Heute

- TreeMap
- Löschen aus Collections
- Algorithmisches Denken
- Sortieren

Organisatorisches

- Zwischentest am 12.12. ab 10h im Hilbertraum
- VO+UE am 12.12. entfallen
- Zusätzliche VO am 10.12. von 9:15-10:45 (Fragestunde)
- Alte Prüfungsangaben unter

  infotech.unileoben.ac.at/lehre/lva.htm

TreeMap

- implementiert Interface *Map*
- Das *keySet* ist aufsteigend sortiert.
- Dazu muss die Klasse der keySet-Elemente natürlich das Interface *Comparable* implementieren.

Schleifen und ArrayLists

- Objekte in ArrayList können innerhalb eines Schleifendurchlaufs geändert werden.

- ***Problemlos:*** aktuelle Elemente ändern oder ersetzen
- ***Problematisch:*** aktuelle Elemente löschen oder Elemente hinzufügen

Schleifen und Collections

Löschen/Hinzufügen von Elementen während Durchlaufs einer Collection:

- ***for-each:*** geht nicht (Exception!)
- ***for****-Schleife:* geht, aber muss Indizes entsprechend ändern (oder Elemente in umgekehrter Reihenfolge durchlaufen)

Löschen aus Collections

Empfohlene Methode:

removeAll:

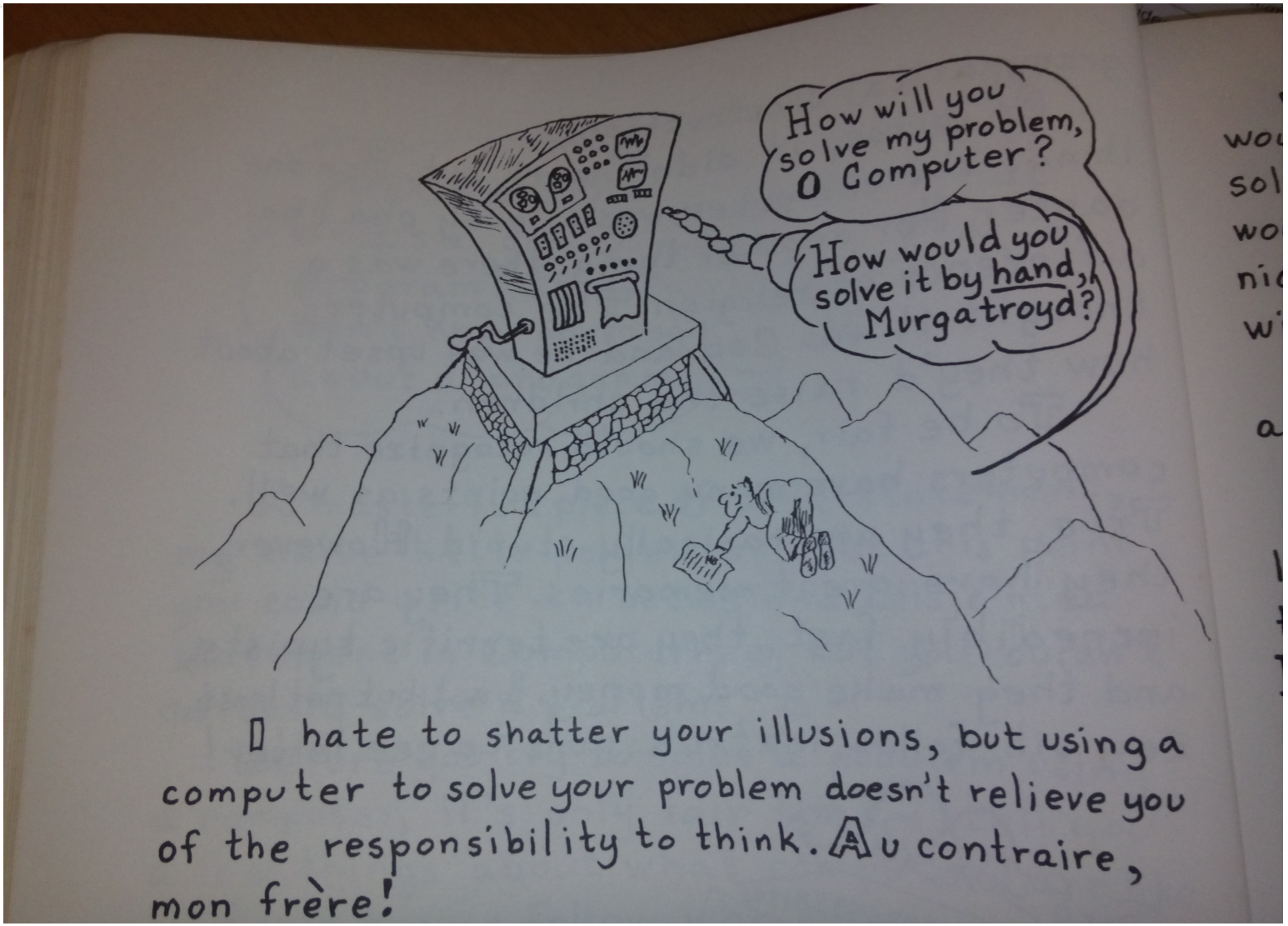
- speichere zu löschende Elemente in einer Collection/List etc.
- mache anschließend *removeAll*

NB: Dazu muss equals überschrieben sein!

Alternative:

über Iterator (nicht offizieller Teil des Stoffs)

Algorithmisches Denken



Algorithmisches Denken

- Wie würde man gegebenes Problem per Hand lösen?
- Abstraktion, Übersetzung in (Pseudo-)Code.

Modularisierung:

- Aufteilung in kleinere, einfache Probleme
- Lösung für Teilprobleme in Hilfsmethoden auslagern

Algorithmisches Denken

Basisoperationen für Listen:

- Hinzufügen (add)
- Löschen (remove)
- Schleife

Basistechiken für Listen:

- Min./Max-Suche
- Suchen von Elementen mit bestimmten Eigenschaften
- Sortieren

Einfache Sortieralgorithmen

Selectionsort:

Für j=1,...,#Elemente-1:

Suche das j-größte Element und vertausche es mit Element j.

Einfache Sortieralgorithmen

Insertionsort:

Für j=1,...,#Elemente:

Füge das j-te Element sortiert unter die ersten j-1 Elemente ein.

Einfache Sortieralgorithmen

Bubblesort:

Vertausche solange benachbarte Elemente in falscher Reihenfolge, bis sortiert.

Selectionsort

```
double[] a;  // zu sortierendes Array
for(int i=0; i<a.length-1; i++)
  {  // suche kleinstes Element von a[i] bis a[a.length-1]
     int min = i;
     for(int j=i+1; j<a.length; j++)
     {  if( a[j] < a[min])
           min = j;
     }
     // vertausche kleinstes Element a[min] mit a[i]
     double hilf = a[min];
     a[min] = a[i];
     a[i] = hilf;
  }
```

Bubblesort

```
double[] a;  // zu sortierendes Array
do{
    boolean veraendert = false;
    for (i=0; i<a.length-1; i++){
        if (a[i] > a[i+1]){
            // vertausche a[i] mit a[i+1]
            double hilf = a[i];
            a[i] = a[i+1];
            a[i+1] = hilf;
            veraendert = true;
        }
    }
  }while(veraendert)
```