welche Information benötigt Mampfi dazu?
Von wem kann er diese Information erhalten?

Mampfi würde sich korrekt in allen Labyrinthen bewegen, wenn in der Bedingung der bedingten Anweisung anstatt des festen Werts 10 immer allgemein die Breite des Labyrinths stehen würde. Die Information, wie breit ein Labyrinth ist, kennt nur das Objekt der Klasse LABYRINTH, in dem sich Mampfi bewegt. Mampfi müsste somit beim Labyrinth nachfragen.

Aufgabe 11.2
Was ist Voraussetzung, dass Mampfi mit dem Labyrinth kommunizieren kann?

Ein Nachrichtenaustausch zwischen Objekten kann nur dann stattfinden, wenn das sendende Objekt die Nachricht dem empfangenden Objekt zustellen kann. Wie bei einer Postzustellung muss das Sendeobjekt das Empfangsobjekt kennen, es muss sich ähnlich einem Adressbucheintrag eine Referenz auf das Empfangsobjekt merken.
Im aktuellen Fall benötigt Mampfi eine Referenz auf das Labyrinth, um dorthin die Frage nach der Breite stellen zu können. Ein Referenzattribut setzt die Beziehung zwischen den Objekten um.

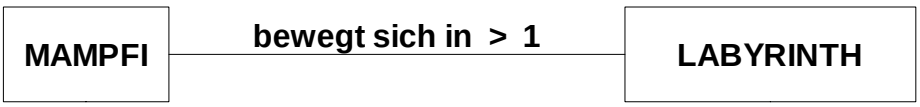

Abbildung 2: Beziehung zwischen MAMPFI und LABYRINTH

Aufgabe 11.3
Wie muss das erweiterte Klassendiagramm der Klasse MAMPFI ergänzt werden, um die Beziehung aus Abbildung 2 umzusetzen? Wie kann Mampfi eine Referenz auf das

Labyrinth erhalten, wenn das Objekt der Klasse Labyrinth schon vor Mampfi erzeugt wurde?

Die Beziehung aus Abbildung 2 wird durch ein Referenzattribut vom Typ LABYRINTH umgesetzt. Damit der Wert des Referenzattributs bei der Erzeugung von Mampfi richtig gesetzt werden kann, muss dem Konstruktor die Referenz auf ein bereits erzeugtes Labyrinth übergeben werden. (Abbildung 3)

Aufgabe 11.4
Setze die besprochenen Ergänzungen in deinem BlueJ-Projekt um.
Beachte beim Testen, dass du zuerst ein Objekt der Klasse Labyrinth erzeugen musst, bevor du Mampfi erzeugst. Überprüfe nach dem Erzeugen der beiden Objekte, ob du über den Objektinspektor von Mampfi auf den Objektinspektor des Labyrinths zugreifen kannst.

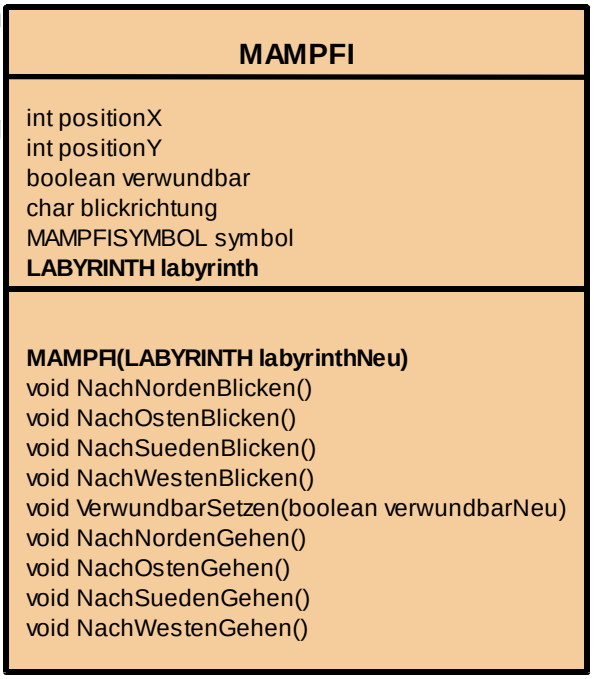

Abbildung 3: Ergänzungen in der Klasse MAMPFI: Referenzattribut labyrinth und ein Eingangsparameter im Konstruktor

Hinweis:
Warum ist es sinnvoll das Labyrinth zuerst zu erzeugen und die Referenz danach als Eingabewert bei der Erzeugung von Mampfi zu übergeben?

