FOP Reference Sheet

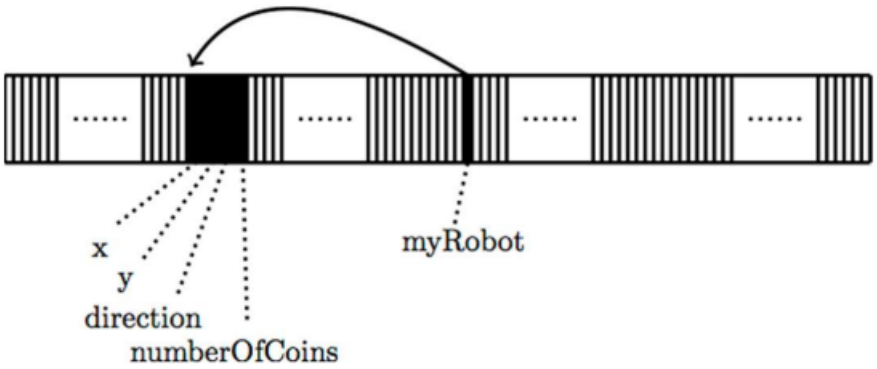
Jonas Milkovits

Last Edited: 26. März 2020

Inhaltsverzeichnis

1 Computerspeicher

Unsere Vorstellung	▷ großes Feld aus Maschinenwörtern mit eindeutiger Adresse
Erzeugung eines neuen Objekts	▷ Reservierung von ungenutztem Speicher in ausreichender Größe
Referenz	▷ Name der Variable, die die Anfangsadresse des Objekts speichert ▷ Kann auch an komplett anderer Stelle als das Objekt gespeichert sein
Speicherort primitiver Datentypen	▷ Name verweist tatsächlich auf Speicherstelle, an der Wert abgespeichet wird
Prozessablauf	▷ Program Counter enthält Adresse der nächsten Anweisung ⇒ Zählt nach jeder Anwendung hoch und verweist auf nächsten Speicher ▷ CPU verarbeitet parallel die momentane Anweisung aus Program Counter
Methodenausführung	▷ Einrichtung einer Variable `StackPointer` bei Programmstart ▷ StackPointer enthält die Adresse des `Call-Stacks` ▷ Bei Methodenaufruf wird im Speicher Platz reserviert, genannt `Frame` ▷ `Frame` wird dann auf dem Call-Stack abgelegt ▷ Der `StackPointer` wird dann mit der Adresse des neuen`Frames` überschrieben ▷ Methodenaufruf vorbei: Frame wird wieder vom `Call-Stack` genommen ▷ `StackPointer` wird auf Adresse des vorherigen `Frames` gesetzt



2 Datenstrukturen

Array	▷ Verwendet zum Speichern von mehreren Variablen des selben Typs ▷ Erzeugung: `int[] test = new int[n];` ▷ `n` gibt in diesem Fall die feste Anzahl der speicherbaren Variablen an ▷ Natürlich auch Arrays von Objekten möglich ▷ Zugriff auf Variablen: `test[0]` für ersten Wert (Index) ▷ Zugriff auf Länge: `test.length`
	▷
	▷
	▷
	▷
	▷
	▷
	▷

3 Datentypen

Konstanten	▷ Variable/Referenz wird dadurch unveränderbar ▷ z.B.: `final myClass ABC = new myClass();` ⇒ Referenz zwar nicht veränderbar, Objekt aber schon ▷ `Integer.MAX_VALUE / Integer.MIN_VALUE` ▷ Unendlich: `Double.POSITIVE_INFINITY / Double.NEGATIVE_INFINITY`
Primitive Dateitypen	▷ Ganze Zahlen: byte → short → int → long ▷ Gebrochene Zahlen: float → double ▷ Logik: boolean ▷ Zeichen: char
Literale	▷ wörtlich hingeschriebene Werte eines Datentyps ▷ Zahlen standardmäßig int, falls `long` gewünscht: `123L oder 123l` ▷ Bei gebrochenen double, falls `float` gewünscht: `12.3F oder 12.3f` ▷ null: Nutzung für Referenzen → verweist auf nichts
Boolean	▷ nur `true` und `false` ▷ Negation `!a` ▷ Logisches Und: `a && b` ▷ Logisches Oder: `a || b` (inklusiv) ▷ Gleichheit: `a == b`
Zeichentyp char	▷ z.B.: `char c = ´a´;` ▷ Interne Kodierung als Unicode ▷ ´\t´ Horizontaler Tab ▷ ´\b´ Backspace ▷ ´\n´ Neue Zeile ▷ Auch Darstellung im Hexacode (´\u039A´)
Enumeration	▷ Zusammenfassung mehrerer Konstanten (feste Anzahl) ▷ Erzeugung meist in eigener .java Datei ▷ `enum MyDirection {DOWN, RIGHT}` ▷ Keine Objekterzeugung von Enumeration möglich ▷ Abspeichern in Variable des Enum-Types ist jedoch möglich ▷ `MyDirection dir = MyDirection.DOWN;`

