

A1

 Jannis
 Liebscher
 301645

a)

$88/2 = 44$	R	0
$44/2 = 22$	R	0
$22/2 = 11$	R	0
$11/2 = 5$	R	1
$5/2 = 2$	R	1
$2/2 = 1$	R	0
$1/2 = 0$	R	1

0 b 1 0 1 1 0 0 0

*

<u>0 0 1 0 1 1</u>	<u>0 0 0</u>	
7	3	0

0 1 3 0

*

<u>0 1 0 1 1</u>	<u>0 0 0</u>
5	8

0 X 5 8

*

<u>8 8 . 0</u>

b)

$$-88 = -(1011000)_2$$

\downarrow invertieren

$$0100111 \quad 7\text{-er K}$$

$\downarrow +1$

$$\underline{0101000} \quad 2\text{-er Komplement}$$

c) 707,107

$$\begin{aligned}
 &= 7 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 7 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \\
 &= 4 + 0 + 7 + 0,5 + 0 + 0,125 \\
 &= \underline{\underline{5,625}}
 \end{aligned}$$

d) 061 → (101001)₂

OK UTT-16: 0000000000.00101001

a) $a = 12 \leq 34$ | boolean, true

$b = 56 / 78$ | int, 0

$c = 90 - 0,7$ | double, 9.0

$d = "23" + 45$ | string und int zusammen geht nicht

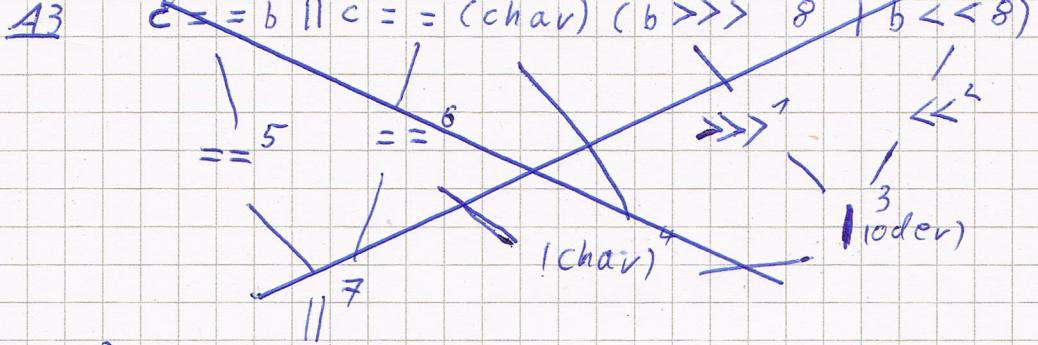
$e = \text{new double}[67]$ | double[67] {0,0,...0}

b) Werttypen: byte, short, int, long, float, double,
char, boolean

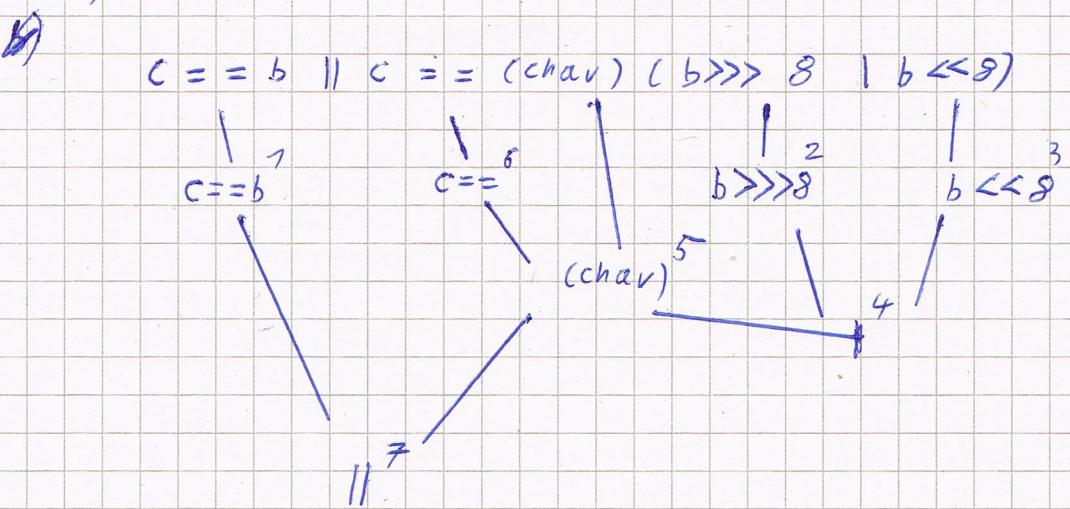
Referenztypen: array[], string, enum

c) Bei Werttypen wird mit '=' ein Wert zugewiesen und mit '==' dieser verglichen.
Bei Referenztypen wird eine (speicher)-
Adresse zugewiesen unter der der "Wert"
zu finden ist. Mit '==' werden dann auch
nur die Adressen verglichen.