Eine alternative Lösung zu der oben beschriebenen wäre, das Labyrinth beim Ausführen des Konstruktors *MAMPFI* zu erzeugen, ähnlich zu den Symbol-Objekten (Quelltext in Abbildung 4):

```
MAMPFI()
    {
        positionX = 3;
        positionY = 2;
        verwundbar = true;
        blickrichtung = 'N';

        labyrinth = new LABYRINTH();

        symbol = new MAMPFISYMBOL();
        symbol.RadiusSetzen(25);
       ...
    }
```

Abbildung 4: Alternative Formulierung des Konstruktors der Klasse MAMPFI - Das Labyrinth wird hier direkt erzeugt und nicht als Eingabewert übergeben.

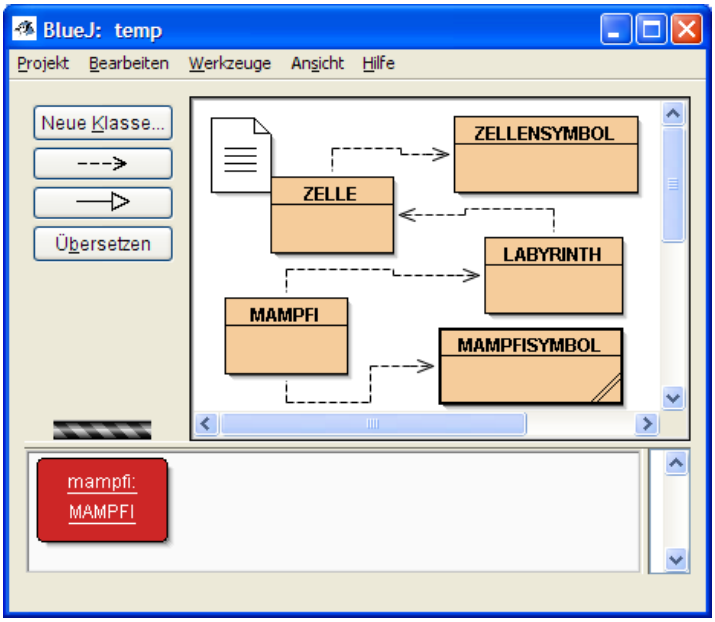

Abbildung 5: Situation nach der Erzeugung eines Objekts der Klasse MAMPFI in BlueJ.

Erzeugt man dann ein Objekt der Klasse MAMPFI, so ist zwar die Beziehung aus Abbildung 2 umgesetzt. Jedoch hat man als Aufrufer, nicht die Möglichkeit, Methoden des Labyrinths aufzurufen. Beispielsweise ist es nach dem Erzeugen von Mampfi nicht möglich die

Methode *GaengeErstellen* des Labyrinths aufzurufen, da das Labyrinth-Objekt von außen nicht sichtbar ist (Abbildung 5). Ausschließlich Mampfi kann Methoden des Labyrinths aufrufen, da ausschließlich das Objekt Mampfi eine Referenz auf das Labyrinth hat.
Beim dem Mampfi darstellenden Symbol ist dies auch so: Nur Mampfi hat eine Referenz auf "sein" Objekt der Klasse MAMPFISYMBOL. In diesem Fall ist es jedoch ausreichend. Das Symbol ist sehr eng an Mampfi gekoppelt ist, es gibt keinerlei Notwendigkeit von außen Methodenaufrufe an das Symbol-Objekt zu schicken.

11.2 Methoden mit einem Rückgabewert

Mit dem Referenzattribut kann nun Mampfi alle Methoden des Labyrinths aufrufen, aber dort gibt es noch keine Methode, die eine Information über die Breite zurück gibt. Entsprechende Methoden *BreiteGeben* und *LaengeGeben* müssen in der Klasse LABYRINTH ergänzt werden (Abbildung 6).

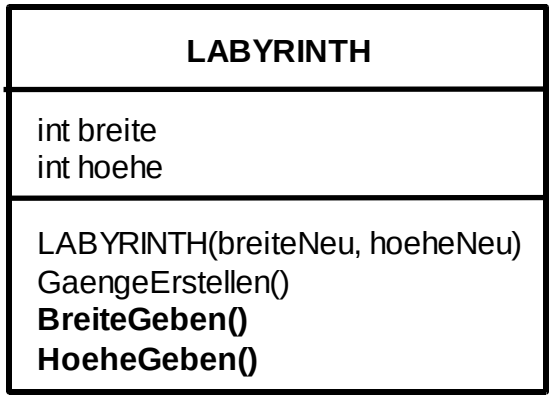

Abbildung 6: Ergänzungen in der Klasse LABYRINTH: Methoden *BreiteGeben* und *HoeheGeben*

Grundlegend unterschiedlich zu den bisherigen Methoden ist, dass beim Aufruf der Methoden *BreiteGeben* und *LaengeGeben* eine Information an den Aufrufer zurückgegeben wird. Diese Information wird **Rückgabewert** genannt. Ähnlich zur Deklaration von Attributen bzw. zur Festlegung von Eingangsparametern bei Methoden muss zuerst der Datentyp des Rückgabewerts festgelegt werden. Da sowohl die Breite als auch die Länge ganze Zahlen sind, ist int der richtige Datentyp für den Rückgabewert. Diese Überlegung lässt sich im erweiterten Klassendiagramm festhalten (Abbildung 7).

