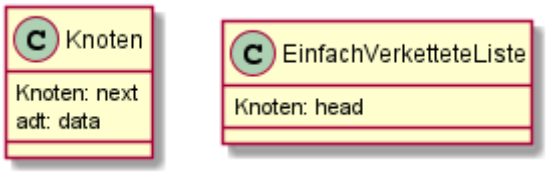
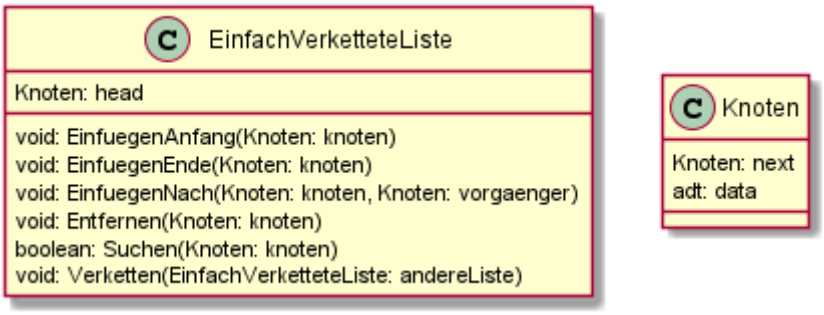
1. a.)

Eine einfach verkettete Liste besteht aus Knoten (engl. Node genannt) welche jeweils auf den nächsten Knoten zeigen. Die Knoten können also, im Gegensatz zu einem Array, über den gesamten Speicher verteilt sein. Die einzelnen Knoten enthalten jeweils ein Listenelement (beliebiger Datentyp) und einen Verweis auf den Knoten mit dem nächsten Listenelement. Eine einfach verkettete Liste ohne weitere Funktionen sieht also so aus:



(adt = abstrakter Datentyp)

Um die Arbeit mit solchen Listen zu vereinfachen werden meistens Funktionen (typischerweise verschieden Formen von Einfügen, Suchen, Entfernen und Verketten) zum Verwalten der Knoten hinzugefügt:



Falls es vermieden werden soll mit Knoten, anstelle der eigentlich relevanten Daten, zu arbeiten könnte eine Liste wie folgt aussehen:

In diesem Beispiel wird nur intern mit dem Knoten gearbeitet und sind somit nicht nach außen hin sichtbar.

1. b.)

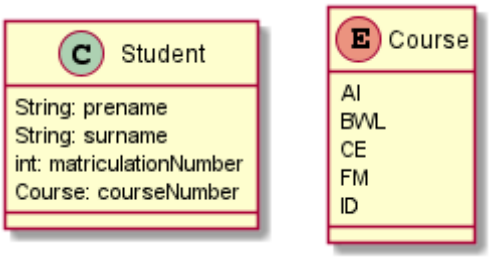
Um Studenten einfacher verwalten zu können wird hierfür ein neuer abstrakter Datentyp mit dem Namen Student erstellt. Der Student hat verschiedene Eigenschaften Vorname, Nachname, Matrikelnummer und Studiengang.

Wahl des Datentypen ist hier eindeutig, da String der einfachste Datentyp zum umfangreichen Verwalten von Zeichenketten ist, ein Char Array ist in allen Fällen außer dem reinen abspeichern der Zeichenkette schwieriger zu handhaben.

Die Matrikelnummer wird als „matriculationNumber“ vom Typ Integer verwaltet. Den Datentypen als Integer für die Matrikelnummer zu wählen, ergibt sich aus dem Namen selbst, es soll eine ganzzahlige Nummer sein.

Da Studiengänge fest definiert sind, bieten sich die Verwendung von Enums an, welche die Möglichkeit anbieten vorab definierte Konstanten festzulegen. Der Studiengang wird als „courseNumber“ als Enum vom Typ Course verwaltet. Die Werte des Course Enums sind AI (Angewandte Informatik), BWL (Betriebswirtschaftslehre), CE (Computer Engineering), FM (Facility Management) und ID (Industrial Design).

Der neue abstrakte Datentyp „Student“ kann auch in der oben gezeigten Liste verwaltet werden.



3.) Eine Methode die eine einfach verkettete Liste effizienter als ein Array bereitstellen kann, ist das Einfügen eines neues Datensatzes an einer bestimmten Stelle ohne den vorherigen Wert an dieser Stelle zu Überschreiben. Eine einfach verkettete Liste muss hierbei die Next-Komponente des Vorgängers auf den neuen einzufügenden Knoten setzten und den ursprünglichen Wert des nächsten Knoten des Vorgängers als Wert für Next im neuen Knoten setzten, das kann mit der Komplexität von O(1) also konstantem Aufwand implementiert werden. Eine selbe Funktionalität in einem Array müsste alle Elemente die nach der gewünschten Position kommen um eins nach hinten verschieben und könnte danach erst den neuen Wert einfügen, Komplexität von O(n). Besonders effizient ist diese Funktion falls an erster Stelle eingefügt werden soll.

