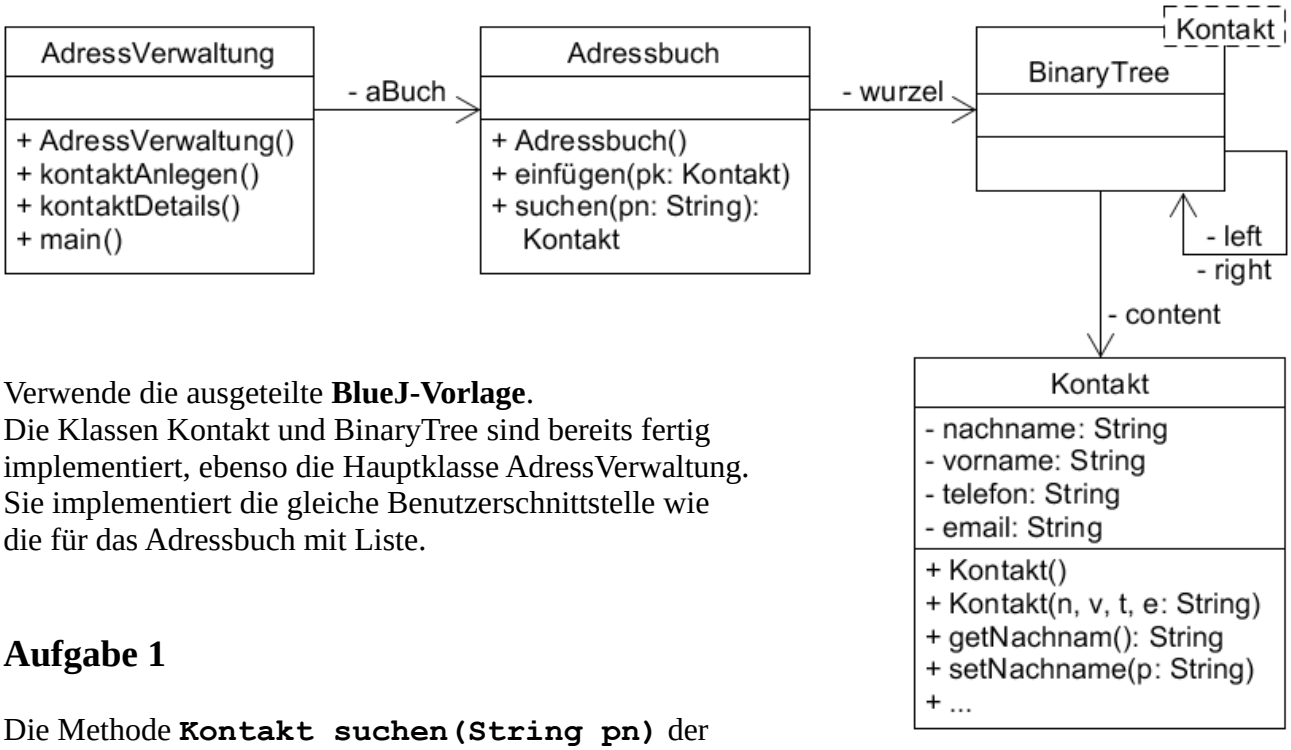
Wie du bereits gesehen hast, kann man für ein Adressbuch einen Suchbaum zur Speicherung der Kontakte verwenden. Das folgende Klassendiagramm zeigt das schon von den linearen Datenstrukturen bekannte Modell, bei dem die Klasse List durch die Klasse BinaryTree ersetzt wurde:



Verwende die ausgeteilte **BlueJ-Vorlage**.
Die Klassen Kontakt und BinaryTree sind bereits fertig implementiert, ebenso die Hauptklasse AdressVerwaltung.
Sie implementiert die gleiche Benutzerschnittstelle wie die für das Adressbuch mit Liste.

Aufgabe 1

Die Methode **`Kontakt suchen(String pn)`** der Klasse Adressbuch erhält einen Nachnamen als Parameter.
Sie sucht einen Kontakt im Suchbaum mit diesem Nachnamen.
Falls dieser Kontakt vorhanden ist, gibt die Methode ihn zurück, andernfalls null.

Entwickle zunächst einen **Algorithmus** für diese Aufgabe in Umgangssprache.
Dann implementiere die Methode.

Zur Erinnerung: alphabetischer Vergleich von Strings mit der Methode compareTo()

str1.compareTo(str2) < 0 → str1 liegt alphabetisch VOR str2
str1.compareTo(str2) = 0 → beide liegen an der gleichen Stelle (d.h. sie sind gleich)
str1.compareTo(str2) > 0 → str1 liegt alphabetisch HINTER str2

Aufgabe 2

Die Methode **`void einfügen(Kontakt pk)`** erhält ein bereits mit Daten gefülltes Kontakt-Objekt als Parameter und fügt dieses an der richtigen Stelle im Suchbaum ein.
Zur Vereinfachung nehmen wir an, dass es keine Kontakte mit dem gleichen Nachnamen gibt.

Entwickle zunächst einen **Algorithmus** für diese Aufgabe in Umgangssprache.
Dann implementiere die Methode.