



Arrays von Objekten

Annabelle Klarl

Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

<http://www.pst.ifi.lmu.de/Lehre/wise-12-13/infoeinf>



Arrays: Wiederholung

Ein Array ist ein Tupel von Elementen gleichen Typs

$$\text{array} = [w_1, w_2, w_3, \dots, w_n]$$

- Reihenfolge relevant: $[w_1, w_2] \neq [w_2, w_1]$
- Zugriff auf ein bestimmtes Element möglich z.B. $a[0] = w_1$
Achtung! Array: $[w_1, w_2, w_3, \dots, w_n]$

	↑	↑	↑	↑
Position:	0	1	2	... n-1
- Elemente müssen den gleichen Typ haben
 - Grunddatentyp z.B. $[1, 2, 3, 4]$, aber nicht $[1, 1.0, 2]$
 - **Klassentyp z.B. $[\text{Konto1}, \text{Konto2}]$, aber nicht $[\text{Konto1}, \text{Fahrzeug1}]$**
 - Arraytyp z.B. $[[1, 2], [3, 4]]$, aber nicht $[[1, 2], [1.0, 2.0]]$



Arrays von Objekten: Beispiel

Sie eröffnen eine Bank mit einem fantasiereichen Namen. Um Ihr Risiko überschaubar zu halten, beschließen Sie nur eine **begrenzte** Anzahl von Bankkonten in Ihrer Bank zu verwalten.

Array kann nur eine begrenzte Anzahl an Bankkonten speichern (Achtung: geeignete Initialisierung)

Bank
-String name -BankKonto[] konten -int anzahlEröffneterKonten
+Bank(String name, int maxAnzahlKonten) +String getName() ...

BankKonto
-int kontoNummer -double kontoStand
+BankKonto(int kontoNummer, double anfangsBetrag) +int getKontoNummer() +double getKontoStand() +void einzahlen(double x) +void abheben(double y)

Anzahl aller schon eröffneten Konten



Arrays von Objekten: Partielle Arrays

Das Array `konten` vom Typ `BankKonto[]` speichert alle aktuell eröffneten Bankkonten bis zu einer Maximalanzahl n , d.h.

- das Array muss so initialisiert werden, dass nur **maximal n** Bankkonten gespeichert werden können.
- die Bank muss sich merken, **wie viele** Bankkonten schon eröffnet wurden.

```
// Konstruktor
public Bank(String name, int maxAnzahlKonten) {
    this.name = name;
    this.konten = new BankKonto[maxAnzahlKonten];
    this.anzahlEröffneterKonten = 0;
}
```



Arrays von Objekten: Partielle Arrays

Das Array `konten` vom Typ `BankKonto[]` speichert alle aktuell eröffneten Bankkonten bis zu einer Maximalanzahl n , d.h.

bei Eröffnung eines neuen Bankkontos muss...

- ... geprüft werden, ob die Max-anzahl n noch nicht erreicht wurde.
- ... die Anzahl der eröffneten Konten um eins erhöht werden.

```
public void kontoEroeffnen(int kontoNummer,  
                           double anfangsBetrag) {  
    if (this.anzahlEroeffneterKonten < this.konten.length) {  
        this.konten[this.anzahlEroeffneterKonten] =  
            new BankKonto(kontoNummer, anfangsBetrag);  
        this.anzahlEroeffneterKonten++;  
    }  
}
```



