DNS-Server

Das Domain Name System (DNS) funktioniert ähnlich wie eine Telefonauskunft. Ein Client schickt einen Domain-Namen wie z. B. *google.de* an seinen DNS-Server und dieser liefert die zugehörige IP-Adresse zurück. Der DNS-Server verwaltet die Domain-Namen und zugehörigen IP-Adressen mithilfe eines binären Suchbaums, wobei die Domain-Namen als Schlüssel dienen. Für die Ordnung der Schlüssel gilt die alphabetische Reihenfolge.

Bei jeder DNS-Anfrage wird zuerst der binäre Suchbaum nach dem Domain-Namen durchsucht. Wird dieser im Suchbaum gefunden, so wird die zugehörige IP-Adresse zurückgeliefert. Andernfalls wird zeitaufwändig ein übergeordneter DNS-Server befragt und dessen Antwort im binären Suchbaum als neuer Knoten gespeichert. Anschließend wird die entsprechende IP-Adresse an den Client zurück-geliefert.

Aufgaben

1 Überführen Sie die beiden Klassen aus Material 1 in ein UML-Klassendiagramm.

(5 BE)

2 Skizzieren Sie den binären Suchbaum, der sich ergibt, wenn Clients nacheinander nach den folgenden Domain-Namen suchen: google.de, ard.de, zdf.de, gez.de, google.de, ct.de, kino.de Es soll davon ausgegangen werden, dass der Suchbaum vor der ersten Anfrage leer ist. IP-Adressen müssen in der Skizze nicht angegeben werden.

(2 BE)

3 Implementieren Sie die Methode *ip_von_domain(self, domain_name: str)* der Klasse *BinBaum*, die zu einem gegebenen Domain-Namen die zugehörige IP-Adresse liefert und gegebenenfalls im Suchbaum einen neuen Knoten speichert.
Mit der Methode *ip_erfragen(self, domain_name: str)* der Klasse *BinBaum* kann die IP-Adresse zu einem Domain-Namen ermittelt werden, der noch nicht im Suchbaum gespeichert ist.

(8 BE)

4 Wenn ein DNS-Server eine bestimmte Anfrage erhält, ist davon auszugehen, dass die gleiche Anfrage in nächster Zeit noch mehrmals erfolgen wird. Zur Optimierung der Zugriffszeit soll daher der Suchbaum bei jeder Anfrage so umgeordnet werden, dass der aktuell gefundene Knoten zur Wurzel wird. Ein binärer Suchbaum mit dieser Eigenschaft heißt *Spreizbaum*.

Damit die Ordnung im binären Suchbaum erhalten bleibt, wird wie folgt umgeordnet: Hat der gefundene Knoten k einen Elternknoten e, so nimmt k die Position von e ein. War k linkes Kind von e, so behält k seinen linken Teilbaum. Der Elternknoten e wird samt seinem rechten Teilbaum rechtes Kind von k. Der rechte Teilbaum von k wird linker Teilbaum von e. Analoges gilt, wenn k rechtes Kind von e ist. Die Aufstiegsbewegung wird solange wiederholt, bis der Knoten k Wurzel des Baumes ist.

Stellen Sie den Vorgang des Umordnens am Suchbaum in Material 2 für den Knoten k = "m" dar. Skizzieren Sie dazu für jeden Umordnungsschritt den binären Suchbaum.

(5 BE)

5 Der Suchbaum des DNS-Servers soll durch einen Spreizbaum ersetzt werden. Die Klasse Spreizbaum hat die Methode *umordnen()*, welche die Umordnungsschritte durchführt. Sie benötigt den Pfad von der Wurzel zu dem Knoten k, der bei der letzten Suche gefunden worden ist. Im Beispiel aus Aufgabe 4 ist dies der Pfad c-t-m. Der Pfad soll in der Klasse *Spreizbaum* als Attribut vom Typ *Keller* gespeichert werden.

5.1 Entwerfen Sie unter Berücksichtigung von Material 3 eine Erweiterung Ihres UML-Klassendiagramms aus Aufgabe 1 um die Klassen *Spreizbaum*, *Keller* und *Element* und erläutern Sie anhand der Klasse *Spreizbaum* das Überschreiben einer Methode.

(8 BE)

5.2 Die Methode *umordnen()* hat verschiedene Fälle zu berücksichtigen, denn k kann die Wurzel, ein innerer Knoten oder ein Blatt sein. Für den Fall, dass k ein Blatt ist, ruft *umordnen()* die Methode *umordnen_blatt()* auf, welche ein Blatt des Spreizbaums an die Position seines Elternknotens aufsteigen lässt. Dieser Umordnungsschritt ist für das Blatt k = "p" in Material 4 dargestellt. Der Pfad zu diesem Blatt ist mit seiner Datenstruktur in Material 5 dargestellt.

Implementieren Sie die Methode *umordnen_blatt()* der Klasse *Spreizbaum*. Zur Vereinfachung darf davon ausgegangen werden, dass der Pfad zum betroffenen Blatt mindestens die Länge drei hat und das Blatt somit einen Eltern- und einen Großelternknoten besitzt.

