Programmierung 2
Studiengang MI / WI

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://mwilhelm.hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt der Vorlesung

Überblick:

- Objekte und Methoden
- Swing
- **Bit-Operationen**
- Exception
- I/O-Klassen / Methoden
- Algorithmen (Das Collections-Framework)
- Design Pattern
- JUnit
- Graphentheorie

Status einer Variablen in Java

- final static int BESTELLT=1;
- final static int OFFEN=2;
- final static int VERSANDT=3;
- final static int BEZAHLT=4;

Eigenschaften einer Variablen in einer Klasse

- final static int ISTGELB=1;
- final static int ISTGROSS=2;
- final static int ISTKLEIN=3;
- final static int ISTVERPACKT=4;
- final static int ISTGESCHENK=5;

- Wert ist 7
- **Eigenschaft?**

Status einer Variablen in Java: ok

- final int BESTELLT=1;
- final int OFFEN=2;
- final int VERSANDT=3;
- final int BEZAHLT=4;

Eigenschaften einer Variablen in einer Klasse, wenn mehrere Eigenschaften möglich sind:

- final static int ISTGELB=1;
- final static int ISTGROSS=2;
- final static int ISTKLEIN=4;
- final static int ISTVERPACKT=8;
- final static int ISTGESCHENK=16;

Logische Verknüpfungen

ODER	0	1
0	0	1
1	1	1

UND	0	1
0	0	0
1	0	1

Logische Verknüpfungen

	0	**1**
NOT	1	0

XOR	**0**	**1**
0	0	1
1	1	0

XOR ergibt eine 1, wenn die Bits unterschiedlich sind!

Logische Verknüpfung NAND

NAND (a,b) = Not And (a,b)

$$c = \overline{a \wedge b} = \bar{a} \vee \bar{b}$$

a	b	c
0	0	1
0	1	1
1	0	1
1	1	0

Logische Verknüpfung NOR

NOR (a,b) = Not Or (a,b)

$$c = \overline{a \vee b} = \bar{a} \wedge \bar{b}$$

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Beispiele:

Diese logische Verknüpfungen werden auch auf Bitfolgen angewandt.

0001	ODER	1000	=	1001
0001	OR	1000	=	1001
0001	$\vee$	1000	=	1001
0001	|	1000	=	1001

1011	UND	1000	=	1000
1011	AND	1000	=	1000
1011	$\wedge$	1000	=	1000
1011	&	1000	=	1000

Beispiele:

Diese logische Verknüpfungen werden auch auf Bitfolgen angewandt.

NOT	1010	=	0101
NICHT	1010	=	0101
¬	1010	=	0101
!	1010	=	0101

1011	XOR 1000	=	0011
1010	XOR 1111	=	????
0101	XOR 1111	=	????

Bit-Operationen in Java

- Logisches AND &&
- Logisches OR ||
- Logisches NOT !

- Bitweises AND &
- Bitweises OR |
- Bitweises NOT ~
- Bitweises XOR ^

Bit-Operationen in Java

```
void test() {
   int i=22;
   int j=2;
   int k;

   k = i & j;
   syso(k);

   k = i | j;
   syso(k);

   k = i ^ j;
   syso(k);

}
```

```
class Bestellung {
  final static int EINGANG=1;
  final static int AUSLIEFERUNG=2;
  final static int RECHNUNG=4;
  String name="";
  ArrayList liste;
  int status=0;

  public Bestellung(…) {}

  public void setStatus(int status){
    this.status |=  status;
  }

  public void delStatus(int status){
     this.status &= ~status;
  }
  // nur die Bits von status werden geprüft
  // wenn alle Bits gesetzt, dann …
  public booelan checkStatus(int status){
    return (this.status & status)==status;
  }
}
```

Bestellung bst=null;

```
 void bnBsp1_Click() {
   bst = new Bestellung("Prog-Bücher",0);
   bst.setStatus(Bestellung.EINGANG);
 }  // bnBsp1_Click

 void bnBsp2_Click() {
   bst.setStatus(Bestellung.RECHNUNG);
 }  // bnBsp2_Click

 void bnBsp3_Click() {
   bst.delStatus(Bestellung.EINGANG);
 }  // bnBsp3_Click

 void bnBsp4_Click() {
   bst.setStatus(Bestellung.MAHNUNG);
 }  // bnBsp4_Click

 void bnBsp5_Click() {
   bst.delStatus(Bestellung.RECHNUNG);
 }  // bnBsp5_Click

void bnBsp6_Click() {                          // neu
   editor.append("\n\nbsp6: check Rechnung: "+ bst.checkStatus( Bestellung.RECHNUNG) );
 }  // bnBsp6_Click
```