

Arrays von Objekten

Dr. Philipp Wendler

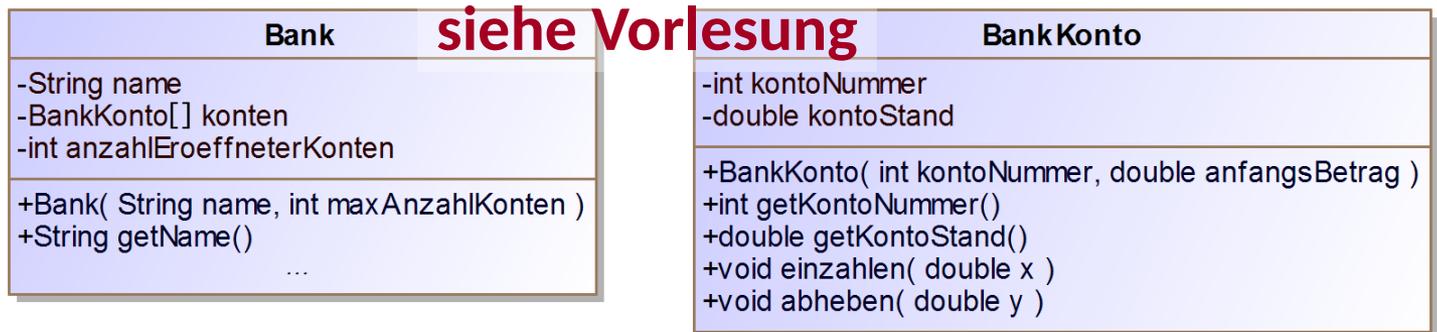
Zentralübung zur Vorlesung

„Einführung in die Informatik: Programmierung und Softwareentwicklung“

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

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.



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	siehe Vorlesung	BankKonto
<pre>-String name -BankKonto[] konten -int anzahlEröffneterKonten +Bank(String name, int maxAnzahlKonten) +String getName() ...</pre>		<pre>-int kontoNummer -double kontoStand +BankKonto(int kontoNummer, double anfangsBetrag) +int getKontoNummer() +double getKontoStand() +void einzahlen(double x) +void abheben(double y)</pre>

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)

**Wiederholung:
partielle Arrays**

Bank	BankKonto
<pre>-String name -BankKonto[] konten -int anzahlEröffneterKonten +Bank(String name, int maxAnzahlKonten) +String getName() ...</pre>	<pre>-int kontoNummer -double kontoStand +BankKonto(int kontoNummer, double anfangsBetrag) +int getKontoNummer() +double getKontoStand() +void einzahlen(double x) +void abheben(double y)</pre>

siehe Vorlesung

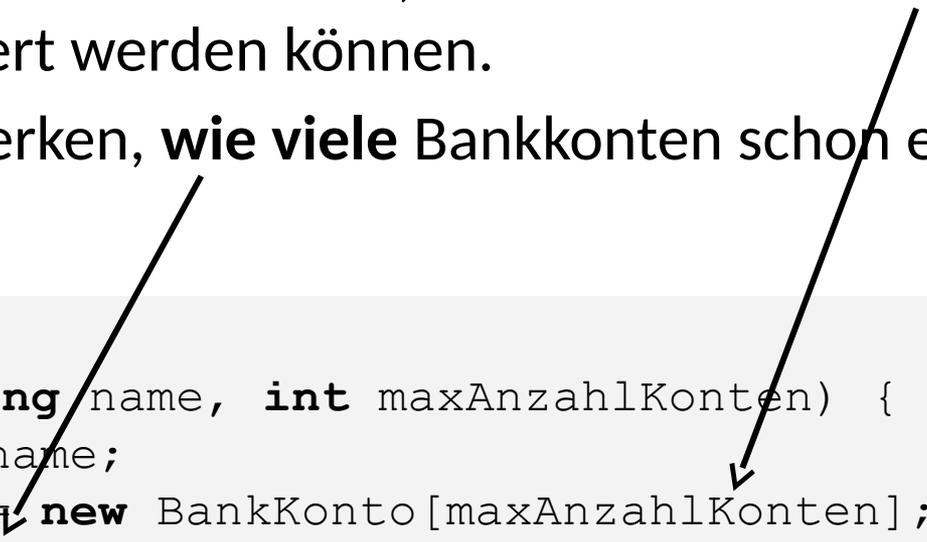
Anzahl aller schon eröffneten Konten

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.

