# ALP III: Datenstrukturen und Datenabstraktion
# 4. Aufgabenblatt
# Übungsgruppe 1.8: Marcel Erhardt

Tobias Lohse/ Marvin Kleinert/ Anton Drewing

14.11.2014

## Aufgabe 1

- Region mid = new State();
  Zuweisungen von Variablen mit konkreterem dynamischen Typ zu Variablen mit allgemeinerem statischen Datentyp sind zulässig; in diesem Fall ist eine Region immer auch ein State

- State md = new Maryland();
  Erklärung siehe oben

- Obj obj = new Place();
  Erklärung siehe oben; Object als allgemeinster Typ

- Place usa = new Region();
  Erklärung siehe oben

- md.printMe();
  Ausgabe: „Mary.“
  Grund: Da in md ein Objekt vom Typ Maryland gespeichert ist, wird nach dem Prinzip der Methodenüberschreibung die Methode des dynamischen Typs verwendet, auch wenn md den statischen Typ State besitzt.

- mid.PrintMe();
  Ausgabe: „Stan.“
  Begründung: Die Variable mid hat den dynamischeb Typ State, sodass gemäß der Methodenüberschreibung eine printMe-Methode der Klasse State verwendet wird;

- xx = (What) obj;
  Die Variable xx ist vom Typ What und die zugewiesene Variable obj wird mit What gecastet.

- ((Place) xx).printMe();
  Ausgabe: „Plato.“
  Begründung: Die Variable xx wird mit Place gecastet; es wird die printMe-Methode des statischen Datentyps Place verwendet, da der dynamische Typ von xx als Interface What keine printMe-Methode besitzt.

- obj = md;
  Es gibt kein Problem bei der Zuweisung, da der statische Typ What von obj allgemeiner ist als der der Typ des in md gespeicherten Objektes Maryland.

- ((Maryland) obj).printMe();
  Ausgabe: „Mary.“

Begründung: Die Variable obj wird mit Maryland gecastet, damit überhaupt erst eine printMe-Methode angewendet werden kann; obj hat den dynamische Datentyp Maryland, Methodenüberschreibung siehe oben.

- ((Maryland) obj.printMe(1);
  Ausgabe: „Regina. 1“
  Begründung: Casting siehe oben; weil ein Parameter übergeben wird, wird nach dem Prinzip der Methodenüberladung die printMe-Methode der nächsten Oberklasse verwendet, die einen Parameter erwartet.

- obj = usa;
  Der Variablen obj mit statischem Typ What kann eine Variable mit dem konkreteren dynamischen Typ Region zugewiesen werden.

- ((Place) obj).printMe();
  Ausgabe: „Regina.“
  Begründung: Variable obj mit dynamischem Typ Region; Methodenüberschreibung.

- usa = mid;
  Der Variablen usa statischem Typ Place kann eine Variable mit dem konkreteren dynamischen Typ Maryland zugewiesen werden.

- ((Place) usa).printMe();
  Ausgabe: „Mary.“
  Begründung: Die Variable usa hat den dynamischen Datentypen Maryland; Methodenüberschreibung; Typcasting an diser Stelle eigentlich unnötig.

## Aufgabe 2

aenderSchluessel(i,s):

- ändere den Schlüssel des Eintrags mit dem Feldindex i in s
- vergleiche s mit dem Schlüssel des Vaterknotens v
  - wenn $s < v$ ist, bubble up
  - andernfalls vergleiche s mit den Schlüsseln der beiden Kinder
    - wenn $s < k1$ und $s < k2$: das Element befindet sich an der richtigen Stelle
    - andernfalls bubble down

Laufzeit (schlechteseter Fall):

$$T(n) = 2 * log(n) = O(log(n))$$

kleiner(s):

- erzeuge eine Ergebnisliste smallerS
- traversiere den Baum preorder bei der Wurzel beginnend und vergleiche s mit dem jeweiligen Schlüssel $s_i$
- wenn $s_i < s$ ist, speichere $s_i$ in der Ergebnisliste, sonst beende die Traversierung des Teilbaumes mit $s_i$ als Wurzel

Laufzeit(schlechtester Fall):

$$T(k) = \underbrace{k}_{\text{Knoten < s}} + \underbrace{(k+1)}_{\text{Vergleich mit max. (k+1) Kindern}} = 2k + 1 = O(k)$$