Projekt lineare Listen

Fachliche Inhalte

Idee der Datenstruktur lineare Listen, Anwendung der List-Klasse aus der Abiturbibliothek

Die Lernenden sollten das Vorläufer-Projekt „Wartezimmer“ zum Thema Schlange behandelt haben.

1. Abfahrtslauf

Aufgabe 1:

Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben.
Begründen Sie, in wieweit für diese Rangliste Konzepte der Struktur Schlange übernommen werden können, warum aber diese Struktur insgesamt nicht passend ist bzw. wenn sie für die Implementierung einer solchen Rangliste verwendet wird, einen hohen Verwaltungsaufwand bedingt.

Die Lernenden erkennen schnell, dass es sich um eine Verkettungs-Situation wie beim Beispiel Wartezimmer handelt. Jeder Abfahrer hat einen eindeutigen Nachfolger oder Vorgänger.

Die Struktur Schlange kann aber nicht einfach benutzt werden, weil die ankommenden Abfahrer an jeder Stelle der Struktur, nicht nur am Ende eingefügt werden können. (Dafür wäre mindestens eine Hilfsschlange nötig.)
Außerdem ist für dieses Einfügen sowie die Ausgabe in der Anzeige ein *Durchlauf* erforderlich, d. h., dass die Elemente der Struktur beginnend beim ersten nacheinander betrachtet werden. Bei einer Schlange kann dagegen nur das Kopfelement betrachtet werden, wenn dieses nicht zeitweise aus der Schlange in eine Hilfsschlange ausgelagert wird.

Aufgabe 2:

Geben Sie Methoden für eine Datenstruktur, die den Anforderungen an eine Rangliste aus Aufgabe 1 erfüllt.

Für den Durchlauf wäre ein *aktuelles Element hilfreich*, das die Position im Durchlauf anzeigt. Wenn diese grundlegende Idee erkannt ist, ergeben sich die Methoden fast zwangsläufig:

Das aktuelle Element muss an den Anfang der Liste und ein Element weiter gesetzt werden können. Es muss abgefragt und gelöscht werden können, außerdem muss vor (oder hinter) ihm eingefügt werden können. Schließlich muss abgefragt werden können, dass der Durchlauf beendet ist. Feinheiten wie das Anhängen und „zum Ende der Liste gehen“ aus dem Listenmodell der Abiturklassen müssen hier nicht erkannt werden.

Modifiziert aus den Materialien der Federführungsgruppe Informatik (September 2013)

Für die Rangliste könnte auch eine Methode gefordert werden, mit der ein Element direkt an der richtigen Stelle eingefügt wird (quasi eine Datenstruktur sortierte lineare Liste). Um diese zu realisieren ich jedoch dann ebenfalls eine Struktur hilfreich, die die oben genannten Anforderungen erfüllt, so dass die Verwendung der linearen Liste aus den Abiturvorgaben motiviert ist.

Nun kann die Dokumentation der Klasse *List* der Abiturklassen betrachtet werden.

Aufgabe 3:

Entwickeln Sie auf Basis der Klasse List ein Schema für den Durchlauf durch eine lineare Liste.

Man erhält folgendes Schema:

```
dieListe.toFirst();
while (dieListe.hasAccess())
{
        Elementtyp aktuellesElement = (Elementtyp) dieListe.getObject();
        aktuellesElement.macheEtwas();
        dieListe.next();
}
```

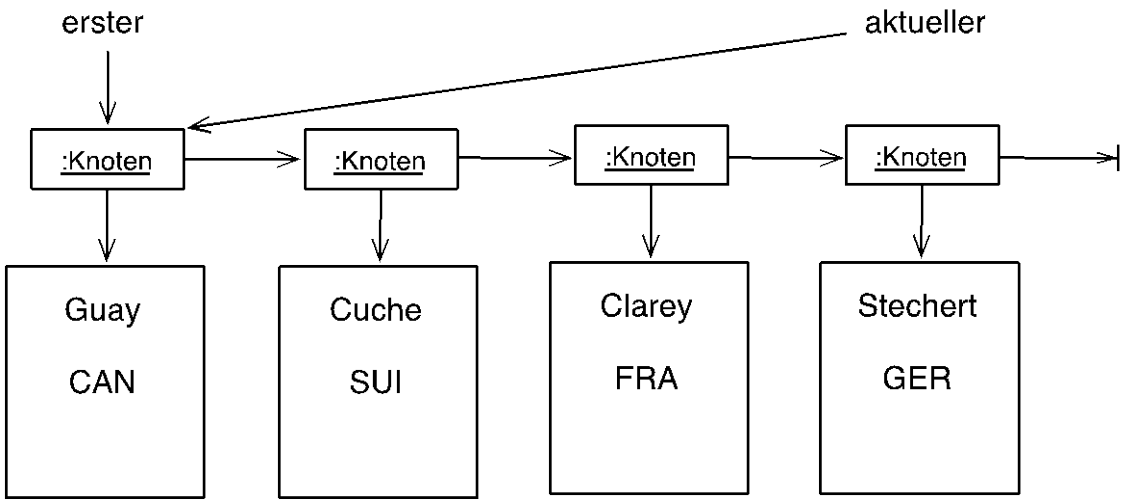
Aufgabe 4:

Entwickeln Sie einen Algorithmus für das Einfügen eines neuen Abfahrtsläufers in die Rangliste.

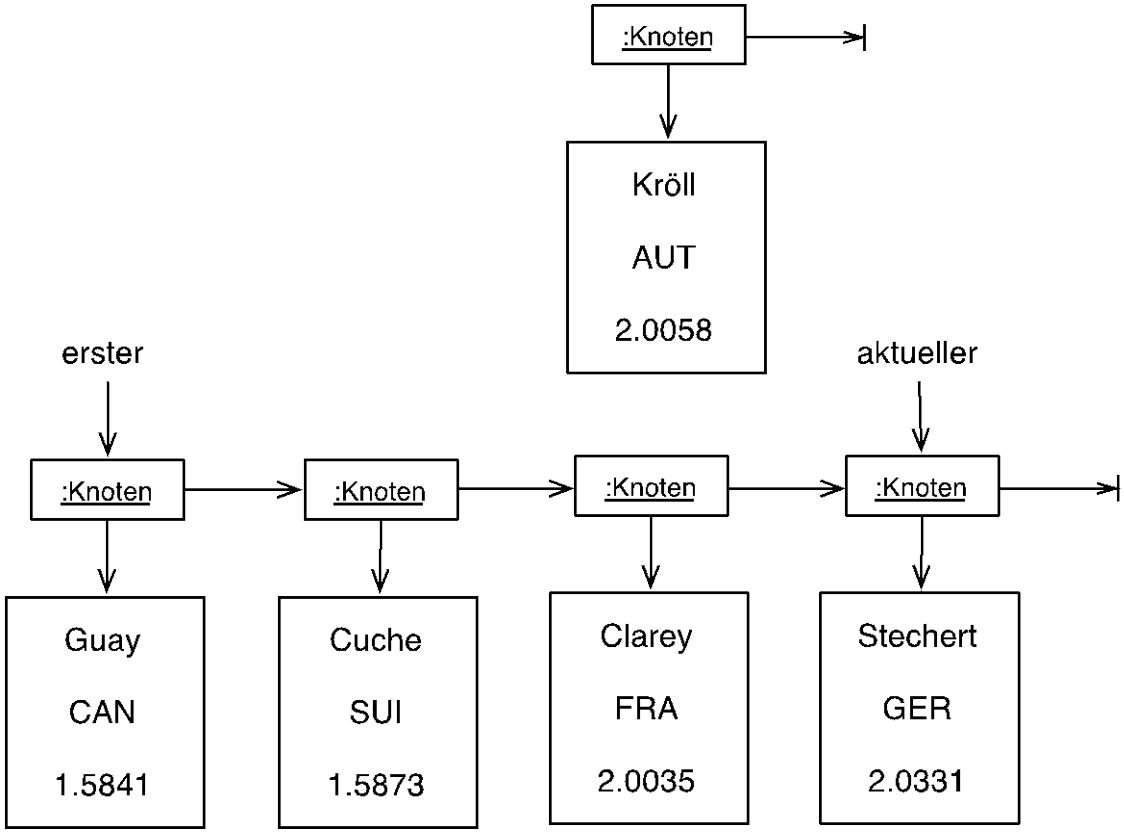
Die Liste muss solange durchlaufen werden, bis ein Läufer mit einer schlechteren Zeit als der des neuen gefunden ist. Dann wird der neue Läufer vor diesem Läufer eingefügt.
Wenn der neue Läufer die schlechteste Zeit hat, wird kein Platz zum Einfügen gefunden. In diesem Fall wird der neue Läufer angehängt.