4 Klassen

Erzeugung	▷ meist in seperater .java Datei ▷ `public class MyClass {}`
Attribute	▷ Eigenschaften der Objekte/Klassen ▷ z.B.: `private int x;` (Objekteigenschaft) ▷ z.B.: `private static int x;` (Klasseneigenschaft)
	▷
	▷
	▷
	▷
	▷

5 Konversionen

Implizit	▷ Immer möglich, wenn kein Informationsverlust entstehen kann ▷ z.B.: kleinerer Datentyp in größeren
Explizit	▷ Meist Informationsverlust ▷ Durchführung durch Angabe des Datentyps in Klammern davor ▷ z.B.: `int i = (int)testDouble;`

6 Methoden

Methodenkopf	▷ Modifier Rückgabewert Methodenname (Parameter) {Anweisung} ▷ z.B.: `public void setX (int x) {this.x = x;}` (Objektmethode) ▷ z.B.: `public static void setY (int y) {this.y = y;}` (Klassenmethode) ▷ `this.x` steht hier für das Objektattribut und nicht den Parameter
Ausführung	▷ Objektmethoden: `myObject.setX(2);` ▷ Klassenmethoden: `MyClass.setY(2);`
return	▷ Wird für Rückgabe bei Methoden mit Rückgabewert benötigt
	▷
	▷
	▷
	▷
	▷

7 Programme und Prozesse

Quelltest	▷ z.B. selbst geschriebener Java-Code
Java-Bytecode	▷ Wird durch Übersetzung des Java-Quelltextes erzeugt
Programm	▷ Sequenz von Informationen
Aufruf eines Programms	▷ Starten eines Prozesses, der die Anweisungen des Programmes abarbeitet
Prozesse	▷ CPU besteht aus mehreren Prozessorkernen ▷ Mehrere Prozesse laufen dementsprechend parallel ▷ Allerdings bearbeitet jeder Kern nur einen Prozess gleichzeitig (sehr kurz) ⇒ Illusion von Multitasking
	▷

8 Schleifen und if

while-Schleife	▷ `while (Bedingung) {Anweisung;}` ▷ Schleife wird ausgeführt, solange die Bedingung wahr ist ▷ {} kann bei einzelner Anweisung auch weggelassen werden
do-while-Schleife	▷ `do {Anweisung;} while (Bedingung);` ▷ Anweisungsblock wird immer mindestens einmal ausgeführt
for-Schleife	▷ `for (Anweisung davor; Bedingung; Anweisung danach) {Anweisung}` ▷ z.B.: `for (int i = 0; i < 10; i++) {...}` ⇒ Zehnmalige Ausführung der Anweisung ▷ Kurzform: `for (Position p : positions) {}`
if-Anweisung	▷ `if (Bedingung) {...}` ▷ Führt den Code in der Anweisung nur aus, falls die Bedingung erfüllt ist ▷ `if (Bedingung) {} else {}` ▷ Code, der ausgeführt wird, falls Bedingung nicht erfüllt ist

9 Syntax

Keywords	▷ Können nur an bestimmten Stellen im Code stehen ▷ z.B. `class, import, public, while,..`
Identifier	▷ Namen für Klassen, Variablen, Methoden,.. ▷ Erstes Zeichen darf keine Ziffer sein ▷ Keine Keywords als Identifier ▷ Identifier sind case-sensitive
Konventionen	▷ Variablen / Methoden beginnen mit Kleinbuchstaben (`testInt`) ▷ Klassen beginnen mit Großbuchstaben (`testClass`) ▷ Wortanfänge im Inneren mit Großbuchstaben ▷ Konstanten bestehen aus `_` und Großbuchstaben (`CENTS_PER_EURO`)
Kommentare	▷ `//` Einzelne Zeile ▷ `/*...*/` Mehrere Zeilen ▷ `/**...*/` Erzeugung von Javadoc