3-schichtige Informationssystem-Architektur

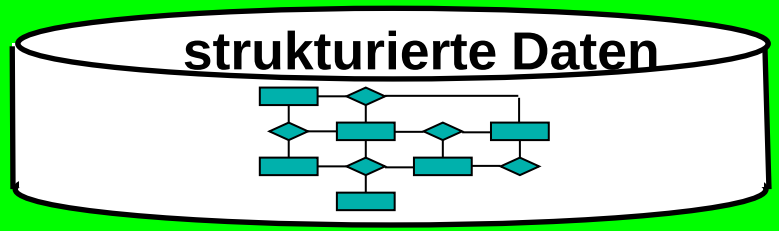
- ***plattformunabhängig***
- ***beliebige Endgeräte***
- ***kein Konfigurationsaufwand***

zuerst:
lokal

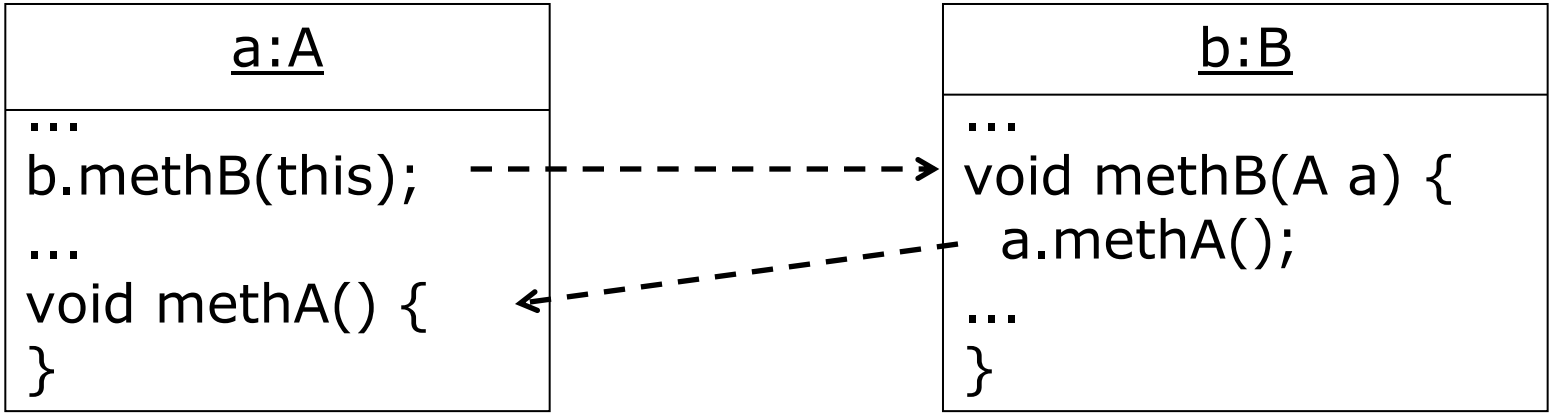
- ***plattformunabhängig***
- ***objektorientierte Architektur***

Server: Anwendungslogik

- ***verteilte Datenbank***
- ***hier einfach Dateisystem des PCs***



Call-Back



Wie kommt a an Referenz von b?
Wo werden die beiden Objekte erzeugt?

Call-Back Beispiel

```
public class A {
// Konstruktor:
  A(B bRef) {
    this.bRef = bRef;
  }
// Methode, die von außen aufgerufen wird:
  void rufeBAuf() {
    System.out.println("Bin in A in rufeBAuf!");
    this.bRef.ichBin(this);
  }
  void zurückVonB() {
    System.out.println("Bin in A in zurückVonB!");
  }
// Merken der Referenz von B:
  private B bRef;
}
```

← Übergabe der Ref. zu b an a.

← Übergabe der Ref. zu a an b.

Call-Back Beispiel 2

```
public class B {
  void ichBin(A a) {
    System.out.println("Bin in B bei ichBin!");
    a.zurückVonB();
  }
}
```

Call-Back

Orb

```
public class Steuerung {

  public static void main(String[] args) {

    B b = new B();

    A a = new A(b);

    a.rufeBAuf();

  }

}
```

Übergabe der Ref. zu b an a.

Damit verliert Steuerung die Kontrolle!

ORB-Beispiel

```
public class A {
  A(Orb orbRef) {
    this.orbRef = orbRef;
  }

  void rufeBAuf() {
    System.out.println("Bin in A in rufeBAuf!");
    this.orbRef.ichBin(this);
  }

  void zurückVonB() {
    System.out.println("Bin in A in zurückVonB!");
  }

  private Orb orbRef;
}
```

←— Übergabe der Ref. von a an orb.

ORB-Beispiel 2

```
public class B {
  void ichBin(Orb orb) {
    System.out.println("Bin in B bei ichBin!");
    orb.zurückVonB();
  }
}
```

Call-Back geht über orb

```
public class Orb {

  public static void main(String[] args) {

    B b = new B();

    A a = new A(b);

    a.rufeBAuf();

  }
}
```

Hier muss `this` rein

Geht nicht, weil von Orb kein Objekt existiert!

ORB-Beispiel 3

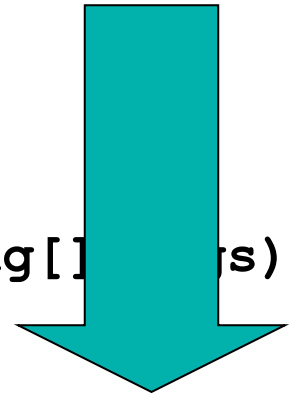
```
public class Orb {

  Orb() {
    b = new B();
    a = new A(this);
    a.rufeBAuf();
  }

  private B b;
  private A a;

  public static void main(String[] args) {
    Orb orb = new Orb();
  }
}
```

Geht schief, weil Orb nicht `zurückVonB()` implementiert



Da Orb Vermittler, muss er alle Dienste von A und B anbieten, die von anderen genutzt werden!

Orb-Implementierung fremder Dienste

```
public class Orb {
   ...
   void ichBin(A a) {
     System.out.println("Bin in Orb bei ichBin!");
     b.ichBin(this);
   }

   void zurückVonB() {
     System.out.println("Bin in Orb bei zurückVonB!");
     a.zurückVonB();
   }

   private B b;
   private A a;
   ...
}
```

Speicherung der Referenzen, da sie in Konstruktor erzeugt und nach Beendigung des Konstruktors weg wären.

Sequenzdiagramm