

FOP Reference Sheet

Jonas Milkovits

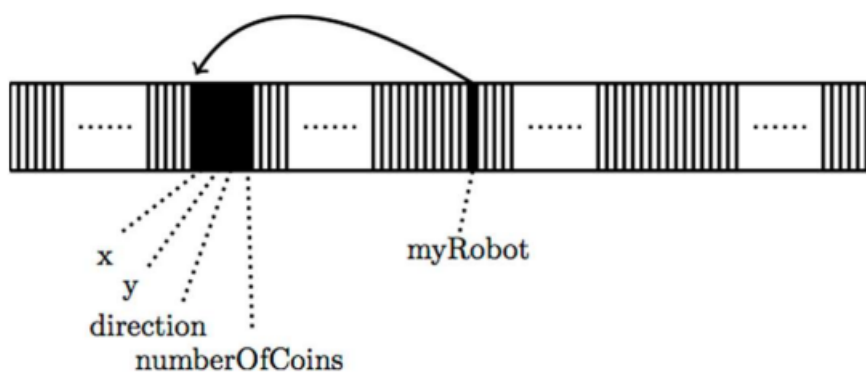
Last Edited: 27. März 2020

Inhaltsverzeichnis

1	Computerspeicher	1
2	Datenstrukturen	1
3	Datentypen	2
4	Interfaces	2
5	Klassen	3
6	Konversionen	3
7	Methoden	3
8	Packages und Zugriffsrechte	3
9	Programme und Prozesse	4
10	Schleifen und if	4
11	Syntax	4
12	Vererbung	5

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 abgespeichert 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
Methodentabelle	▷ Enthält bei Objekt die Anfangsadressen der verfügbaren Methoden



2 Datenstrukturen

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

3 Datentypen

Konstanten	<ul style="list-style-type: none"> ▷ Variable/Referenz wird dadurch unveränderbar ▷ z.B.: <code>final myClass ABC = new myClass();</code> ⇒ Referenz zwar nicht veränderbar, Objekt aber schon ▷ <code>Integer.MAX_VALUE</code> / <code>Integer.MIN_VALUE</code> ▷ Unendlich: <code>Double.POSITIVE_INFINITY</code> / <code>Double.NEGATIVE_INFINITY</code>
Primitive Datentypen	<ul style="list-style-type: none"> ▷ Ganze Zahlen: <code>byte</code> → <code>short</code> → <code>int</code> → <code>long</code> ▷ Gebrochene Zahlen: <code>float</code> → <code>double</code> ▷ Logik: <code>boolean</code> ▷ Zeichen: <code>char</code>
Literale	<ul style="list-style-type: none"> ▷ wörtlich hingeschriebene Werte eines Datentyps ▷ Zahlen standardmäßig <code>int</code>, falls <code>long</code> gewünscht: <code>123L</code> oder <code>123l</code> ▷ Bei gebrochenen <code>double</code>, falls <code>float</code> gewünscht: <code>12.3F</code> oder <code>12.3f</code> ▷ <code>null</code>: Nutzung für Referenzen → verweist auf nichts
Boolean	<ul style="list-style-type: none"> ▷ nur <code>true</code> und <code>false</code> ▷ Negation <code>!a</code> ▷ Logisches Und: <code>a && b</code> ▷ Logisches Oder: <code>a b</code> (inklusive) ▷ Gleichheit: <code>a == b</code>
Zeichentyp char	<ul style="list-style-type: none"> ▷ z.B.: <code>char c = 'a';</code> ▷ Interne Kodierung als Unicode ▷ <code>'\t'</code> Horizontaler Tab ▷ <code>'\b'</code> Backspace ▷ <code>'\n'</code> Neue Zeile ▷ Auch Darstellung im Hexacode (<code>'\u0041'</code>)
Enumeration	<ul style="list-style-type: none"> ▷ Zusammenfassung mehrerer Konstanten (feste Anzahl) ▷ Erzeugung meist in eigener .java Datei ▷ <code>enum MyDirection {DOWN, RIGHT}</code> ▷ Keine Objekterzeugung von Enumeration möglich ▷ Abspeichern in Variable des Enum-Typs ist jedoch möglich ▷ <code>MyDirection dir = MyDirection.DOWN;</code>