Abbildung 7: Der Rückgabewert der Methoden *BreiteGeben* und *HoeheGeben* hat als Datentyp int.

In Java sieht die Umsetzung der Methode *BreiteGeben* wie folgt aus:

```
int BreiteGeben()
{
    return breite;
}
```

Abbildung 8: Quelltext der Methode *BreiteGeben*

Die Methode besteht aus einer einzigen Anweisung, den Wert des Attributs breite zurückzugeben. Dies erfolgt über die Rückgabeanweisung **return**[1]. Jede Methode mit Rückgabewert muss eine Rückgabeanweisung enthalten, sonst gibt der Compiler eine Fehlermeldung aus.

1 engl.: zurückgeben, antworten

Hinweis:
Wie schon bei Methoden ohne Rückgabewert steht im erweiterten Klassendiagramm der **Methodenkopf** (erste Zeile einer Methodenbeschreibung).

Allgemein haben in Java Methoden mit Rückgabewert folgende Form:

```
DatentypRückgabewert Methodenname(Parameterliste)
{
    //Beschreibung der
     //Anweisungsfolge
   return AttributOderKonkreterWert;
}
```

Abbildung 9: Allgemeiner Aufbau einer Methode mit Rückgabewert und Eingangsparametern

Wie Abbildung 9 zeigt, sind – falls nötig – auch bei einer Methode mit Rückgabewert Eingangsparameter möglich. **Vor** der Rückgabeanweisung können andere Anweisungen ausgeführt werden.

Hinweise:
- An dieser Stelle wird der Sinn des bisher verwendete "void" vor dem Methodennamen nochmal sehr deutlich: Im Gegensatz zur Angabe des Datentyps des Rückgabewerts kennzeichnet das Wort void (engl. nichtig, ungültig) , dass es eben **keinen** Rückgabewert gibt.
- Quelltext nach der Rückgabeanweisung wird nicht mehr ausgeführt.
- Jede Methoden kann zwar mehrere Eingabewerte, jedoch höchstens einen Ausgabewert haben.
- Methoden mit einem Rückgabewert werden auch als Funktionen bezeichnet.

Aufgabe 11.5
Ergänze in der Klasse LABYRINTH die Methoden *BreiteGeben* und *HoeheGeben* und teste sie für mindestens zwei unterschiedliche Labyrinthgrößen..

Aufgabe 11.6
Ändere die Methoden *NachOstenGehen* und *NachSuedenGehen* in der Klasse MAMPFI so um, dass mit Hilfe einer Objektkommunikation wie in Abbildung 8 dargestellt Mampfi für ein beliebig breites bzw. langes Labyrinth nicht aus der Spielfläche wandert. Teste an mindestens zwei unterschiedlichen Labyrinthgrößen.

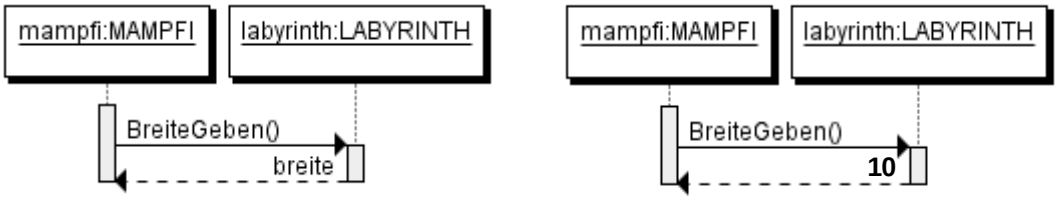

Abbildung 10: Objektkommunikation allgemein (links) und speziell mit dem bisher von uns verwendeten Labyrinth der Breite 10 (rechts)

11.3 Voraussetzung für eine funktionierende Objektkommunikation

Kapitel 11.1 und 11.2 macht nochmal deutlich, welche beiden Voraussetzungen für eine Objektkommunikation erfüllt sein müssen. In den beiden Kapiteln wurden Vorbereitungen getroffen, damit in der Klasse MAMPFI folgender Methodenaufruf möglich ist:

```
labyrinth.BreiteGeben()
```

Über diesen Methodenaufruf erfragt Mampfi beim Labyrinth dessen Breite (Abbildung 10). Dies funktioniert nur, weil

1) Mampfi eine Referenz auf das Labyrinth hat (Abbildung 11). Mampfi kann somit das richtige Objekt fragen.
2) Das Labyrinth eine Methode hat (Abbildung 7), die genau zur Anfrage von Mampfi passt.

