Dynamische Datenstrukturen II: MAP

Warum „Map“?

Nachteile von Arrays/Arraylists:

* Suche Sequentiell
* Index numerisch
* Index beginnt mit 0
* Ohne Lücken aufsteigend

Blick zu INSY:

|  |
| --- |
| A |

A1 A2

A1….key

A2….daten

? index nicht-numerisch, „beliebiger“ Datentyp

|  |  |
| --- | --- |
| **Key** | **Value** |
| [blau] | x‘ 0000FF |
| [grün] | x‘ 00FF00 |
| [rot] | x‘ FF0000 |
| … |  |

Reihenfolge automatisch:

Strings

~~Primitive Datentypen~~

Int 🡪 Integer

………

Spielregeln für Map:

* Key muss eindeutig sein
* Value darf mehrmals vorkommen
* Value wird über den key gefunden/auf ihn zugegriffen
* Value 🡪 key? Value liefert mehrere keys! N keys!

Menge von Keys (KeySet)

* Reihenfolge? Undefiniert!

Kann definiert werden

Strings vergleichen .equals()

.compareTo()

Definieren hat eine Spielregel:

* Interface comparable
* Sortiert 🡪 TreeMap
* Unsortiert 🡪 HashMap

Aufgabe 03 „halbintelligenter Übersetzter“ (Mitlernend)

Eingabe: String „Java ist auch eine Insel“

Collections

List:

* ArrayList
* LinkedList

Map:

* HashMap
* TreeMap

Set:

* HashSet
* TreeSet

Hash 🡪 geordnet

Tree 🡪 ungeordnet

Tree: Binärer-Baum

b < a

c > a

d < b

e > b

Hash: 1. Ziel: Speicherort beim sperichern finden

2. Ziel: beim Lesen Speicher finden

Ist eine Arraylist, die sich dynamisch vergrößert falls eine gewisse Prozentangabe der Füllung überschritten wurde. Ansonsten speichert man Pro ArrayList-Slot eine Linked List ein…damit das nicht passiert sollte man eine ausreichende Kapazität an ArrayList-Slots angeben.