Es empfiehlt sich, diese Situation mit der aus der Schlange bekannten graphischen Darstellung zu veranschaulichen.

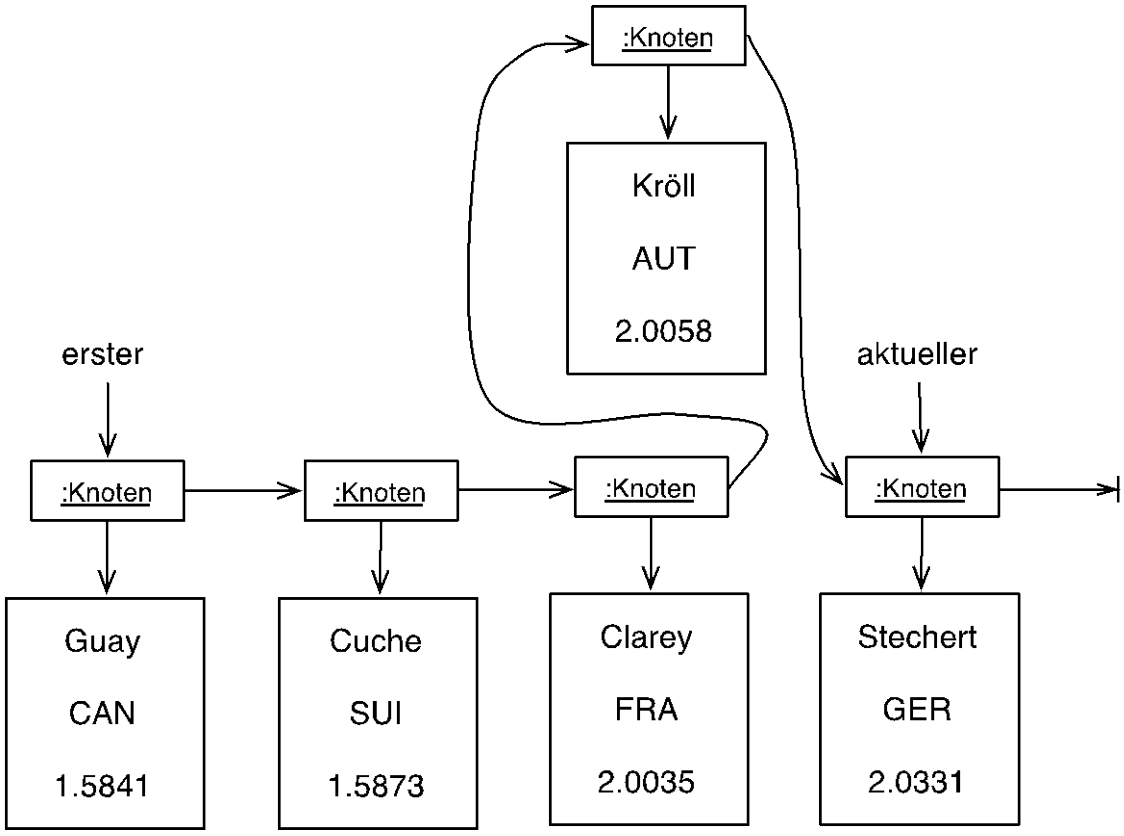
So kann die Liste nach vier Startern aussehen:


erster
aktueller
:Knoten
:Knoten
:Knoten
:Knoten
Guay
CAN
Cuche
SUI
Clarey
FRA
Stechert
GER

Nun wird ein neuer Abfahrer eingefügt. Solange seine Zeit schlechter ist als die des aktuellen Abfahrers rückt der Verweis auf den aktuellen ein Element weiter. Es ergibt sich z. B. folgende Situation:



Also muss der neue Abfahrer Kröll vor dem Abfahrer Stechert eingefügt werden. Daher werden die Verweise entsprechend „verbogen“.



Aufgabe 5:

Implementieren Sie die Rangliste eines Abfahrtslaufes auf der Basis eines Prototypen.

Es sollte aber darauf hingewiesen werden, dass es in der Klasse List nicht möglich ist, nach dem Durchlauf mit „insert“ am Ende einzufügen, sondern dass hier eine Fallunterscheidung nötig ist. Außerdem sollte thematisiert werden, warum es bei den beiden mit „und“ verknüpften Bedingungen der while-Schleife auf die Reihenfolge ankommt.

In der Musterlösung wird die Abfahrtsliste als Unterklasse der Klasse List realisiert.