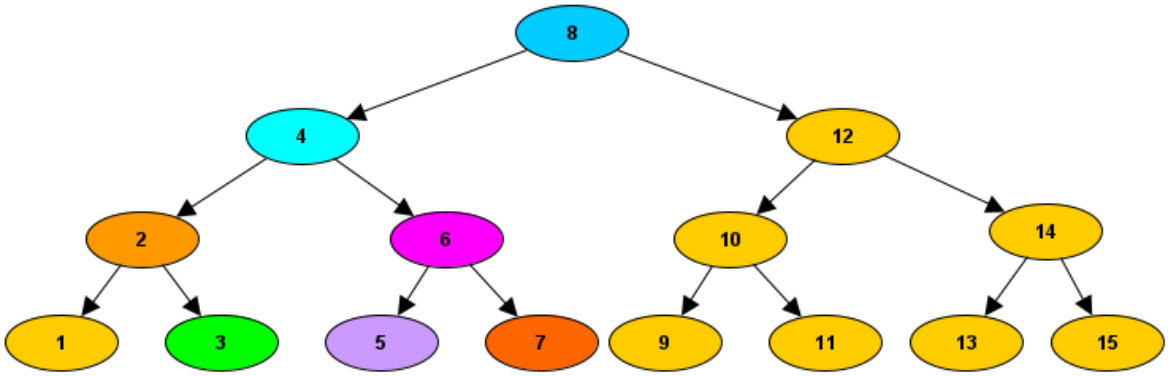
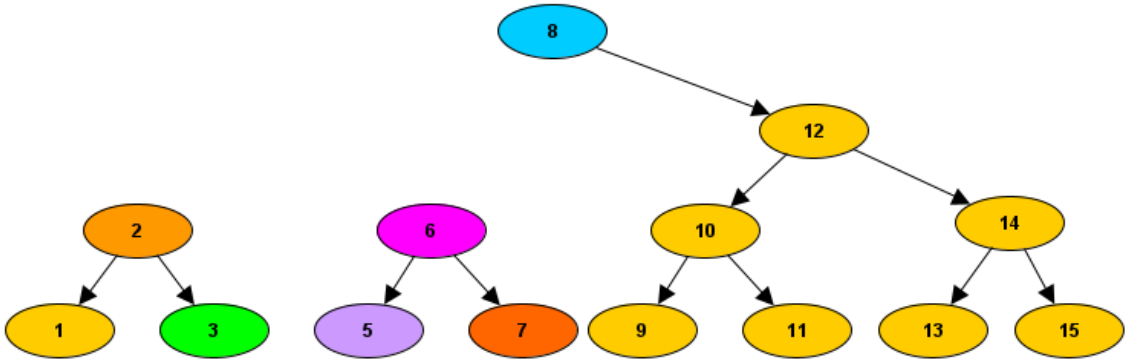
2.5 Entfernen von KNOTEN

Um den BAUM abzuschließen fehlt uns nun nur noch das Entfernen von KNOTEN. Man stellt schnell fest, dass das Entfernen eines Blattes nicht unbedingt das Problem ist. Vielmehr die Verallgemeinerung auf das Entfernen eines beliebigen KNOTEN ist problematisch.
Betrachten wir einen Beispielbaum:



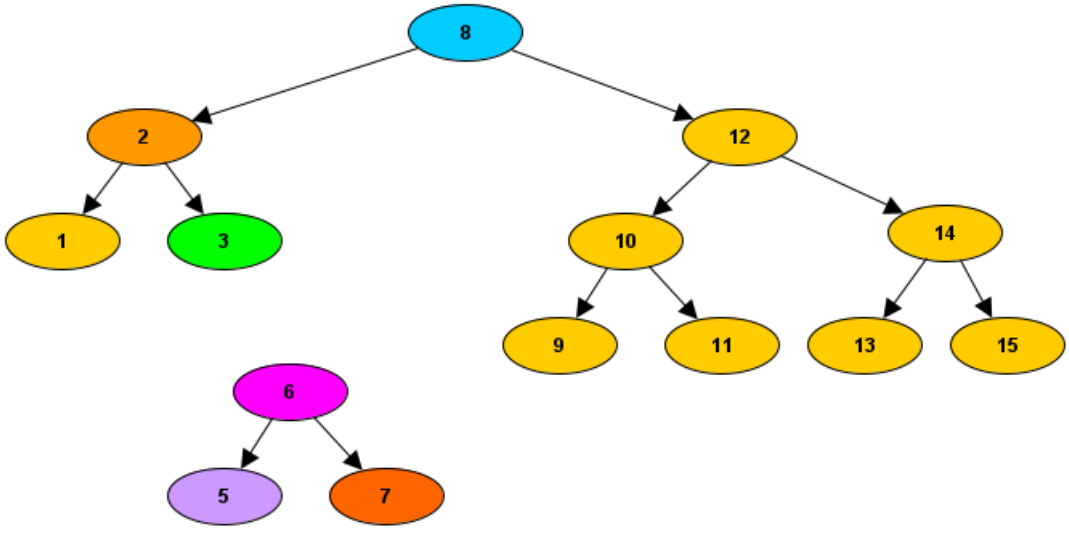
Überlegen wir uns, was beim Entfernen des KNOTEN „4“ passiert. Es entsteht quasi eine Lücke im BAUM:



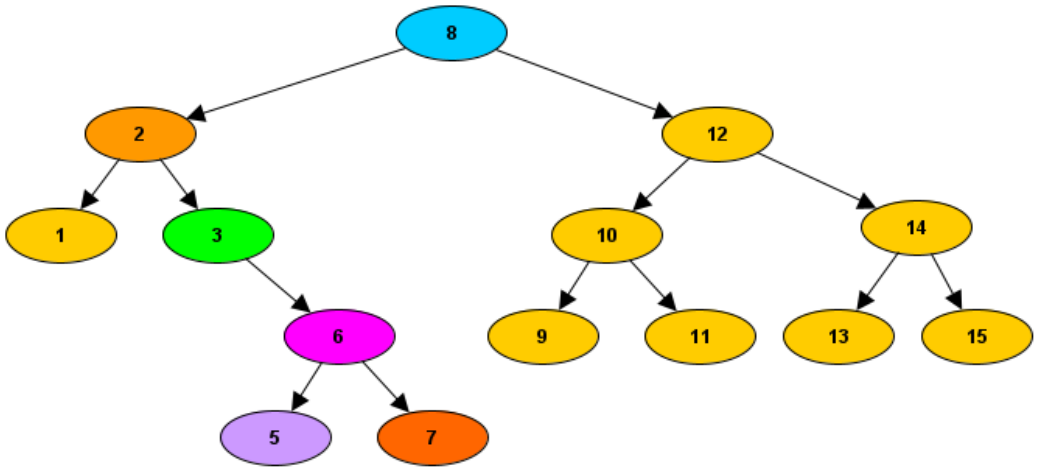
Es gibt nun zwei Möglichkeiten:

- Ersetze den entfernten KNOTEN durch seinen linken Nachfolger
- Ersetze den entfernten KNOTEN durch seinen rechten Nachfolger

Theoretisch ist es egal, daher betrachten wir hier die Version mit dem linken Nachfolger:



Nun stellt sich die Frage, wo man den KNOTEN „6“ mit seinen Nachfolgern Einfügen sollte. Da alle KNOTEN dieses Teilbaumes vorher rechts von „4“ waren, müssen sie logischerweise nun auch rechts des „größten“ KNOTENs im neu angeordneten Teilbaum („1“,“2“,“3“) sein. Wir fügen „6“ also rechts von „3“ ein:

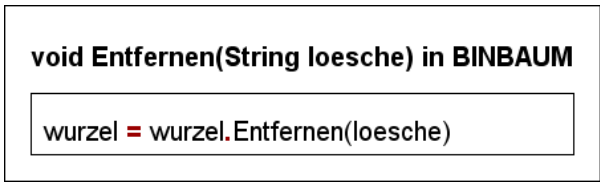


Bleibt nur noch dies in Pseudocode/Struktogramme zu übersetzen.

Wir stellen uns wieder zu jeder Klasse die Fragen, ob die Methoden einen Rückgabewert und/oder Parameter brauchen.
Da wir einen KNOTEN entfernen und diesen ersetzen (durch einen Nachfolger, ggf. einen ABSCHLUSS), wird die Methode im KNOTEN und ABSCHLUSS einen Rückgabewert haben, da mit dem Vorgänger kommuniziert werden muss. Wir betrachten mit dieser Überlegung die einzelnen Klassen.

In BINBAUM:

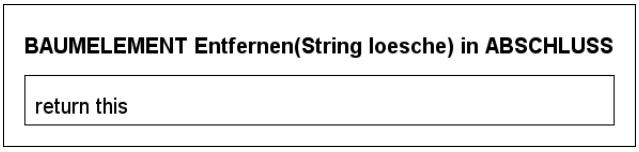
Die Methode in BINBAUM muss, wie meist, nur die Anfrage an seine Wurzel weitergeben. Dabei könnte der zu entfernende KNOTEN die Wurzel selber sein. Dann würde diese entfernt und durch ihren linken Nachfolger ersetzt. Die Wurzel würde dann die neue Wurzel an den BINBAUM melden daher folgendes Struktogramm:



Die Wurzel wird auf die Rückgabe der Methode des KNOTEN gesetzt. Hier ändert sich nur etwas, wenn wir die Wurzel selber entfernen.