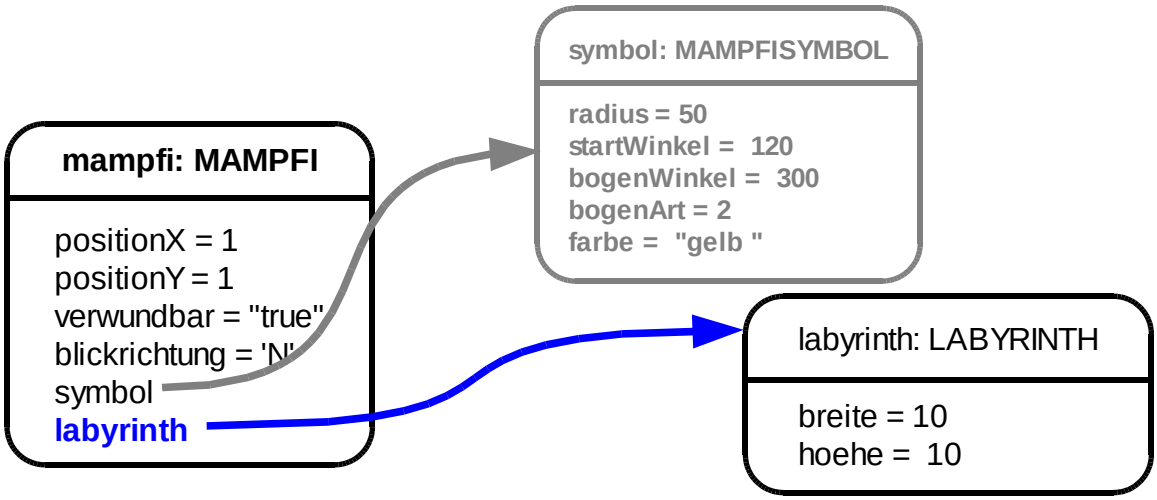

Abbildung 11: Das Referenzattribut labyrinth in der Klasse MAMPFI ermöglicht eine Anfrage von Mampfi an das Labyrinth

Im Folgenden sind die beiden Voraussetzungen für die Objektkommunikation nochmals allgemein formuliert.

1) **Das die Nachricht sendende Objekt hat eine Referenz auf das empfangende Objekt.** D. h. eine Objektbeziehung muss vorhanden sein. Nur so kann die Kommunikation zielgerichtet vom Sender zum Empfänger durchgeführt werden.
2) **Das die Nachricht empfangende Objekt besitzt als Schnittstelle eine passende Methode.** Da die Objektkommunikation über Methodenaufrufe abläuft, kann der Sender (ohne Fehlermeldung) nur die Methoden aufrufen, die der Empfänger anbietet.

Aufgabe 11.7

Begründe, ob die folgenden Methodenaufrufe **in einer Klasse MAMPFI** möglich wären oder nicht. Die am Seitenende abgebildeten erweiterten Klassendiagramme helfen bei einer stichhaltigen Begründung.

a) symbol.RandSichtbarSetzen("gelb")
b) mampfisymbol.FuellFarbeSetzen("rot")
c) labyrinth.SpielFlaecheGeben()
d) zelle.IstMauerSetzen(true)
e) symbol.StartWinkelSetzen(30)
d) labyrinth.HoeheGeben(10)
e) symbol.NachNordenBlicken()
f) zelle.positionYSetzen(5)
g) labyrinth.BreiteGeben()