Ein ähnliches Problem für das Array besteht falls ein Element entfernt werden soll, und nicht einfach nur überschrieben. Das Array müsste hierbei alle Elemente nach der gewünschten Position nach eins nach vorne verschieben, Komplexität von O(n). Bei einer einfach verketteten Liste ist die Komplexität ebenfalls O(n) da der Next-Wert des Vorgängers aktualisiert werden muss und es keine andere Möglichkeit gibt zum Vorgänger zu gelangen, als die gesamte Liste zu durchlaufen und den Vorgänger zu suchen. Bei doppelt verketteten Listen ist Komplexität hingegen O(1), da über den Vorgängerwert einfach zum Vorgänger gewechselt werden kann, hierbei ist allerdings zu beachten das auch der Nachfolger des zu entfernenden Elements angepasst werden muss, da der Vorgänger sich verändert.

Arrays sind einfach und doppelt Verketteten Listen im Zugriff auf ihre Elemente per Index überlegen. Auf Elemente in einem Array kann per Random Access in O(n) zugriffen werden. Beide verketteten Listen müssen jeweils bis zum gewünschten Index iterieren, O(n).

Da Arrays mit einer festen Größte erstellt werden, muss das gesamte Array in ein neues kopiert werden, falls das Array nun mehr Elemente als die ursprünglich definierte Größe zulässt aufnehmen soll. Bei verketteten Listen besteht dieses Problem nicht, es kann immer ein neues Element am Ende der Liste eingefügt werden.

5.) Seien Vertauschen und Vergleichen gleichwertige Operationen.

Selectionsort

Nachfolgend wird der Pseudocode des Selectionsort auf seine Komplexität analysiert.

var a = sequence
N = a.lenght // 1 OP
for i= 0 to N - 2 do // N - 1 OP
min = i // 1 OP
for j = I + 1 to N - 1 do // N/2 OP
if a[j] < a[min] // 1 OP
then min = j // 1 OP
t = a[min] // 1 OP
a[min] = a[i] // 1 OP
a[i] = t // 1 OP
end
end
	T(a) = 1 + (N - 1)(1 + N/2(1 + 1) + 1 + 1 + 1)
	T(a) = 1 + (N - 1)(4 + 2(N/2))
	T(a) = 1 + (N - 1)(4 + N)
	T(a) = ~~1~~ + ~~4~~N + N² - ~~4~~ – N
	T(a) = ~~N~~ + N² – ~~N~~
	T(a) = N²

Die Analyse zeigt, dass der Selectionsort eine Komplexitätsklasse von O(n²) besitzt.

Meine Implementation des Selectionsort:

```
int minimum; // 1 OP
for (int i = 0; i < list.size(); i++) { // N - 1 OP
    minimum = i; // 1 OP
    for (int e = i + 1; e < list.size(); e++) { // N/2 OP
        if (comparator.compare(list.get(e), list.get(minimum)) < 0) {//1 OP
            minimum = e; //1 OP
        }
    }
    swap(list, i, minimum); // 3 OP
}

protected void swap(IListable<T> list, int i, int j) {
    T memorizedObject = list.get(i); //1 OP
    list.set(i, list.get(j)); //1 OP
    list.set(j, memorizedObject); //1 OP
}
```

T(a) = 1 + (N - 1)(1 + N/2(1 + 1) + 3)

T(a) = 1 + (N - 1)(4 + 2(N/2))

T(a) = 1 + (N - 1)(4 + N)

T(a) = ~~1~~ + 4N + N² - 4 – N

T(a) = ~~N~~ + N² – ~~N~~

T(a) = N²

Auch meine Implementierung ist von der Komplexitätsklasse O(N²).

Unabhängig von den Aufgaben:

Bei der letzten Abgabe merkten Sie an, dass der Gradle-Wrapper-Skript für Linux nicht funktioniert.

Da ich leider kein Linux System habe, kann ich das schwer überprüfen. Ich füge trotzdem vorsichtshalber 2 Screenshots an, die zeigen das es zumindest unter Windows 10 funktioniert (1. Intellij Idea, 2. Windows CLI). Ich hoffe es funktioniert diesmal.