

## Klassen

Dr. Philipp Wendler

Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

<https://www.sosy-lab.org/Teaching/2018-WS-InfoEinf/>

WS18/19

### Beispiel 1: Klasse Zimmer

Gegeben sei eine **Klasse Zimmer**  
mit den Eigenschaften  $m^2$ , Anzahl Türen, Anzahl Fenster

#### Was ist ein Objekt der Klasse Zimmer?

- Kellerzimmer mit den Eigenschaften  
 $m^2$ , Anzahl Türen
- Büro mit den Eigenschaften  
 $m^2=15$ , Anzahl Türen=1
- Büro mit den Eigenschaften  
 $m^2=15$ , Anzahl Türen=1, Anzahl Fenster=3
- Büro mit den Eigenschaften  
 $m^2=15$ , Anzahl Türen=1, Anzahl Fenster=3, Mitarbeiter=Annabelle

## Klassen vs. Objekte

- Eine **Klasse** ist eine Schablone/Konstruktionsplan für eine Menge von gleichartigen Gegenständen, Dingen, Konzepten...
  - Z.B. **Klasse Mensch** mit den Eigenschaften Name, Geschlecht, Alter, Größe
- Ein **Objekt** füllt diese Schablone mit bestimmten Werten und ist eine konkrete Ausprägung der Klasse.
  - Z.B. **Objekt Annabelle** vom Klassentyp Mensch mit den Eigenschaften Name=Annabelle, Geschlecht=weiblich, Alter=30, Größe=165
  - Z.B. **Objekt Rolf Hennicker** vom Klassentyp Mensch mit den Eigenschaften Name=Rolf Hennicker, Geschlecht=männlich, Alter=..., Größe=...

### Allgemeiner Aufbau einer Klasse in Java (1)

```
public class C {
    private type1 attr1;
    ...
    private typen attrn;
}

public C(params) {body}
...
...
}
```

Attribute (Eigenschaften)

Konstruktoren

## Beispiel 2: Klasse Mensch (1)

```
public class Mensch {
    private String name;
    private String geschlecht;
    private int alter;
    private int groesse;

    public Mensch(String name0, String geschlecht0,
        int alter0, int groesse0) {
        this.name = name0;
        this.geschlecht = geschlecht0;
        this.alter = alter0;
        this.groesse = groesse0;
    }
}
```

Attribute (Eigenschaften)

Konstruktor

## Methoden einer Klasse

- Ein Klasse legt nicht nur die charakteristische Eigenschaften fest, sondern auch das charakteristische Verhalten.
- Das charakteristische Verhalten wird durch Methoden beschrieben:
  - Methoden können **Auskunft über den aktuellen Zustand** eines Objekts geben.
  - Methoden können den aktuellen Zustand eines Objekts **verändern**.
  - Methoden können möglicherweise den aktuellen Zustand **anderer Objekte verändern**.
  - Methoden können **komplexe Berechnungen** anstellen.

## Allgemeiner Aufbau einer Klasse in Java (2)

```
public class C {
    private type1 attr1;
    ...
    private typen attrn;

    public C(params) {body}
    ...

    public type1/void methodName1(params1) {body1}
    ...
    public typek/void methodNamek(paramsk) {bodyk}
}
```

Attribute (Eigenschaften)

Konstruktoren

Methoden

## Beispiel 2: Klasse Mensch (2)

```
public class Mensch {
    ... //wie vorher
    public int getGroesse() {
        return this.groesse;
    }

    public void wachsen(int cm) {
        this.groesse = this.groesse + cm;
    }

    public boolean istSehrGross() {
        return this.groesse >= 190;
    }
}
```

gibt Auskunft über den aktuellen Zustand

verändert den aktuellen Zustand

stellt komplexe Berechnungen an

## Beispiel 1: Klasse Zimmer

Gegeben sei eine **Klasse Zimmer** mit den Eigenschaften m<sup>2</sup>, Anzahl Türen, Anzahl Fenster

### Welche Methode hat einen Rückgabebetyp?

- eine Methode, um ein zusätzliches Fenster einzubauen verändert den aktuellen Zustand
- eine Methode, um festzustellen, ob das Zimmer fensterlos ist gibt Auskunft über den aktuellen Zustand
- eine Methode, um festzustellen, ob das Zimmer groß genug für zwei Mitarbeiter ist stellt komplexe Berechnungen an
- eine Methode, um die Tür nach einem Einbruch polizeilich zu versiegeln verändert den aktuellen Zustand