4 Interfaces

Erzeugung	<ul style="list-style-type: none"> ▷ Meist in eigener Datei ▷ <code>public interface MyInterface {...}</code> ▷ Alle Methoden und das Interface müssen <code>public</code> sein
Methoden	<ul style="list-style-type: none"> ▷ Werden hier nicht implementiert, sondern nur definiert ▷ <code>public</code> kann weggelassen werden, da ohnehin notwendig
Verwendung	<ul style="list-style-type: none"> ▷ <code>implements MyInterface</code> nach Klassenname
	▷
	▷
	▷
	▷
	▷

5 Klassen

Erzeugung	<ul style="list-style-type: none">▷ meist in separater .java Datei▷ <code>public class MyClass {}</code>
Attribute	<ul style="list-style-type: none">▷ Eigenschaften der Objekte/Klassen▷ z.B.: <code>private int x;</code> (Objekteigenschaft)▷ z.B.: <code>private static int x;</code> (Klasseneigenschaft)
Konstruktor	<ul style="list-style-type: none">▷ Wird zur Erzeugung von neuen Objekten einer Klasse verwendet▷ Methode mit selben Namen wie Klasse und ohne Rückgabotyp▷ z.B.: <code>public MyClass (int x, int y) {this.x = x; this.y = y;}</code>▷ Erzeugung eines neuen Objekts: <code>MyClass test = new MyClass(2,4);</code>▷ Falls kein Konstruktor angegeben wird → Standardkonstruktor

6 Konversionen

Implizit	<ul style="list-style-type: none">▷ Immer möglich, wenn kein Informationsverlust entstehen kann▷ z.B.: kleinerer Datentyp in größeren
Explizit	<ul style="list-style-type: none">▷ Meist Informationsverlust▷ Durchführung durch Angabe des Datentyps in Klammern davor▷ z.B.: <code>int i = (int)testDouble;</code>

7 Methoden

Methodenkopf	<ul style="list-style-type: none">▷ Modifier Rückgabewert Methodenname (Parameter) {Anweisung}▷ z.B.: <code>public void setX (int x) {this.x = x;}</code> (Objektmethode)▷ z.B.: <code>public static void setY (int y) {this.y = y;}</code> (Klassenmethode)▷ <code>this.x</code> steht hier für das Objektattribut und nicht den Parameter
Ausführung	<ul style="list-style-type: none">▷ Objektmethode: <code>myObject.setX(2);</code>▷ Klassenmethode: <code>MyClass.setY(2);</code>
return	<ul style="list-style-type: none">▷ Wird für Rückgabe bei Methoden mit Rückgabewert benötigt

8 Packages und Zugriffsrechte

Packages	<ul style="list-style-type: none">▷ Zusammenfassung von mehreren Dateien▷ <code>import package.*;</code>▷ <code>*</code> steht für alle Definitionen aus <code>package</code>▷ Import-Anweisungen müssen immer am Anfang des Quelltextes stehen
Zugriffsrechte	<ul style="list-style-type: none">▷ Klassen/Enum: nur <code>public</code> oder nichts<ul style="list-style-type: none">⇒ Nur eine Klasse darf <code>public</code> sein (Damit auch Dateiname)▷ <code>private</code>: Zugriff innerhalb der Klasse▷ Keine Angabe: <code>private</code> + im Package▷ <code>protected</code>: Keine Angabe + in allen Subklassen▷ <code>public</code>: <code>protected</code> + an jeder Import-Stelle

9 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
	▷

10 Schleifen und if

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

11 Syntax

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

12 Vererbung

Zweck	▷ Weitergabe von allen Methoden und Attributen
Verwendung	▷ <code>public class MySubClass extends MyClass {}</code>
Konstruktor	▷ Aufruf des Konstruktors der Superklasse mithilfe von <code>super(Parameter);</code> ▷ Dieser Aufruf erfolgt im Konstruktor der Subklasse ▷ z.B.: <code>public MySubClass (int x) { super(x);<v></code>
Overwrite	▷ Methoden in Subklassen können auch neu geschrieben werden ⇒ Die Implementation der Superklasse wird sozusagen überschrieben ▷ Selber Name und Parameterliste notwendig
Overload	▷ Methoden mit selbem Bezeichner, aber unterschiedlicher Parameterliste ▷ Die Methode wird überladen ▷ Konstruktoren kann man auch überladen ⇒ Für manche Werte werden dann Standardwerte gesetzt ⇒ Anderer Konstruktor auch in Konstruktor aufrufbar (<code>this(1);</code>)
	▷
	▷
	▷