|Aufgabe 1: Multiple Choice

a) AFFE: nein (wegen E oder so)

(Hinweis: oktal – hexadezimal – Umrechnung wissen ?)

b) J-Unit: ja (endlich viele)
c) Object: Nein
d) Integer Wrapperklasse: Ja
e) Protected #: Ja
f) Nein, Default-Package hat nichts mit java.lang zu tun
g) Case-Anweisung: Ja
h) InputStream: Ja

Aufgabe 2: Wert und Typ von Ausdrücken:

Teilaufgabe	Typ	Rückgabewert
s[7/3] -> s[2]	String	„2.0"
Integer.parseInt(s[1])*3.0f	float	9.0f
(1<<1)<<2 -> 1: 00000001 1<<1: 00000010 (1<<1)<<2: 00001000	Integer	8
s[s.length -1].equals(EiP)	Boolean	false
Double.parseDouble(s[2]+1)==Integer.parseInt(s[1]) D.pD(s[2]+1) ergibt 2.01d (!)	boolean	false
1==0||!(false&!true) -> false ||!(false)	boolean	true

Aufgabe 3: Fehler zur Compile- und Laufzeit

a) Graphic und Box:
   - **abstract** boolean black (wahlweise der Methode einen Rumpf geben, aber dann mit return)
   - draw() muss implementiert werden bei Box
   - visible ist private -> Sichtbarkeit ändern oder Getter & Setter

b) Würfel
   NullPointerException in Zeile 14, weil „seiten" und „generator" im Konstruktor lokale Variablen auf dem Stack sind, und die Instanzvariable generator oben noch den default-wert „null" hat.

Aufgabe 4: Array

```
class Sudoku {
public static boolean gueltigeZeile(int[] zeile) {

boolean[] gesehen = new boolean[9];

for(int i = 0; i<9; i++){

        if(gesehen[zeile(i)-1]) return false;

        else gesehen[zeile(i)-1] = true;}

return true; }}
```

Aufgabe 5: Reg-Ex

a) Passt: 206

   Passt nicht: 06, 256

b) Passt: abc, accccbc
   Passt nicht: a

c) ([KDTLS]?)(x?)([a-h][1-8])

Aufgabe 6: Interface

```
interface Bedingung<E> {
	boolean istErfuellt(E element);
}
```

```
import java.util.LinkedList;
import java.util.List;

class ListFilter {
	public static <E> List<E> filtern(List<E> list, Bedingung<E> b){
		List<E> result = new LinkedList<E>();
		for (E element : list) {
			if(b.istErfuellt(element))
				result.add(element);
			}
return result;
}
}
```

Extra-Klasse:

```
Class EnthaeltAtZeichen implements Bedingung<String>{

	Boolean istErfuellt(String element){

		Return element.contains(„b“);}}
```

```
public static void main(String[] args) {
	<String> List = Arrays.asList(args);
		Bedingung<String> atZeichen = new(E enthaeltAtZeichen();
		List<String> gefiltert = filtern(list, atZeichen)
}
```

Aufgabe 7: Threads

Code:

```
Theke theke = new Theke (2)
Theke.start();
Kunde a = new Kunde (“A”, theke);
a.start();
Kunde b = new Kunde (“B”, theke);
b.start();
Kunde a = new Kunde (“C”, theke);
c.start();
```

Mögliche Ausgabe:

A – 1

C – 0

B - -1

Erklärung: /

Run-Methode:

```
Synchronized(theke){
    try{
        while(theke istLeer()) theke.wait();
    }
    // yield würde Theke ganz besetzen
    catch(irgendeine Exception – Interrupted glaub ich);{}
    theke.nimmPizza(this);
    theke.notifyAll();}
```

Aufgabe 7.2: Enums

```
private double preis;

PizzaSorte(double preis){

        this.preis = preis;}

REGINA(7.5), HAWAII(8.0), FUNGHI(8.5);
```

Aufgabe 8: Hoare-Kalkül:

Invariante: a mod b = A mod B

1. Eintritt
   (Pre & B) => INV
   (a = A & b = B) => (a mod b = A mod B)

2. Durchgang
   (B & INV){p}(INV)
   (a>=b & a mod b = A mod B) {a=a-b} (a mod b = A mod B)
   =>
   (a>=b & a mod b = A mod B)=> (a mod b = A mod B)
   Originalantwort von Ohlbach: und das ist jetzt Mathematik, das fällt nicht mehr in das Fachgebiet hier ;)

3. Austritt aus der Schleife:
   (INV & !B) => (POST)
   (a mod b = A mod B &a<b) => (a = A mod B)

   Darf er sich beschweren?
   Nein, Vorbedingung war a>0