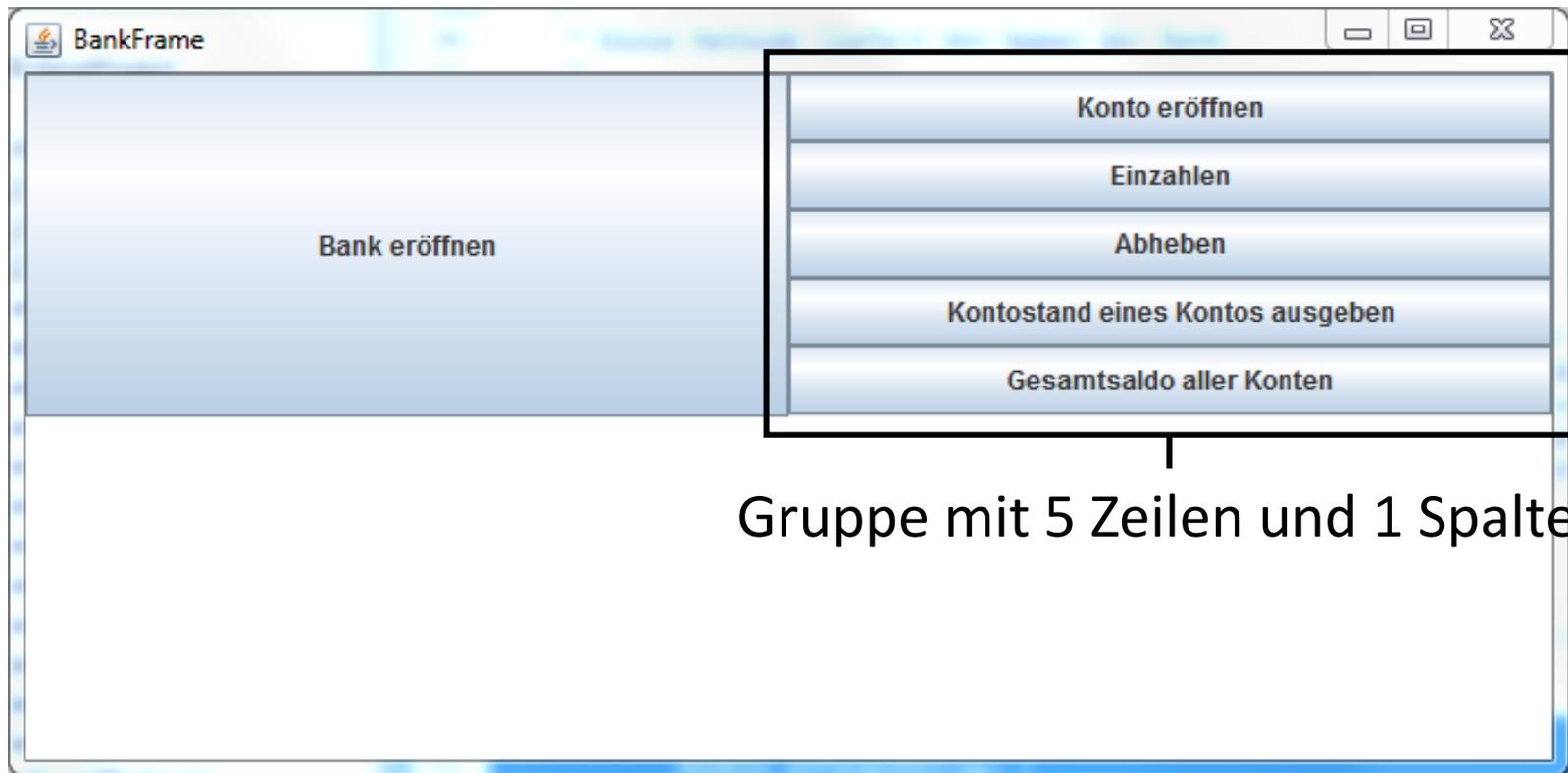
Arrays von Objekten: Bank-GUI

Implementierung wie bisher



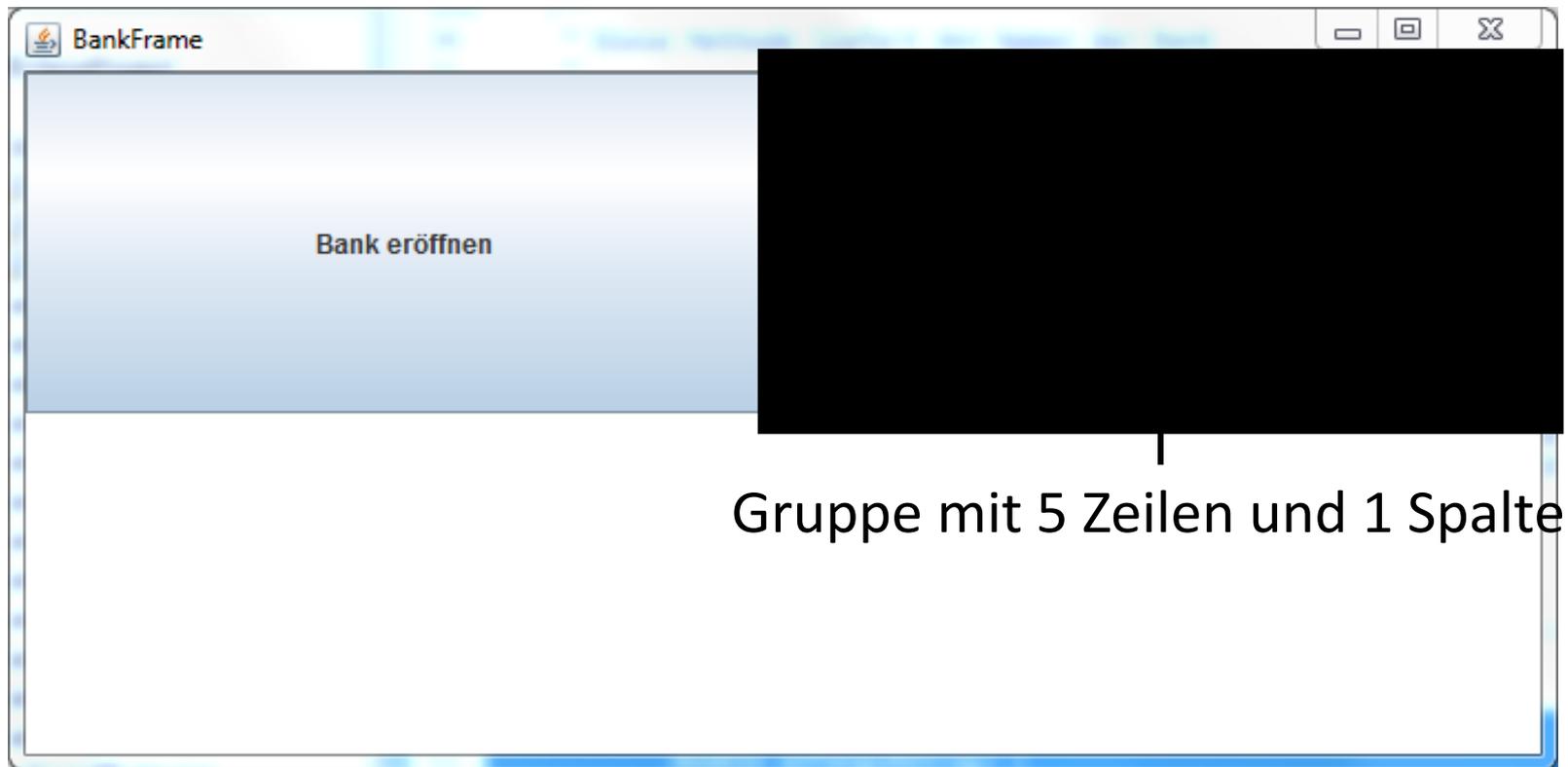


Arrays von Objekten: Bank-GUI

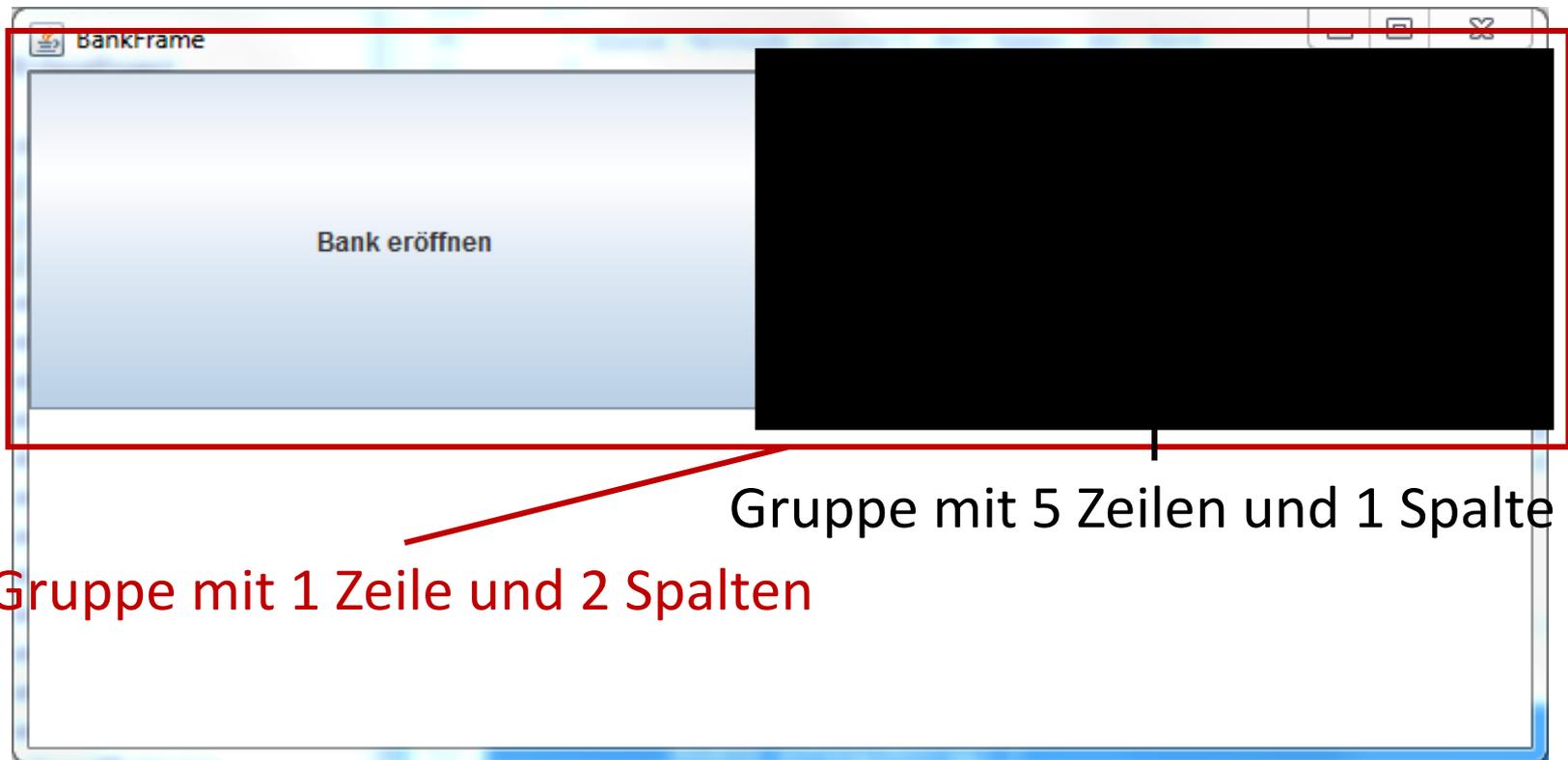




Arrays von Objekten: Bank-GUI



Arrays von Objekten: Bank-GUI



Gruppe mit 5 Zeilen und 1 Spalte

Gruppe mit 1 Zeile und 2 Spalten



Arrays von Objekten: Bank-GUI