## Beispiel 3: Objekte im Speicher

```
public class Vertauscher {
    public void vertausche(int a, int b) {
        int tmp = a;
        a = b;
        b = tmp;
    }

    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        int x = 1, y = 2;
        System.out.println("x=" + x + ", y=" + y);
        v.vertausche(x, y);
        System.out.println("x=" + x + ", y=" + y);
    }
}
```

### Welchen Wert haben x und y am Ende des Programms?

- x = 1, y = 1
- x = 1, y = 2
- x = 2, y = 1
- x = 2, y = 2

## Objekterzeugung und -verwendung

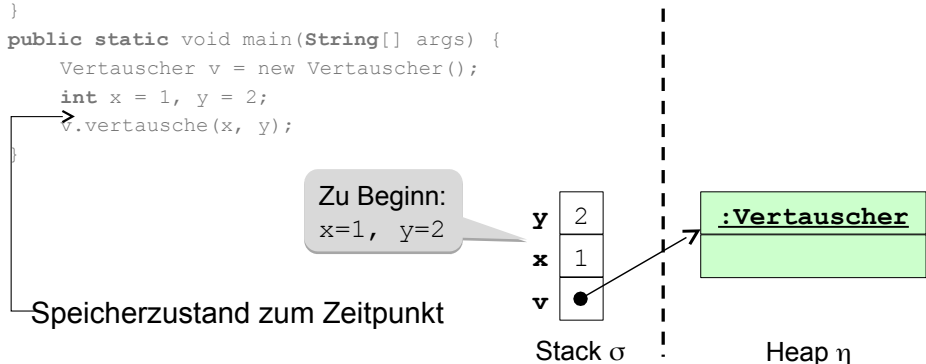
Objekte werden meistens in Methoden anderer Klassen erzeugt und benutzt:

```
public class MenschTest {
    public static void main(String[] args) {
        Mensch annabelle =
            new Mensch("Annabelle", "weiblich", 28, 165);
        int g1 = annabelle.getGroesse(); //ergibt 165
        annabelle.wachsen(10);
        int g2 = annabelle.getGroesse(); //ergibt 175
    }
}
```

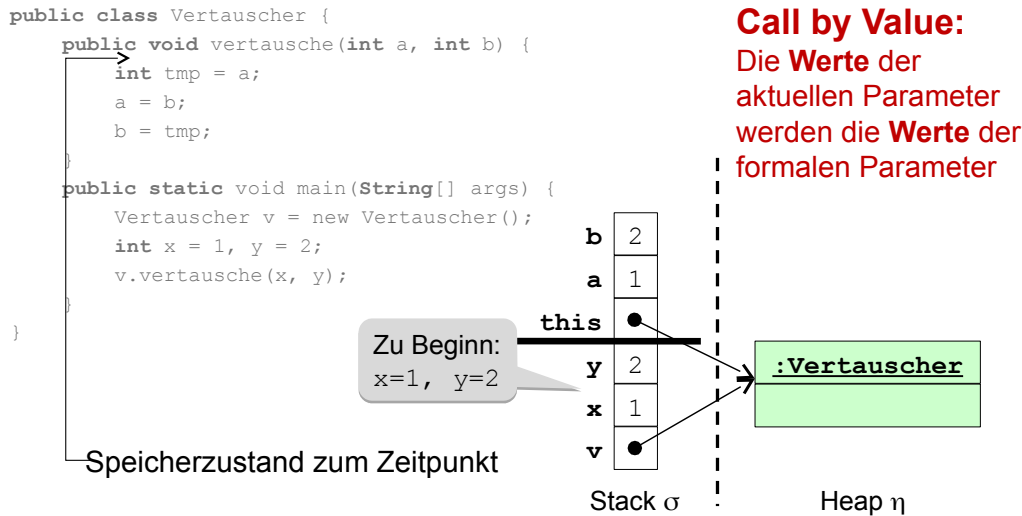
## Beispiel 3: Speicherentwicklung (1)

```
public class Vertauscher {
    public void vertausche(int a, int b) {
        int tmp = a;
        a = b;
        b = tmp;
    }

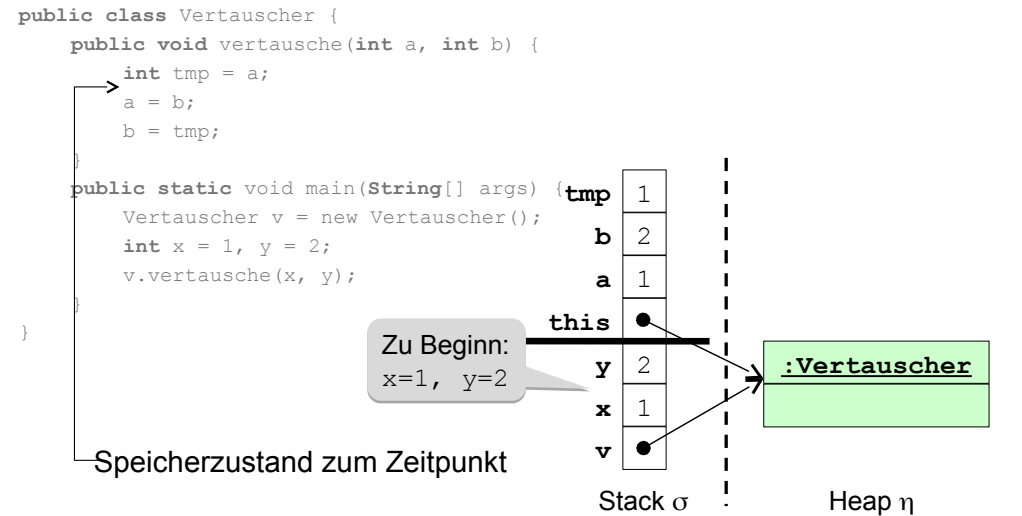
    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        int x = 1, y = 2;
        v.vertausche(x, y);
    }
}
```



