Freie Universität Berlin

Prof. Dr. Claudia Müller-Birn, Barry Linnert

Objektorientierte Programmierung, SoSe 17

Übung 06

TutorIn: Thierry Meurers
Tutorium 10

Stefaan Hessmann, Jaap Pedersen, Mark Niehues

20. Juni 2017

---

1 Aufgabe 1

a) $2^{\lfloor \log_2 n \rfloor} \leq 2^{\log_2 n} = n \implies 2^{\lfloor \log_2 n \rfloor} \in \mathcal{O}(n)$

b) $3^{\lfloor \log_2 n \rfloor} = (2^{\log_2 3})^{\lfloor \log_2 n \rfloor} \leq (2^{\log_2 3})^{\log_2 n} = n^{\log_2 3} \leq n^2 \implies 3^{\lfloor \log_2 n \rfloor} \in \mathcal{O}(n^2)$

c)

$$\begin{aligned} 2^{2^{\lfloor \log_2 n \cdot \log_2 n \rfloor}} &\leq 2^{2^{\log_2 n \cdot \log_2 n + 1}} \\ &= 2^{2^{\log_2 n \cdot \log_2 n} \cdot 2} \\ &= 2^{n^{\log_2 n} \cdot 2} \\ &= 4^{n^{\log_2 n}} \end{aligned}$$

Mit $4^{n^{\log_2 n}} \in \Theta(4^{n^{\log_n n}}) = \Theta(4^n) \implies 2^{2^{\lfloor \log_2 n \cdot \log_2 n \rfloor}} \in \mathcal{O}(4^n)$

Im letzten Schritt wurde folgende Regel angewandt: Für alle $a > 1$ und $b > 1$ gilt $\log_a n \in \Theta(\log_b n)$. Wir haben also benutzt, dass $n > 1$ ist. Quelle: Script Alp3 des letzten Jahres.

2 Aufgabe 2

a) Intellij, wegen Bedienungskonzept bereits aus pycharm bekannt.

b) Hello World Programm:

Listing 1: Hello World

```
0 public class HelloWorld{
1     public static void main(String [] args){
2         System.out.println("Hello World");
3     }
4 }
```

3 Aufgabe 3

a) x = 0: Integer

b) x = false: Boolean

c) x = true : olean

d) x = 7: Integer

e) x = 1.0: Double

f) x = 3: Integer

g) x = false: Boolean

h) x = 2: Integer

i) x = -6: Integer

j) x = 8: Integer

k) x = 24: Integer

l) x = Infinity: Double

4 Aufgabe 4

```
1 int a = 2, b = 1, c = 0;
2 c = a-- + b++; // c = 3, a = 1, b = 2
3 c = c++; // c = 4
4 c = --a + b++; // c = 2, a = 0, b = 3
```

Nach Ausführung also: $a = 0, b = 3, c = 2$.

5 Aufgabe 5

a) Debuggin Möglichkeiten:

1. Debugging mit Breakpoints
2. Variablien Viewer während des Debuggin und Expression Evaluation
3. Werte während des Debugging-Durchlaufs verändern
4. Klassen während eines Durchganges neu laden (HotSwap)

b) Mögliche Fehler die übersehen werden

1. Logik-Fehler
2. Unbeachtete Plattformabhängigkeiten (JRE Version)
3. Nicht abgefangene falsche Eingaben

c) Weiter Möglichkeiten um Fehler zu vermeiden:

1. Assertions im Code um Falsche Nutzereingaben zu erkennen
2. Test-Driven-Developement: Zunächst Testaufrufe Programmieren, die einzelene Code Fragmente auf das gewünschte Verhalten hin testen
3. Korrektheit formell testen (z.B. Hoare-Kalkül)

4. Testen, testen, testen..