Programmiertechnik 1

Oskar Borkenhagen
301639

Aufgabe 1)

a.) 88 binär: $1011000 = 2^5 + 2^4 + 2^6 = 89$

oktal: $88 / 8 = 11$ R: 0

$11 / 8 = 1$ R: 3

$3 / 8 = 0$ R: 3

$\Rightarrow$ 330

hex: $88/16 = 5$ R: 8

$5/16 = 0$ R: 5

$\Rightarrow$ 58

Gleitkomma: $.88e^2$ oder $88.$

b.) -88 binär 88: 01011000

1er Komp.: 10100111

2er Komp.: ~~10100111~~ 10101000

d.) 1 : 061. $\rightarrow$ hex: $61/16 = 3$ R: 13 (D)

$3/16 = 0$ R: 3

1 in UTF-16 = U+003D $\rightarrow$ 3D

c.) 101,101 $\Rightarrow$ UK: $101 = 2^2 + 1 = 5$

NK: $,101 = \frac{1}{2} + \frac{1}{8} = 0,125$

$\Rightarrow$ 5,125

Aufgabe 2)

a.) boolean a = true
int b = 0
double c = 9.0
String d = "2345"
Double e = 67.0

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

Referenztypen: Arrays also: Typ[], String, enum und alles was von Objekt abl

c.) == : Bei Werttypen wird auf Wert-Gleichheit ~~gut~~ geprüft, also 1==1 => true
Bei Referenztypen wird die Identität geprüft, also ob der Speicherort der Selbe ist.

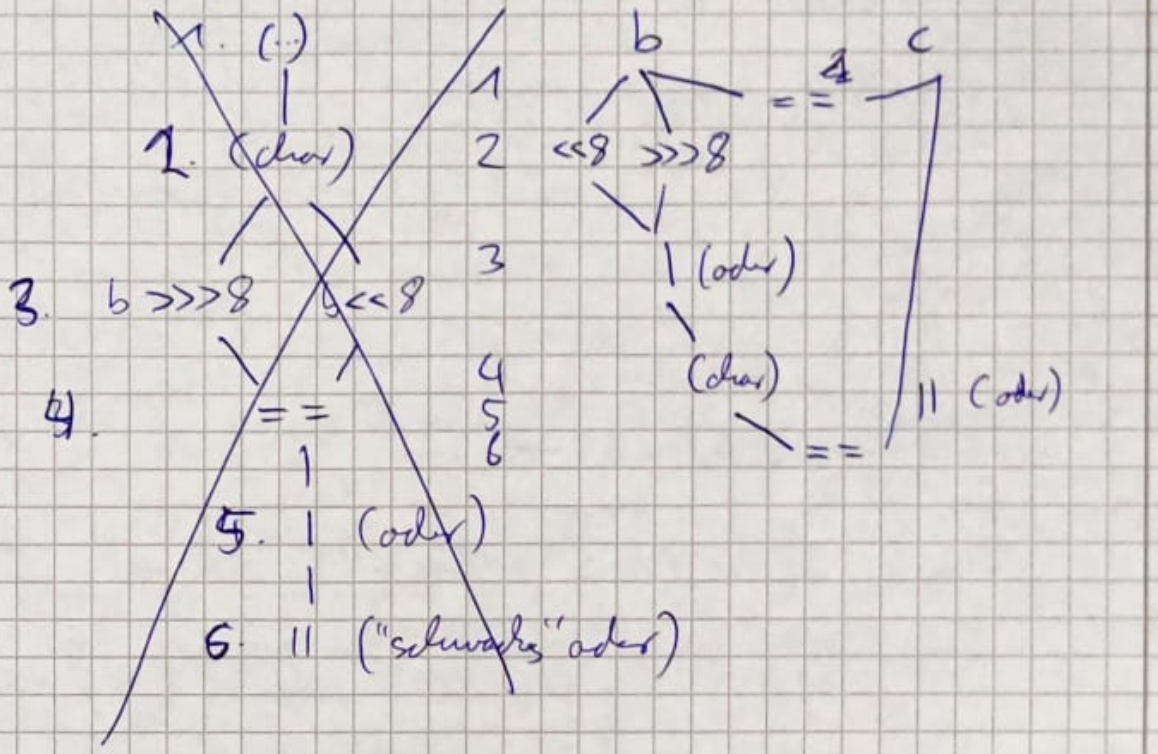
= : Bei Werttypen eine Wertzuweisung.
Bei Referenztypen wird auf eine Referenz verwiesen, die einen Werttyps beinhaltet oder verweist.