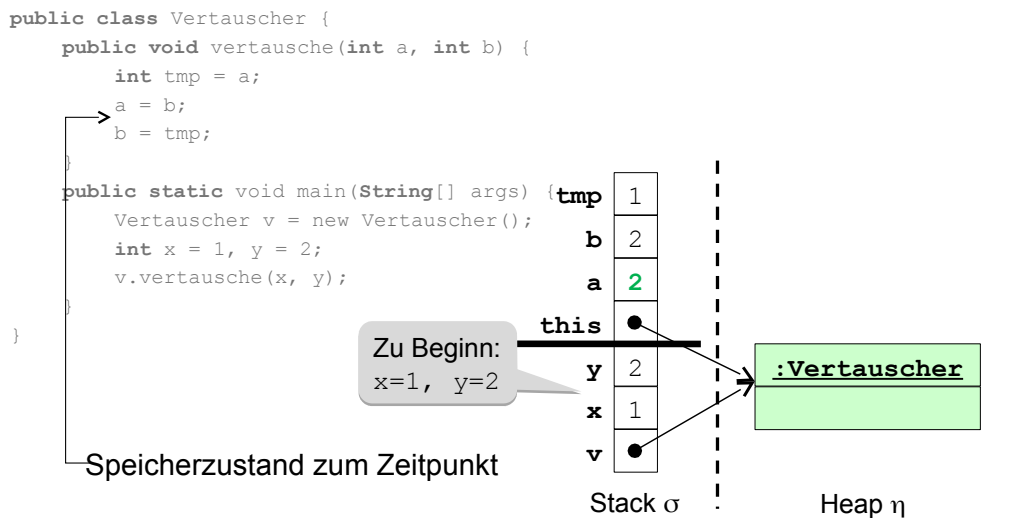
### Beispiel 3: Speicherentwicklung (2)



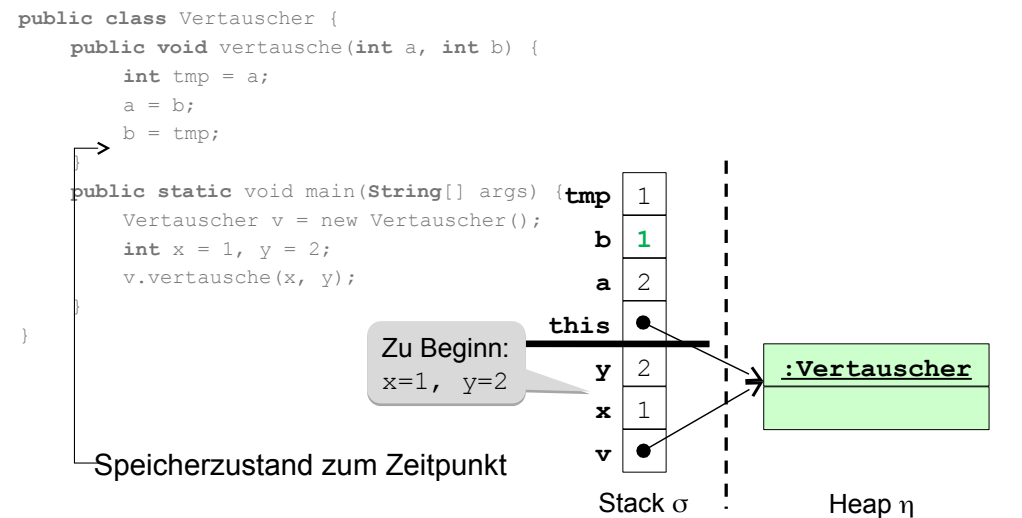
### Beispiel 3: Speicherentwicklung (3)