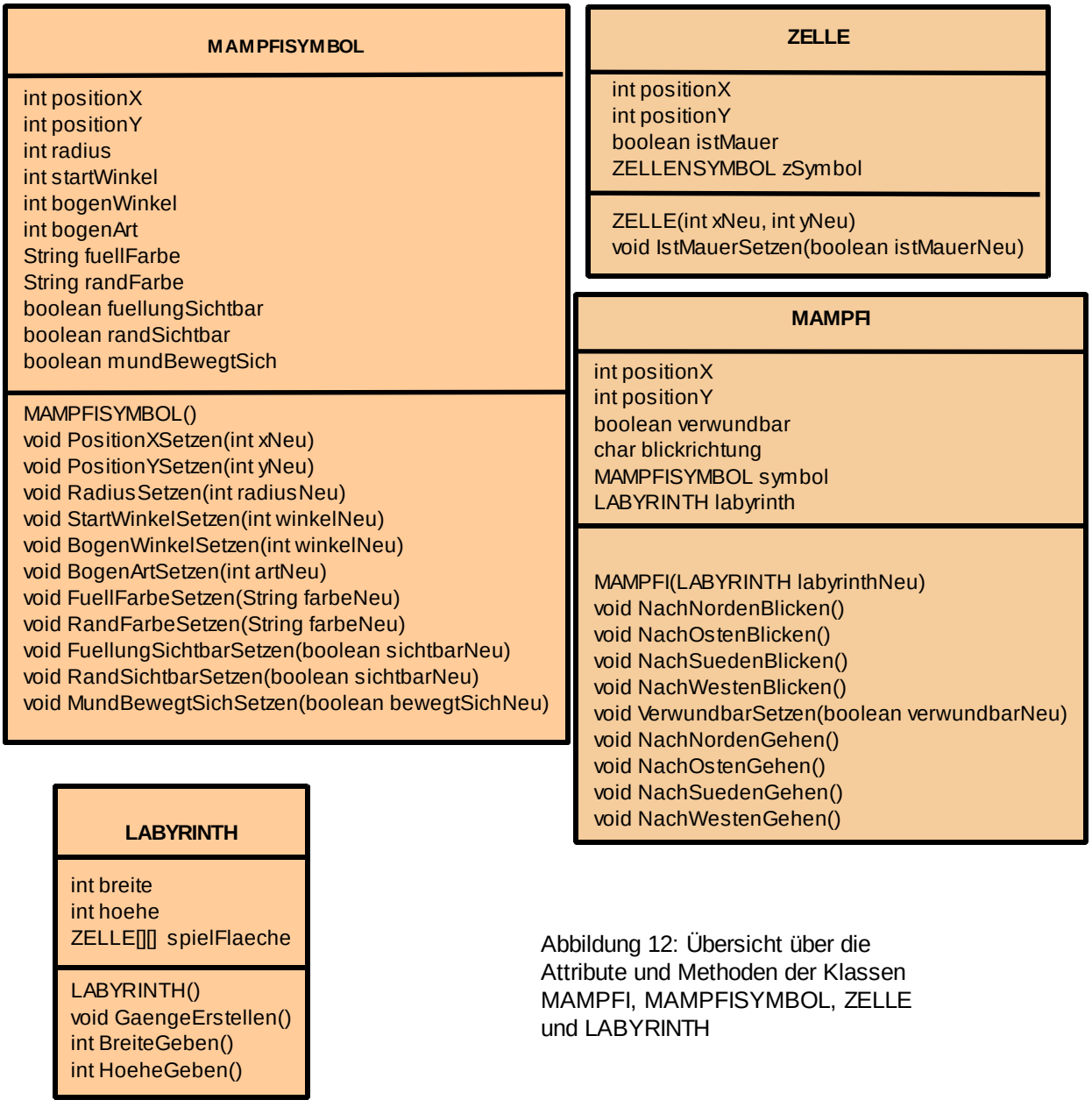

Abbildung 12: Übersicht über die Attribute und Methoden der Klassen MAMPFI, MAMPFISYMBOL, ZELLE und LABYRINTH

11.4 Mauern als Hindernisse bei der Bewegung von Mampfi

Über einen Aufruf der Methode *GaengeErstellen* des Labyrinths werden einige Zellen der Spielfläche zu Mauern (Aufgabe 11 in Kapitel 9). Ziel ist es, dass Mampfi solche Mauern nicht betreten kann.

Aufgabe 11.8
Ergänze dein BlueJ-Projekt so, dass Mampfi Mauern auf der Spielfläche nicht betreten kann. Folgende Überlegungen/Schritte sind dabei hilfreich:

- In welchem Objekt ist die Information Mauer bzw. keine Mauer gespeichert?
- Wie kann Mampfi zu dieser Information gelangen? (Tipp: Eine Kommunikation kann auch über Zwischenstationen erfolgen. --> Welche Methoden müssen in welchen Klassen ergänzt werden?)
- Wo in der Klasse MAMPFI müssen Ergänzungen vorgenommen werden?

Hinweis:
Diese Aufgabe ist anspruchsvoll, da zur Lösung mehrere Veränderungen in unterschiedlichen Klassen nötig sind. Du benötigst jedoch keine neuen Kenntnisse, deshalb versuche dich an der Aufgabe. Solltest du an einer Stelle nicht mehr weiterkommen, so findest du auf den folgenden Seiten eine schrittweise Anleitung.