Gruppe mit 1 Zeile und 2 Spalten

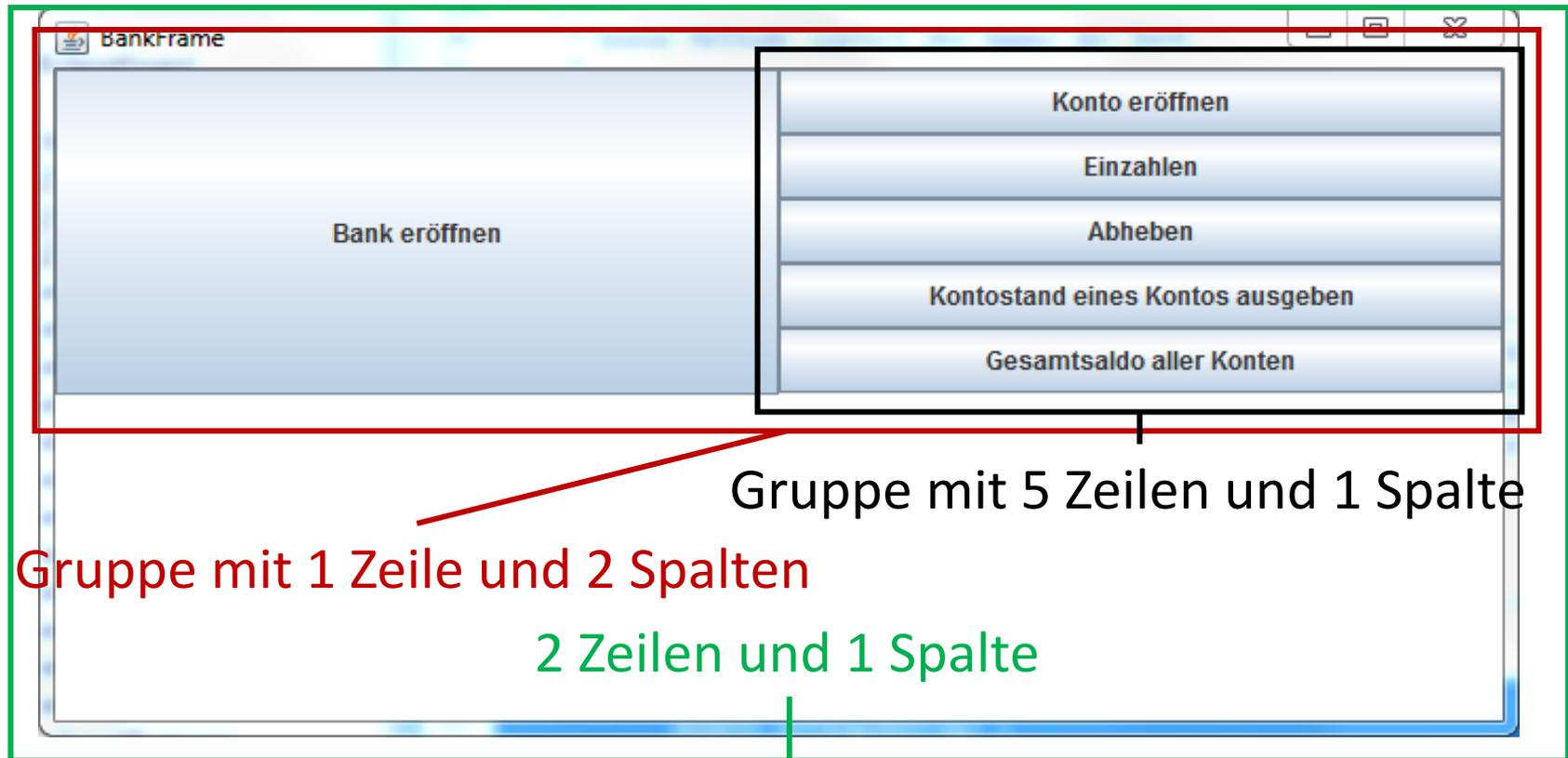


Arrays von Objekten: Bank-GUI





Arrays von Objekten: Bank-GUI





Arrays von Objekten: Button “Bank eröffnen” (I)

- Erzeugung (Konstruktor von `BankFrame`):

```
this.bankEroeffnenButton =  
    new JButton("Bank eröffnen");
```

- Platzierung (Konstruktor von `BankFrame`):
siehe Gruppierungen

- `ActionListener` registrieren (Konstruktor von `BankFrame`):

```
this.bankEroeffnenButton.addActionListener(this);
```

- Ereignisbehandlung

```
public void actionPerformed(ActionEvent e) {  
    if (e.getSource() == this.bankEroeffnenButton) {  
        this.bankEroeffnen();  
    }  
}
```



Arrays von Objekten: Button “Bank eröffnen” (II)

- Die Funktionalität des Buttons wird in der Methode `bankEroeffnen` der Klasse `BankFrame` umgesetzt

```
private void bankEroeffnen() {  
    ... // Fehlerbehandlung, falls Bank schon eröffnet  
    String name = JOptionPane.showInputDialog("Name");  
    String einlesenMax =  
        JOptionPane.showInputDialog("Max-Anzahl");  
    int max = Integer.parseInt(einlesenMaxKonten);  
  
    this.bank = new Bank(name, max);  
    this.ausgabeBereich.setText("Bank eröffnet");  
}
```