### Beispiel 3: Speicherentwicklung (4)

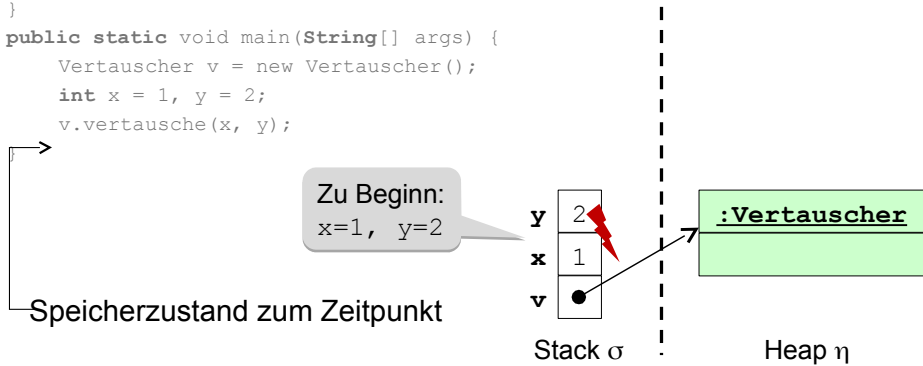


### Beispiel 3: Speicherentwicklung (5)



### Beispiel 3: Speicherentwicklung (6)

```
public class Vertauscher {
    public void vertausche(int a, int b) {
        int tmp = a;
        a = b;
        b = tmp;
    }
    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        int x = 1, y = 2;
        v.vertausche(x, y);
    }
}
```



### Beispiel 3 (verbessert): Objekte im Speicher

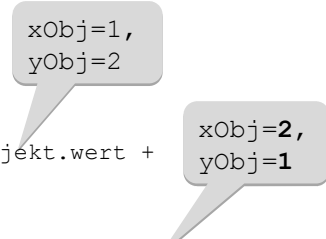
Wir verwenden statt Werten vom Grunddatentyp `int` Objekte einer Klasse `IntObjekt`.

```
public class IntObjekt {
    public int wert;

    public IntObjekt(int wert) {
        this.wert = wert;
    }
}
```

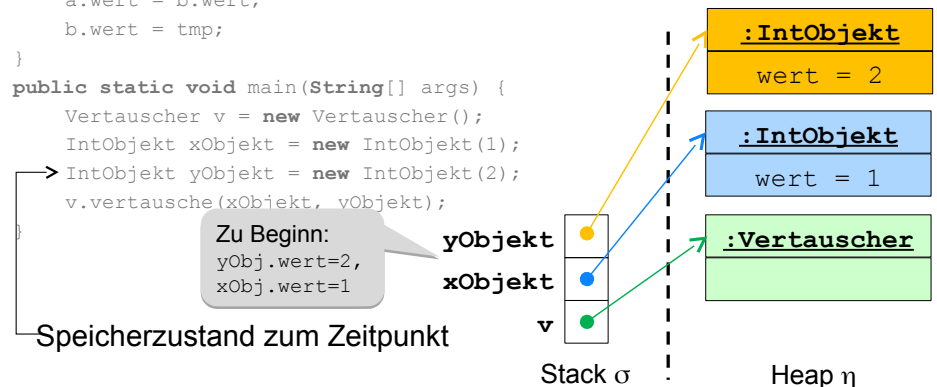
### Beispiel 3 (verbessert): Objekte im Speicher

```
public class Vertauscher {
    public void vertausche(IntObjekt a, IntObjekt b) {
        int tmp = a.wert;
        a.wert = b.wert;
        b.wert = tmp;
    }
    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        IntObjekt xObjekt = new IntObjekt(1);
        IntObjekt yObjekt = new IntObjekt(2);
        System.out.println("xObjekt.wert=" + xObjekt.wert +
            ", yObjekt.wert=" + yObjekt.wert);
        v.vertausche(xObjekt, yObjekt);
        System.out.println("xObjekt.wert=" + xObjekt.wert +
            ", yObjekt.wert=" + yObjekt.wert);
    }
}
```