Wenn Mampfi bei seinen Bewegungen im Labyrinth (z. B. durch den Aufruf der Methoden *NachNordenGehen*, *NachOstenGehen*) keine Mauern betreten soll, benötigt er die Information, ob auf der Zelle, auf die er gehen soll, eine Mauer ist oder nicht.

Aufgabe 11.9
Wo ist diese Information gespeichert? Das betreffende Objekt hat noch noch keine Methode, mit der es die Information nach außen kommunizieren kann. Wie lautet der Kopf solch einer Methode?

In der Klasse Zelle gibt es ein Attribut IstMauer. Ist der Wert true, so ist auf dieser Zelle eine Mauer, sonst nicht. Damit andere Objekte diese Information abfragen können, benötigt die Klasse ZELLE eine Methode *IstMauerGeben*. Der Datentyp des Rückgabewerts ist boolean, weil das Attribut IstMauer von diesem Datentyp ist. Die Methode benötigt keine Eingabewerte, so dass der Methodenkopf entsprechend dem erweiterten Klassendiagramm in Abbildung 13 ergibt.

Aufgabe 11.10
Ergänze in der Klasse ZELLE deines BlueJ Projekts die Methode *IstMauerGeben* und teste sie. Beim Testen reicht das Erzeugen eines einzigen Zellenobjekts aus. Jedoch sollte das korrekte Arbeiten der Methode *IstMauerGeben* für die Werte true und false des Attributs IstMauer überprüft werden.

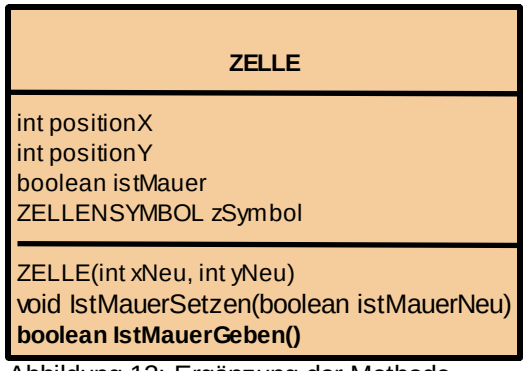

Abbildung 13: Ergänzung der Methode *IstMauerGeben* in der Klasse ZELLE

Aufgabe 11.11
Warum kann Mampfi die Information, ob auf einer Zelle eine Mauer ist, noch nicht abfragen, obwohl jedes Zellenobjekt jetzt die Methode *IstMauerGeben* besitzt? Was muss geändert werden, um die Objektkommunikation zu ermöglichen?

Die Objektkommunikation ist noch nicht möglich, da es noch keine Beziehung zwischen Mampfi und Objekten der Klasse ZELLE gibt. Da die Klasse MAMPFI keine Referenz auf Zellen hat, kann die Anfrage nicht zielgerichtet gesendet werden.
Eine Möglichkeit wäre es, diese Beziehung, wie in Abbildung 14, dargestellt herzustellen.

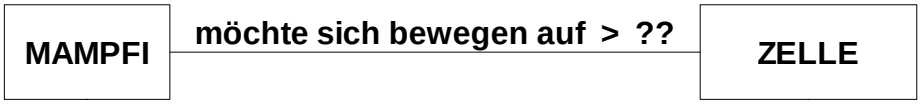

Abbildung 14: Beziehung zwischen Objekten der Klassen MAMPFI und ZELLE

Jedoch ist dieser Weg mit weiteren Fragen bzw. vielen Umstellungen verbunden, beispielsweise:

- Zu wie vielen Zellen muss eine Beziehung hergestellt werden? Zu einer, der Zelle der Zielposition? Zu vier, zu den Nachbarzellen in nördlicher, östlicher, südlicher und westlicher Richtung? Zu allen Zellen des Labyrinths?
- Werden nicht alle Zellen des Labyrinths referenziert, müssen nach jedem Schritt z. B. die vier Nachbarzellen aktualisiert werden, denn sie haben sich verändert.
- Werden alle Zellen des Labyrinths referenziert stellt sich die Frage, ob dies nicht doppelte Arbeit ist, denn diese Referenzen gibt es schon in der Klasse LABYRINTH.

Aus diesen Gründen ist es geschickter für die Anfrage von Mampfi einen Umweg über das Labyrinth zu nehmen, denn indirekt gibt es schon eine Beziehung zwischen Mampfi und den Zellen (Abbildung 15).

Abbildung 15: Über die "Zwischenstation" Labyrinth gibt es bereits eine Beziehung zwischen Mampfi und den Zellen.