d) Object, durch toString wird die
String-Darstellung von Beispiel geliefert.



a)



b) 't'

Jannis
Liebsche
307645

a) Hans' und Heinz' und Hertas und
erstes Java-Programm funktioniert!

b)

```
String s = " ";
for (int i = 0; args.length - 1; i++) {
    System.out.print(s + "%s", args[i]);
    s = " und ";
}
```

c) ~~for Schleife aus b)~~

~~if~~

`String vorname = args[i]`

`if (vorname.charAt(vorname.length() - 1) == 's'
|| 'x'
|| 'z')`

`System.out.print("\\"))`

`} else {`

`System.out.print('s');`

a) `int i = 63;`

`String s = valueOf(i);`

b) Die String-Darstellung ist die Referenz/
(Speicher)Adresse des Objekts

c) `String s = "wert"`

`String v = " aba "`

`if (s.equals(v)) --`

d) Bevor man mit `out(s + v)` Strings
verknüpfen konnte wurde dies mit dem
Stringbuilder mit `"append(string)"`
realisiert. Das geht immer noch

a)

A.java

```
public final class A {  
    public int static int KV = 11;  
    public int iv = 22;  
    public void set(int p) {  
        iv = p;  
    }  
}
```

b) B.java

```
public final class B {  
    private B() {}  
    public static void main(String[] args) {  
        public A v = new A;  
        System.out.println("%d", A.KV);  
        System.out.println("%d", A.iv);  
    }  
}
```

c) - private Konstruktoren

- fabrikmethoden, die die Werte überprüfen
- sinnvolle Typen (int, double etc)
- private instanzvariablen auf die nur
 - über fabrikinstanz methoden

A7

307645
Janis
Liebsche

a) C.java

```
public abstract class C {  
    private int PV;  
    public C(int P) {  
        this.PV = P;  
        return;  
    }  
    public final int get() {  
        return this.PV;  
        return this.PV;  
    }  
}
```

b) D.java

```
public class D extends C {  
    public D() {  
        this.PV = 7;  
        return;  
    }  
}
```

c) E.java

```
public final class E {  
    private E() {}  
    public static void main(String[] args) {  
        public C a = new C();  
        public D b = new D();  
    }  
}
```

d) als "final int this" vor "int p"

Geschwindigkeit.java

```
public final class Geschwindigkeit {  
    private final double speed;  
  
    private final double SPEEDOFL = 299792485;  
  
    private Geschwindigkeit(double d) {  
        if (d >= 0 && d <= SPEEDOFL) {  
            this.speed = d;  
        } else {  
            throw new IllegalArgumentException("Speed must be non-negative and less than or equal to the speed of light (" + SPEEDOFL + " m/s).");  
        }  
    }  
  
    public Geschwindigkeit valueOf(double d, String s) {  
        enum überlesen, if (s.equals("km\\h")) {  
            aber keine zeit  
            zum ändern.  
            eigentlich  
            mit switch/case  
            wie bei getValve  
            {  
                if (s.equals("km\\h")) {  
                    return new Geschwindigkeit(d * KMH);  
                } else if (s.equals("kn")) {  
                    return new Geschwindigkeit(d * KNOM);  
                } else if (s.equals("mph")) {  
                    return new Geschwindigkeit(d * MPHOM);  
                } else {  
                    throw new IllegalArgumentException("Unknown unit of speed: " + s);  
                }  
            }  
        }  
    }  
}
```

```
public final double KMH = 0.27778;  
public final double KN = 0.57444;  
public final double MPH = 0.44704;
```

A2

Jannis
Liebscher
304645

```
public double getValue(enum e String e) {
    switch (e) {
        case KMPH_KMH: return this.speed * KMH;
        break;
        case KN: return this.speed * KN;
        break;
        case MPH: return this.speed * MPH;
        break;
    default: throw new IllegalArgumentException();
    }
}
```

b)

- public String toString()
- public boolean equals(Object o)
- public int hashCode()

c) Die Comparable Schnittstelle um eine
natürliche Ordnung festzulegen.