In ABSCHLUSS:

Der ABSCHLUSS muss nie entfernt werden. Bei ihm sollte die Entfernen Methode nur ankommen, wenn der gesuchte KNOTEN nicht im BAUM existiert hatte.



Der ABSCHLUSS gibt sich selber an den Vorgänger zurück.

In KNOTEN:

Der KNOTEN ist nun der Knackpunkt der Methode. Er muss die ganze Arbeit übernehmen. Glücklicherweise können wir schon auf die anderen Methoden des BAUM zurückgreifen und uns so die Arbeit erleichtern:

BAUMELEMENT Entfernen(String loesche) in KNOTEN

<table>
<tr><td rowspan="3">BAUMELEMENT t;
t = linksNachfolger.Einfuegen(rechtsNachfolger);
return t;</td><td colspan="2">MeineDaten.IstGleich(loesche)
Wahr — Falsch</td></tr>
<tr><td colspan="2">MeineDaten.IstKleinerAls(loesche)
Wahr — Falsch</td></tr>
<tr><td>rechtsNachfolger = rechtsNachfolger.Entfernen(loesche)</td><td>linksNachfolger = linksNachfolger.Entfernen(loesche)</td></tr>
<tr><td colspan="3">return this</td></tr>
</table>

Wenn die Daten des KNOTEN mit den Gesuchten übereinstimmen, dann löscht er sich, indem er seinen rechten Nachfolger an den linken weitergibt und dann den linken Nachfolger zurückgibt. Wichtig ist es hier, eine Einfügen Methode zu haben, die BAUMELEMENTE einfügen kann, da sonst der „Link“ zum KNOTEN und seinen Nachfolgern verloren geht. Die Einfügen Methode sorgt dann dafür, dass der rechte Nachfolger passend beim linken Nachfolger eingefügt wird. Dadurch, dass der BAUM über Referenzen arbeitet, reicht es den Nachfolger zu verschieben, da alle darunter kommenden KNOTEN (im Bsp oben „5“ und „7“) automatisch richtig eingeordnet, da sie als Nachfolger von „6“ vernetzt sind (Achtung: nur in der Einfügen Methode mit BAUMELEMENTEN).
Wenn der KNOTEN nicht der zu löschende ist, dann geht es ähnlich, wie beim Einfügen. Je nachdem in welchem der Nachfolger-Teilbäume der zu löschende KNOTEN sein sollte, gibt man an den entsprechenden Nachfolger weiter. Als Rückgabe gibt er sich selber zurück, da sich an ihm nichts ändern sollte.

Hinweis: Die Einfügen Methode mit KNOTEN sollte im ABSCHLUSS nicht ABSCHLUSS Objekte als Nachfolger setzen, da es sein kann, dass wir einen KNOTEN einfügen, der schon Nachfolger hat.

Aufgaben:

1) Überlege anhand eines BAUM mit mindestens drei Ebene, wie der Ablauf des Entfernen-Algorithmus ablaufen sollte. Betrachte das Entfernen eines Blattes, eines inneren Knotens und der Wurzel.

2) Überlege dir, welche Aufgabe beim Entfernen jeweils den Klassen ABSCHLUSS, BINBAUM und KNOTEN zukommt. Entwickle Struktogramme, die den Ablauf veranschaulichen.

3) Ergänze deine Klassen BINBAUM, BAUMELEMENT, KNOTEN und ABSCHLUSS um die Methode Einfügen mit Parameter des Datentyps BAUMELEMENT.

4) Übernehme deine Struktogramme in den Code deiner Vorlage (Swing_Binbaum_04). Teste deinen Algorithmus ausführlich.

5) Wir haben gesehen, dass das Entfernen von KNOTEN dafür sorgen kann, dass der BAUM an Tiefe gewinnt, wenn wir Elemente entfernen. Erweitere deinen KNOTEN um die Funktionalität, dass er ausgeben kann, wie tief der BAUM ab ihm selber noch ist. Überlege, wie man diese Funktionalität nutzen könnte, um beim Entfernen die Variante zu wählen, die den BAUM weniger wachsen lässt.

6) Schreibe eine weitere Methode, die überprüft, ob der BAUM ideale Höhe hat und falls nicht den BAUM neu aufbaut, so dass er ideal gebaut ist. Tipp: Nutze die neuen Methoden des BAUM.