(7 BE)

Material 1

Klasse BinBaum

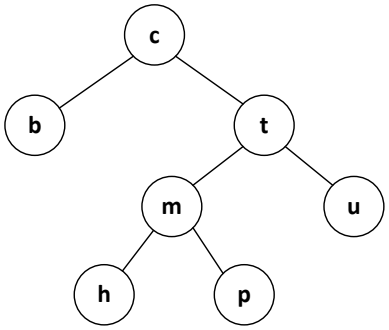
```
class BinBaum:

    def __init__(self):
        self._wurzel: Knoten = None

    def ip_von_domain(self, domain_name: str) -> str:
        # in Aufgabe 3 zu implementieren

    def ip_erfragen(self, domain_name: str) -> str:
        # hier muss ein übergeordneter DNS-Server angefragt werden
        # die Methode soll als implementiert angenommen werden
```

Klasse Knoten

```
class Knoten:

    def __init__(self, name: str, ip: str):
        self.__name: str = name
        self.__ip: str = ip
        self.__links: Knoten = None
        self.__rechts: Knoten = None

    def get_name(self) -> str:
        return self.__name

    def get_ip(self) -> str:
        return self.__ip

    def get_links(self) -> Knoten:
        return self.__links

    def set_links(self, links: Knoten):
        self.__links = links

    def get_rechts(self) -> Knoten:
        return self.__rechts

    def set_rechts(self, rechts: Knoten):
        self.__rechts = rechts
```

Material 2

Suchbaum



Material 3

Klasse Keller

```
class Keller:
    def __init__(self):
        self.__oberstes_element: Element = None

    def push(self, knoten: Knoten):
        neues_element: Element = Element(knoten)
        neues_element.set_nächstes(self.__oberstes_element)
        self.__oberstes_element = neues_element

    def pop(self):
        knoten: Knoten = None
        if self.__oberstes_element is not None:
            knoten = self.__oberstesElement.get_knoten()
            self.__oberstes_element =
                self.__oberstes_element.get_nächstes()
        return knoten
```

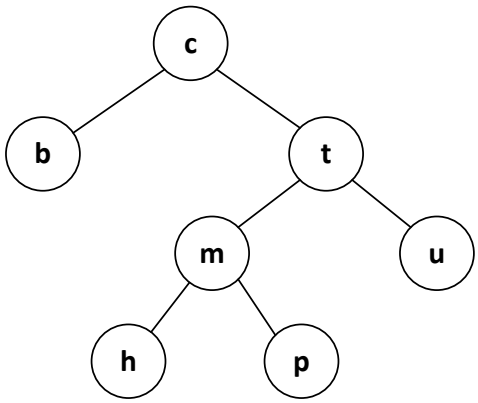
Klasse Element

```
class Element:
    def __init__(self, knoten: Knoten):
        self.__knoten: Knoten = knoten
        self.__nächstes: Element = None

    def get_knoten(self) -> Knoten:
        return self.__knoten

    def get_nächstes(self) -> Element:
        return self.__nächstes

    def set_nächstes(self, nächstes: Element):
        self.__nächstes = nächstes
```

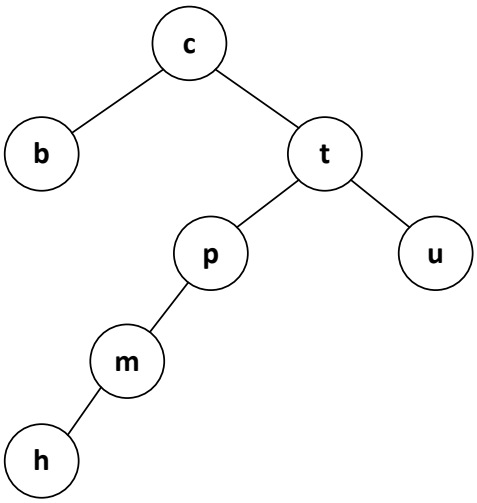
Material 4

Aufstieg des Blattes k = "p" an die Position seines Elternknotens

Suchbaum vor dem Aufruf von *umordnen_blatt()*

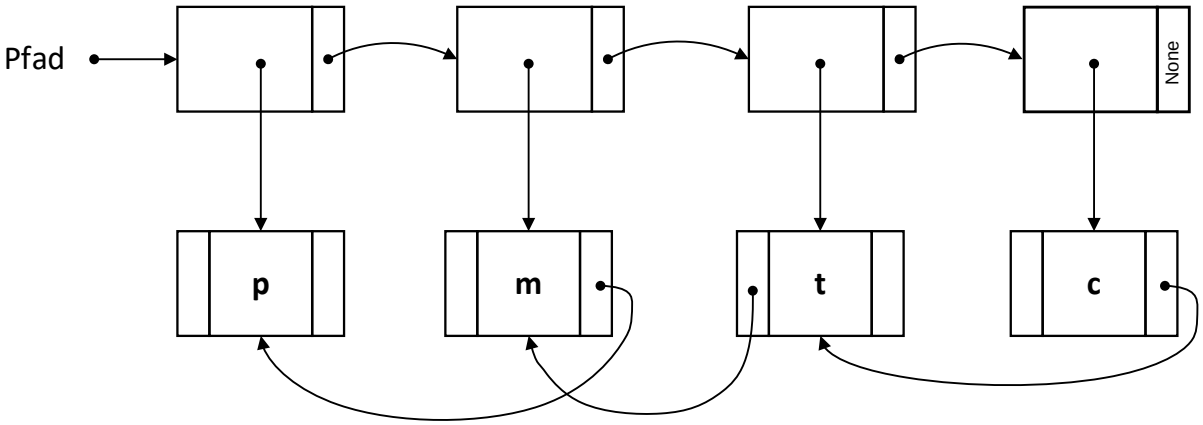


Suchbaum nach dem Aufruf von *umordnen_blatt()*



Material 5

Objektdiagramm der Datenstruktur Pfad



Hinweis
Für die Aufgabenstellung nicht relevante Zeiger und Objekte sind aus Gründen der Übersichtlichkeit weggelassen worden.