Mampfi würde somit das Labyrinth fragen "Ist auf der Zelle mit der posittionX und positionY eine Mauer?". Da das Labyrinth das zunächst auch nicht weiß, fragt es die beteffende Zelle mit Hilfe des Aufrufs der Methode *IstMauerGeben*. Die Antwort, die das Labyrinth von der Zelle erhält gibt es an Mampfi weiter. In Abbildung 16 ist die Kommunikation in einem Sequenzdiagramm dargestellt. An dem Rückgabewert `true` erkennt man, dass die Zelle ist in diesem Fall eine Mauer ist.

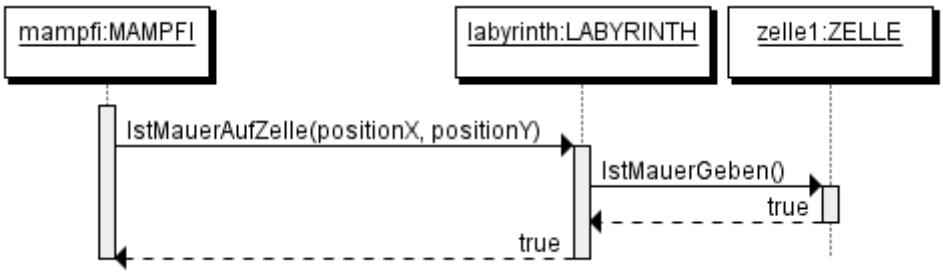

Abbildung 16: Anfrage von Mampfi über die "Zwischenstation" Labyrinth, ob auf der Zelle eine Mauer ist.

Aufgabe 11.12
Warum ist das Labyrinth noch nicht "bereit" für die Anfrage von Mampfi? Welche Ergänzung muss in der Klasse LABYRINTH vorgenommen werden? Antwort bereits möglichst genau!

Die Klasse LABYRINTH bietet noch keine Methode an, die Mampfi eine Anfrage ermöglicht. Es muss eine Methode ergänzt werden. Ein aussagekräftiger Name für die Methode ist *IstMauerAufZelle* . Die Methode muss einen Wert vom Typ boolean zurückgeben und sie benötigt als Eingabewerte die Position der Zelle, die gefragt werden soll. Somit ergibt sich der Methodenkopf wie in nebenstehendem Klassendiagramm abgebildet.

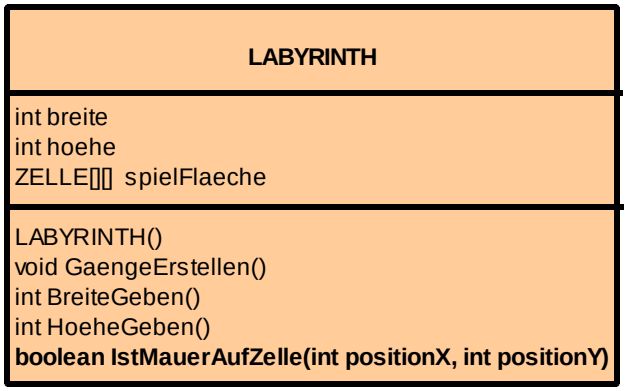

Abbildung 17: Ergänzung der Methode *IstMauerAufZelle* in der Klasse LABYRINTH

Aufgabe 11.13
Ergänze in der Klasse LABYRINTH deines BlueJ Projekts die Methode *IstMauerAufZelle*. Ein effektives Testen ist möglich, indem du zuerst ein Labyrinth erzeugst und die Methode *GaengeErstellen* dieses Labyrinths aufrufst. Dann hast du Zellen mit und ohne Mauern und kannst überprüfen, ob die Methode *IstMauerAufZelle* richtige Rückgabewerte gibt.

Endlich kann nun Mampfi die Information erhalten, ob auf einer Zelle eine Mauer steht oder nicht.

Aufgabe 11.14
Welche Methoden müssen nun in der Klasse MAMPFI geändert werden? Wie sehen die Änderungen in den Methodenrümpfen aus?

Alle Methoden in der Klasse MAMPFI, die für eine Bewegung zuständig sind müssen ergänzt werden. Es bleibt die Abfrage, ob die Bewegung wegen der Labyrinthgrenzen durchgeführt werden kann. Ist dies der Fall, darf der Schritt jedoch nur ausgeführt werden, wenn die Zielzelle keine Mauer ist. Somit ist eine weitere bedingte Anweisung nötig. Als Bedingung kann man einen Aufruf der Methode *IstMauerAufZelle* an das Labyrinth verwenden, der entsprechend mit true oder false beantwortet wird. Die Bewegung wird nur durchgeführt, wenn dieser Rückgabewert false ist, d.h. die Zielzelle keine Mauer ist.