Arrays von Objekten: Button “Konto eröffnen” (I)

- Erzeugen, Platzieren, `ActionListener`, Ereignisbehandlung analog
- Die Funktionalität des Buttons wird in der Methode `kontoEroeffnen` der Klasse `BankFrame` umgesetzt

```
private void kontoEroeffnen () {  
    ... // Fehlerbehandlung, falls keine Bank eröffnet  
    int kontoNr = Integer.parseInt(  
        JOptionPane.showInputDialog("Kontonummer"));  
    double anfangsBetrag = Double.parseDouble(  
        JOptionPane.showInputDialog("Anfangsbetrag"));  
    boolean eroeffnet =  
        this.bank.kontoEroeffnen(kontoNr, anfangsBetrag);  
    if(eroeffnet) this.ausgabeBereich.setText("eröffnet");  
}
```



Arrays von Objekten: Button “Konto eröffnen” (II)

■ Konto in der Bank eröffnen

d.h. Methode `kontoEroeffnen` in der Klasse `Bank`

```
public boolean kontoEroeffnen(int kontoNummer,  
                               double anfangsBetrag) {  
    if (this.anzahlEroeffneterKonten <  
        this.konten.length) {  
        this.konten[this.anzahlEroeffneterKonten] =  
            new BankKonto(kontoNummer, anfangsBetrag);  
        this.anzahlEroeffneterKonten++;  
        return true;  
    }  
    return false; }  
    ↖  
    Partielles Array!
```



Arrays von Objekten: Button “Einzahlen” (I)

- Erzeugen, Platzieren, `ActionListener`, Ereignisbehandlung analog
- Die Funktionalität des Buttons wird in der Methode `einzahlen` der Klasse `BankFrame` umgesetzt

```
private void einzahlen() {  
    ... // Fehlerbehandlung, falls keine Bank eröffnet  
    int kontoNr = Integer.parseInt(  
        JOptionPane.showInputDialog("Kontonummer"));  
    double betrag = Double.parseDouble(  
        JOptionPane.showInputDialog("Betrag"));  
    boolean eingezahlt =  
        this.bank.einzahlen(kontoNr, betrag);  
    if (eingezahlt) this.ausgabeBereich.setText("eingez.");  
}
```



Arrays von Objekten: Button “Einzahlen” (II)

- Einzahlen auf bestimmtes Konto

d.h. Methode `einzahlen` in der Klasse `Bank`

```
public boolean einzahlen(int kontoNr,  
                        double betrag) {  
    BankKonto konto = this.sucheBankkonto(kontoNr);  
    if (konto != null) {  
        konto.einzahlen(betrag);  
        return true;  
    }  
    return false;  
}
```



Arrays von Objekten: Button “Einzahlen” (III)

