Interna von Java

Speicherbereiche

Heap: enthält Objekte und Klassen

- `new` allokiert Speicher für Objekt
- Garbage-Collection gibt unnötigen Speicher zurück
- wächst bei Bedarf (solange Speicher verfügbar)

Stack: enthält Daten aufgerufener Methoden

- *Stackframe* bei Methodenaufruf angelegt (push)
- Stackframe bei Rückkehr aus Methode abgebaut (pop)
- hat vorgegebene Maximalgröße

Objekt, Klasse, Stackframe = Datenblock unterteilt in Felder

Objekt (Aufbau des Datenblocks)

- 1. Feld: Referenz auf die Klasse, die Objekt erzeugt hat
- Anzahl und Art weiterer Felder hängen von Klasse ab
- jedes weitere Feld enthält Inhalt einer Objektvariablen (Referenz in den Heap oder Daten eines primitiven Typs)
- intern: *Offset* von Objektanfang statt Variablennamen

Klasse (Aufbau des Datenblocks)

- Anfang: Methodentabelle (VFT = Virtual Function Table)
- jede nicht-statische Methode hat intern *Index* statt Namen
- VFT$[i]$ enthält Referenz auf Code der Methode i
- dynamic binding = Methodenaufruf über VFT$[i]$
- static binding (statische Methode) = direkter Aufruf
- Code der Methoden und statische Variablen stehen irgendwo in Klasse (z.B. durch Offset von Anfang festgelegt)

Stackframe (Aufbau des Datenblocks)

- Referenz auf Stelle nach dem Aufruf (in einer Klasse)
- Referenz auf Stackframe der aufrufenden Methode
- formale Parameter
- lokale Variablen

Globale Daten:

- Beginn des aktuellen Stackframes (A)
- Ende des aktuellen Stackframes (B)

Methodenaufruf

1. Referenz auf Stelle nach dem Aufruf auf Stack legen
2. A (= Stackframe des Aufrufers) auf Stack legen
3. aktuelle Parameter (= Argumente) auf Stack legen
4. $A = B + 1$ (Beginn des aktuellen Stackframes setzen)
5. $B = A +$ Stackframe-Größe
   (hängt von Parametern und lokalen Variablen ab)
6. zum Code der aufgerufenen Methode springen

Rückkehr aus Methode (return)

1. Rückgabewert in ein Register legen
2. $B = A - 1$
3. $A =$ Stackframe des Aufrufers (in Stackframe)
4. zurück zum Code des Aufrufers springen (in Stackframe)

Vererbung

- zusätzliche Objektvariablen an Ende des Objekts anfügen (neue Offsets vergeben)
- VFT der Unterklasse: Kopie jener der Oberklasse, wobei
  - zusätzliche Methoden hinten angehängt (neue Indizes)
  - Einträge überschriebener Methoden überschrieben
- ererbte Methoden funktionieren automatisch richtig

Interface

- pro implementiertem Interface besitzt Klasse eigene VFT
- unterscheidet sich von normalem VFT der Klasse nur durch Anzahl und Reihenfolge (Indizes) der Methoden
- Klasse besitzt Array dieser Interfaces (Index $\approx$ Interface)
- wenn deklarierter Typ einer Variablen ein Interface: dynamisches Binden über VFT dieses Interfaces