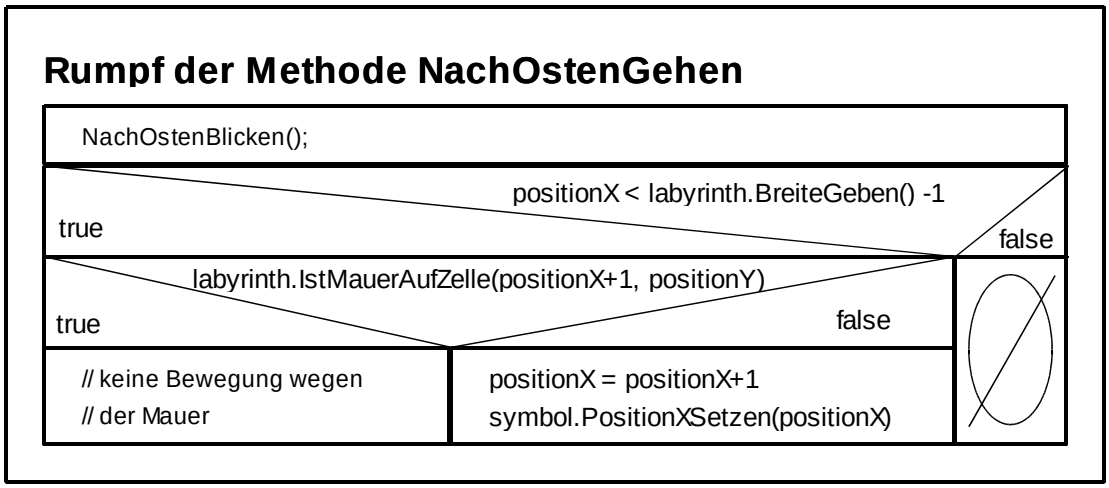

Abbildung 18: Anfrage von Mampfi über die "Zwischenstation" Labyrinth, ob auf der Zelle eine Mauer ist.

Aufgabe 11.15
Wie lauten die beiden Bedingungen in der Methode *NachNordenGehen*?

Aufgabe 11.16
Ergänze in der Klasse MAMPFI deines BlueJ Projekts die Methoden *NachOstenGehen*. *NachNordenGehen* usw. und teste diese in einem Labyrinth mit Mauern.

Aufgabe 11.17
a) Führe ein Rollenspiel passend zur Situation in Abbildung 17 durch. Beteiligte Objekte sind der Aufrufer, Mampfi, das Labyrinth und zwei Zellen mit den Namen Zelle1 und Zelle2. Der Aufrufer ruft zuerst die Methode NachWestenGehen und dann die Methode NachNorden gehen auf.
b) Zeichne das Sequenzdiagramm zu a)

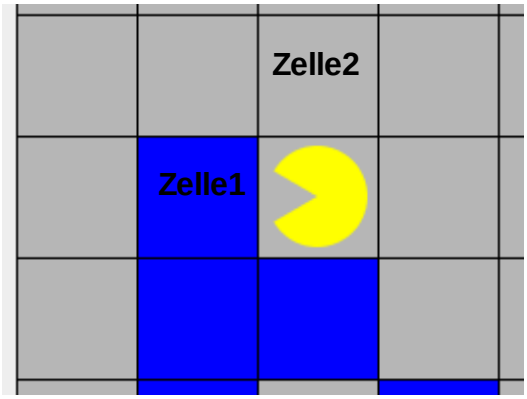

Abbildung 19: Ausschnitt aus einer Situation im Labyrinth.

11.5 Zusammenfassung

Aufgabe 11.18
Fasse die wesentlichen Inhalte dieses Kapitels in deinem Heft zusammen. Neu sind Methoden mit Rückgabewerten.
Wiederholt und vertieft wurden wichtige Voraussetzungen für eine funktionierende Objektkommunikation.

Aufgabe 11.19 Alternative Bewegung Teil 2 (Fortsetzung Aufgabe 10.5)
Passe die Methoden *VorwaertsGehen*, *RechtsGehen* usw. so an, dass Mampfi vor Mauern stehen bleibt.

Aufgabe 11.20 Geheimwege Teil 2 (Fortsetzung Aufgabe 10.6)

In Abbildung 5 von Kapitel 10 sind die Zellen, von denen aus Geheimwege möglich sind durch eine andere Farbe markiert. Überlege wie sich dies modellieren und in dem BlueJ Projekt dann umsetzen lässt.
Eine alternative Möglichkeit für Geheimwege ist, dass beim Erreichen eines besonderen Feldes Mampfi zu einem anderen Feld springt, ähnlich wie beim Berühren eines Portkeys in dem Roman "Harry Potter". Überlege ebenfalls, wie sich dies modellieren und in dem BlueJ Projekt umsetzen lässt.