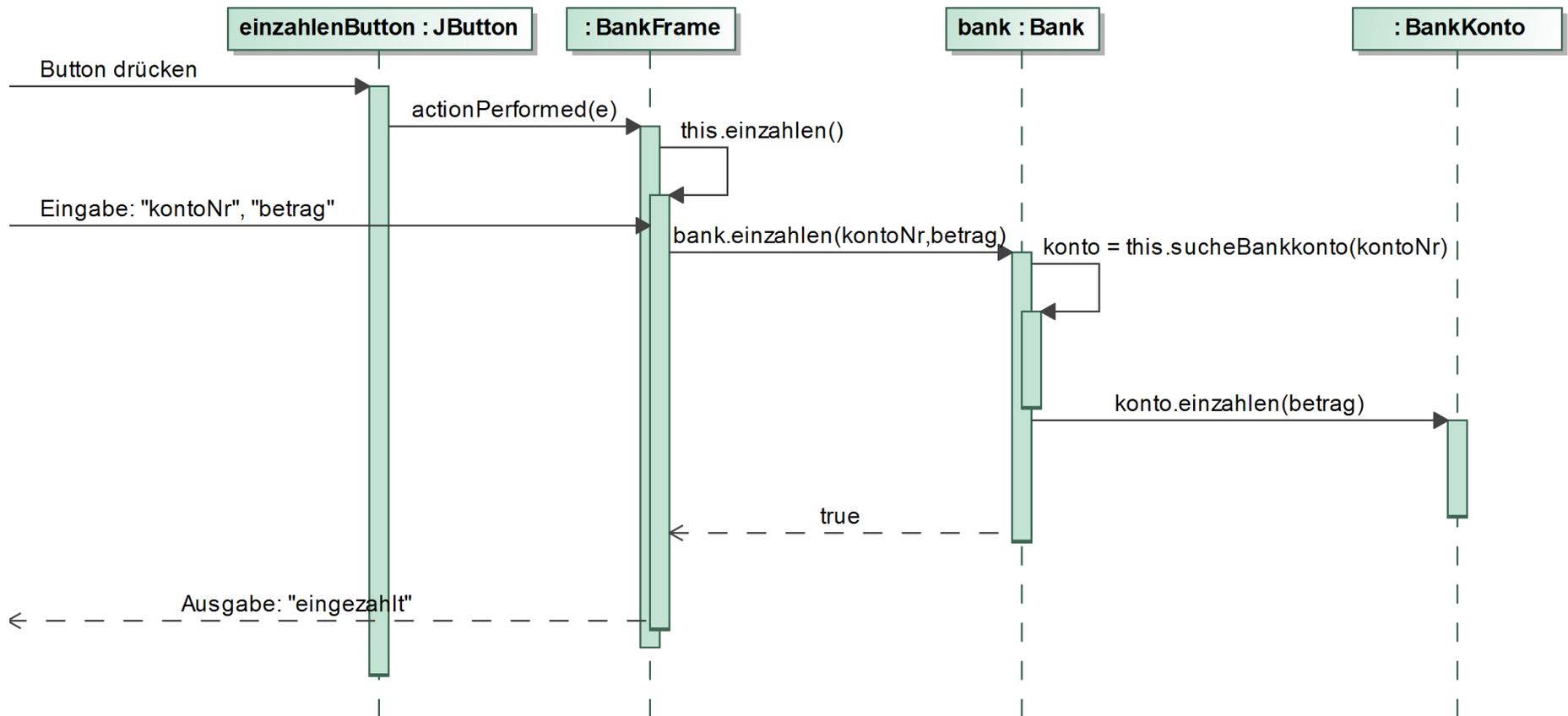
■ Suche eines bestimmten Kontos

d.h. Methode `sucheBankkonto` in der Klasse `Bank`

```
public Bankkonto sucheBankkonto(int kontoNr) {  
    for (int i = 0;  
         i < this.anzahlEroeffneterKonten; i++) {  
        Bankkonto konto = this.konten[i];  
        if (konto.getKontoNummer() = kontoNr) {  
            return konto;  
        }  
    }  
    return null;  
}
```



Arrays von Objekten: Button “Einzahlen” (IV)





Arrays von Objekten: Button “Abheben” & “Kontostand”

Analog zum Button “Einzahlen”



Arrays von Objekten: Button “Gesamtsaldo” (I)

- Erzeugen, Platzieren, `ActionListener`, Ereignisbehandlung analog
- Die Funktionalität des Buttons wird in der Methode `gesamtSaldoBerechnen` der Klasse `BankFrame` umgesetzt

```
private void gesamtSaldoBerechnen() {  
    ... // Fehlerbehandlung, falls keine Bank eröffnet  
    double gesamtSaldo = this.bank.gesamtSaldo();  
    this.ausgabeBereich.setText(  
        "Der Gesamtsaldo ist " + gesamtSaldo);  
}
```



Arrays von Objekten: Button “Gesamtsaldo” (II)

- Gesamtsaldo über alle Konten in der Bank berechnen
d.h. Methode `gesamtSaldo` in der Klasse `Bank`

```
public double gesamtSaldo() {  
    double gesamtSaldo = 0.0;  
    for (int i = 0;  
         i < this.anzahlEroeffneterKonten; i++) {  
        BankKonto aktuellesKonto = this.konten[i];  
        gesamtSaldo = gesamtSaldo  
            + aktuellesKonto.getKontoStand();  
    }  
    return gesamtSaldo;  
}
```