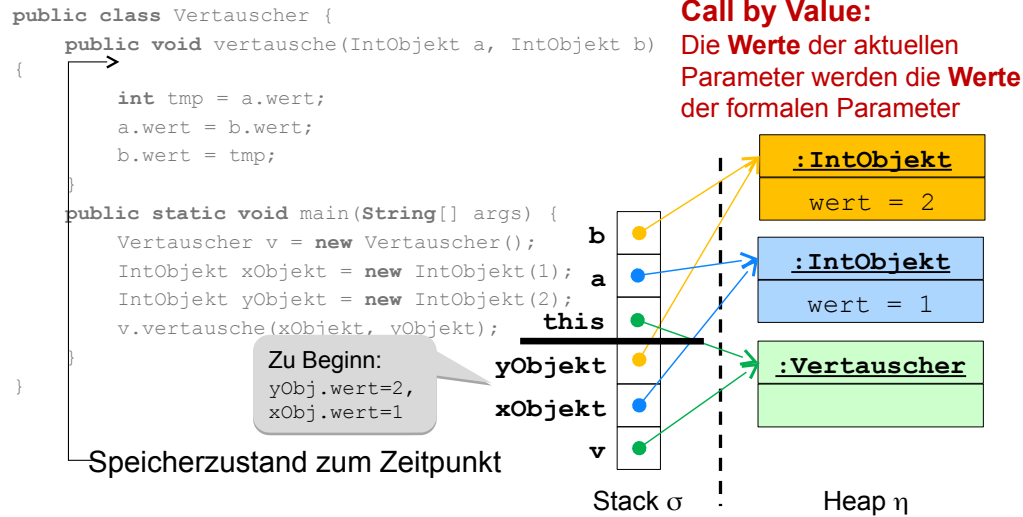


### Beispiel 3 (verbessert): Speicherentwicklung (1)

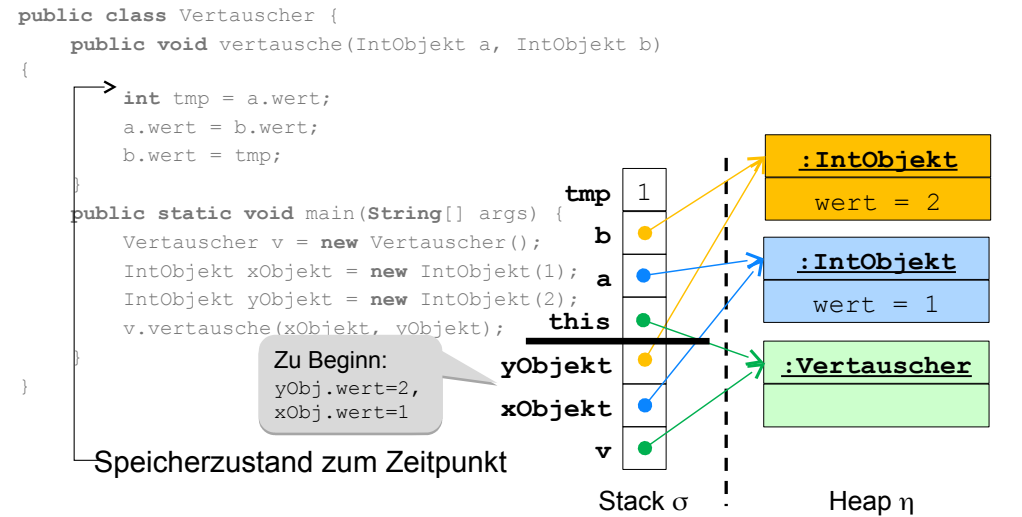
```
public class Vertauscher {
    public void vertausche(IntObjekt a, IntObjekt b) {
        int tmp = a.wert;
        a.wert = b.wert;
        b.wert = tmp;
    }
    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        IntObjekt xObjekt = new IntObjekt(1);
        IntObjekt yObjekt = new IntObjekt(2);
        v.vertausche(xObjekt, yObjekt);
    }
}
```