d.) Objekt ist die Oberste Klasse aller Objekte. Das Beispiel funktioniert nur, weil new Boolean() eine automatische ~~Wert~~ Typanpassung erfährt (Autoboxing). ~~Dies~~ Dies gilt, da Objekt ~~an alles~~ auf alle Unterklassen projiziert werden kann. → Polymorphie

Oscar Barzenleger
301639

Aufgabe 3)

a) `c == b || c == (char)(b >>> 8 | b << 8)`



b) `\ufeff` → 1111111011111111

b >>> 8 → 0000000011111110

=> char c = ~~'\u000f'~~

= '\u00fe'

Aufgabe 4)

a) ~~'und' und s und erstes Java-Programm funktioniert!~~

~~Hans' und Heinz~~

X Hans' und Heinz' und Hertas erstes Java-Programm funktioniert!

b)
```
for(int i = 0; i < args.length; ++i) {
    System.out.print(s + args[i]);
    char c = args[i].charAt(args[i].length() - 1);
//Zeile 8-20 bleiben gleich
}
```

c.)

```
if(c=='s' || c=='x' || c=='z') {
  System.out.print('\'');
} else {
  System.out.print('s');
}
```

Aufgabe 5)

a.) String.valueOf(...)

z.B. ~~String valueOf(int x)~~

```
int x = 3;
String s = String.valueOf(x);
```

b.) Die Standard toString() Methode gibt den Klassennamen und den Hashcode zurück, also z.B.: Klasse1@3174

c.) Mit dem .equals() Aufruf
also z.B.: if (a.equals(b)) {...}

d.) StringBuilder wird verwendet um effizient an einen String anzuhängen mit der append() - Methode
z.B.: new StringBuilder(a).append(b) würde -> ab liefern.

Oskar
Bortenhagen
301639

Aufgabe 6)

a)

```
public class A {
    public static int kv = 11;
    public int iv = 22;

    public void set (int p) {
        iv = p;
    }
```

b)

```
public class B {
    public static void main (String[] args) {
    A v = new A();
    System.out.println(String.valueOf(A.kv));
    String tmp = String.valueOf(v.iv);
    System.out.println(tmp);
    }
```

c)

- Alles was nicht öffentlich sein muss wird auf privat gesetzt!
- Getter und Setter für Variablen
- if oder switch Abfragen in einer Fabrikmethode oder im Konstruktor

Aufgabe 7)

a)
```
public abstract class C {
    private int pv;

    public C(int p) {
        this.pv = p;
    }
    public final int get() {
        return pv;
    }
}
```

b)
```
public class D extends C {

    public D() {
        super(7);
    }
}
```

c)
```
public class E {
    public static void main(String[] args) {
        D d = new D();
        C c = new C(5);
        System.out.println(c.get());
        // gibt 5 aus
    }
```

d) this ist als die aktuelle Instanz aus der Klasse zu verstehen. Bei einem Aufruf c.get() ist das this in get() gleichzusetzen mit dem Objekt c

Oskar Bartenbagen
301639

Aufgabe 8)

a)
```
public final class Geschwindigkeit {
  private final double mps;
  private Geschwindigkeit (double s) {
    if ( s < 0) {
    throw new IllegalArgumentException ();
    }
    Double Ls = 299792458.0;
    if(Double.compare (S, Ls) > 0) {
     throw new IllegalArgumentException();
    }
    this.mps = s;
  }
public static Geschwindigkeit valueOf(double s, Einheit e) {
  switch(e) {
  case Einheit.KMH:
       s = 0,27778 * s;
       break;
  case Einheit.KN:
       s = 0,51444 * s;
       break;
  case Einheit.MPH:
       s = 0,44704 * s;
       break;
  }
    return new Geschwindigkeit ( s );
}
public double getValue (Einheit e) {

   // Nächste Seite
```

```
switch (e) {
 case Einheit.KMH:
   return this.mps / 0,27778;
 case Einheit.KN:
   return this.mps / 0,51444;
 case Einheit.MPH:
   return this.mps / 0,44704;
 default:
   return this.mps;
 }
}
```

b.) equals: gibt ein boolean zurück
toString: liefert hier die Instanzvariable mps als String
hashcode: selbst-definierte eindeutige, ganzzahlige Zahl, also int

c.) Das Interface Comparable<T> müsste implementiert werden, also die Methode compareTo(Object o). Dies stellt in der Klasse quasi eine natürliche Ordnung her.