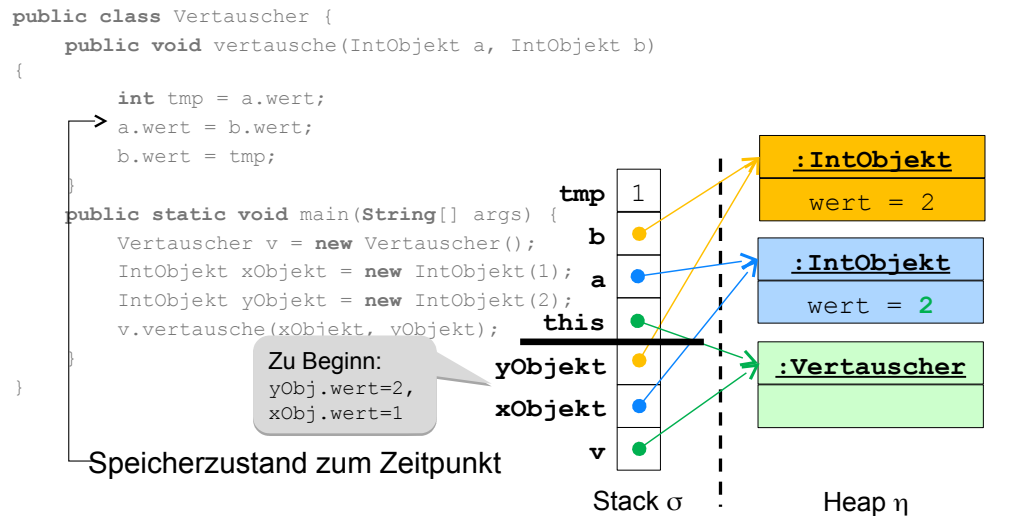
### Beispiel 3 (verbessert): Speicherentwicklung (2)



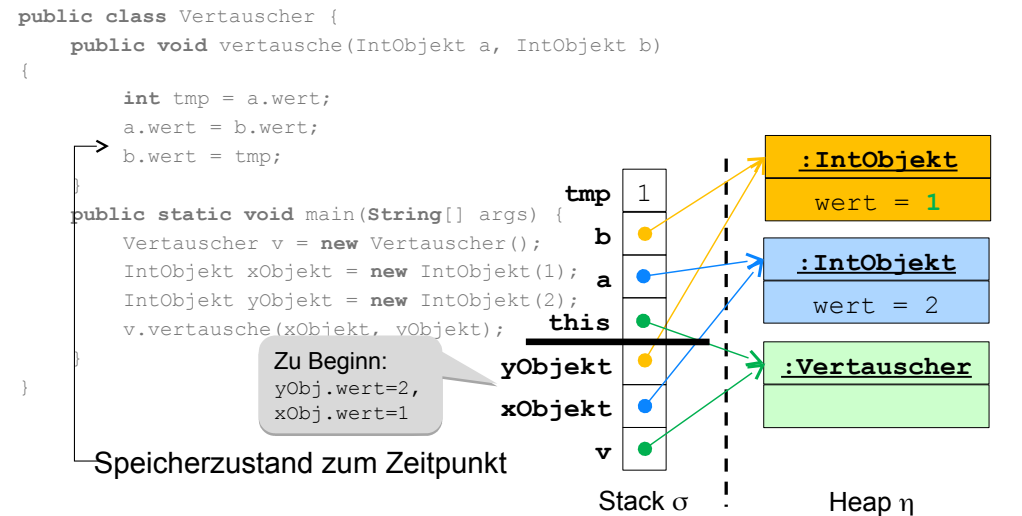
### Beispiel 3 (verbessert): Speicherentwicklung (3)



### Beispiel 3 (verbessert): Speicherentwicklung (4)



### Beispiel 3 (verbessert): Speicherentwicklung (5)



### Beispiel 3 (verbessert): Speicherentwicklung (6)

```
public class Vertauscher {
    public void vertausche(IntObjekt a, IntObjekt b)
    {
        int tmp = a.wert;
        a.wert = b.wert;
        b.wert = tmp;
    }
    public static void main(String[] args) {
        Vertauscher v = new Vertauscher();
        IntObjekt xObjekt = new IntObjekt(1);
        IntObjekt yObjekt = new IntObjekt(2);
        v.vertausche(xObjekt, yObjekt);
    }
}
```

Zu Beginn:  
yObj.wert=2,  
xObj.wert=1

Speicherzustand zum Zeitpunkt