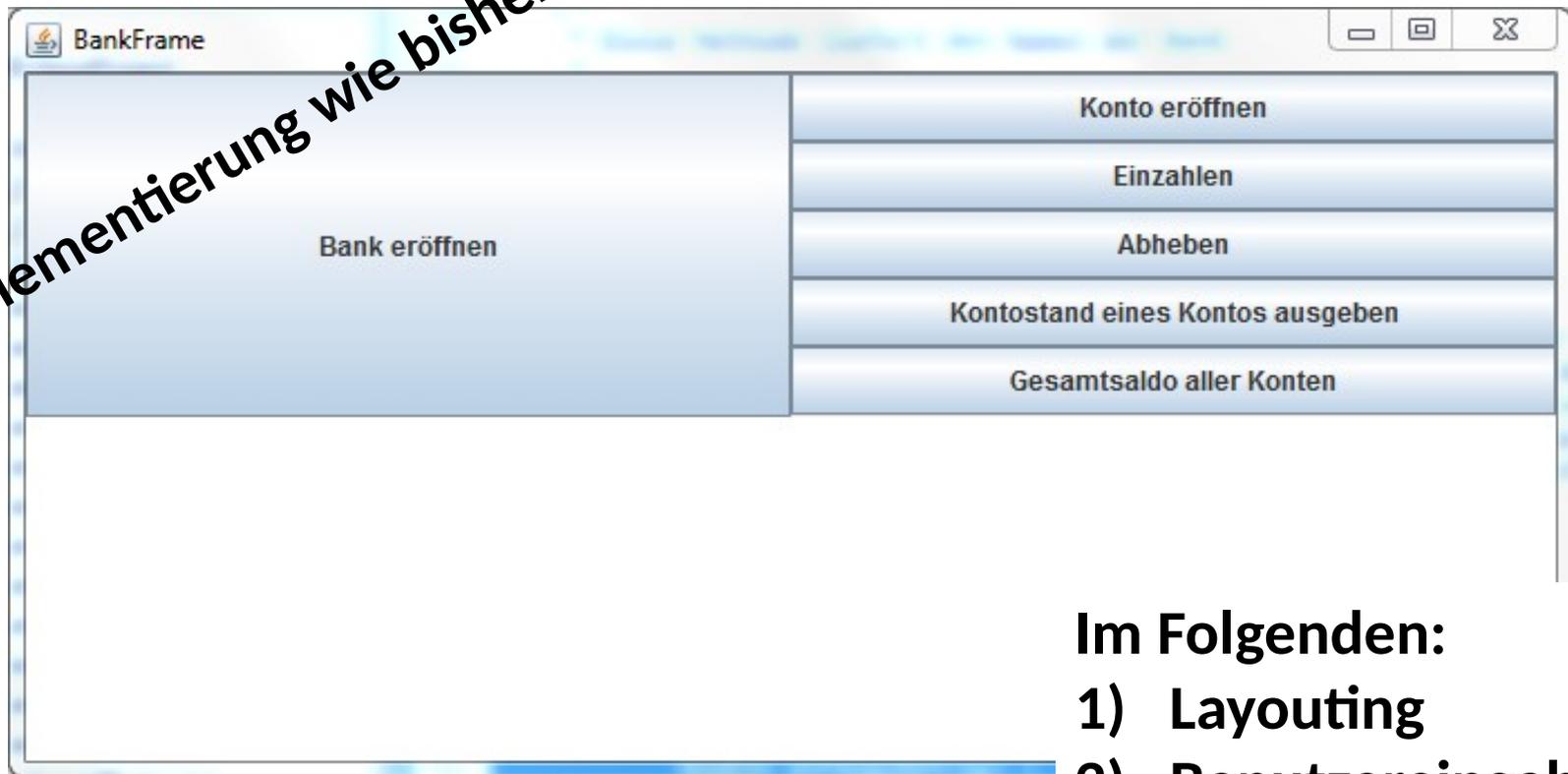
bei Eröffnung eines neuen Bankkontos muss ...

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

```
public boolean kontoEroeffnen(int kontoNr, double betrag) {
    if (this.anzahlEroeffneterKonten < this.konten.length) {
        this.konten[this.anzahlEroeffneterKonten] =
            new BankKonto(kontoNr, betrag);
        this.anzahlEroeffneterKonten++;
        return true;
    }
    return false;
}
```

Arrays von Objekten: Bank-GUI

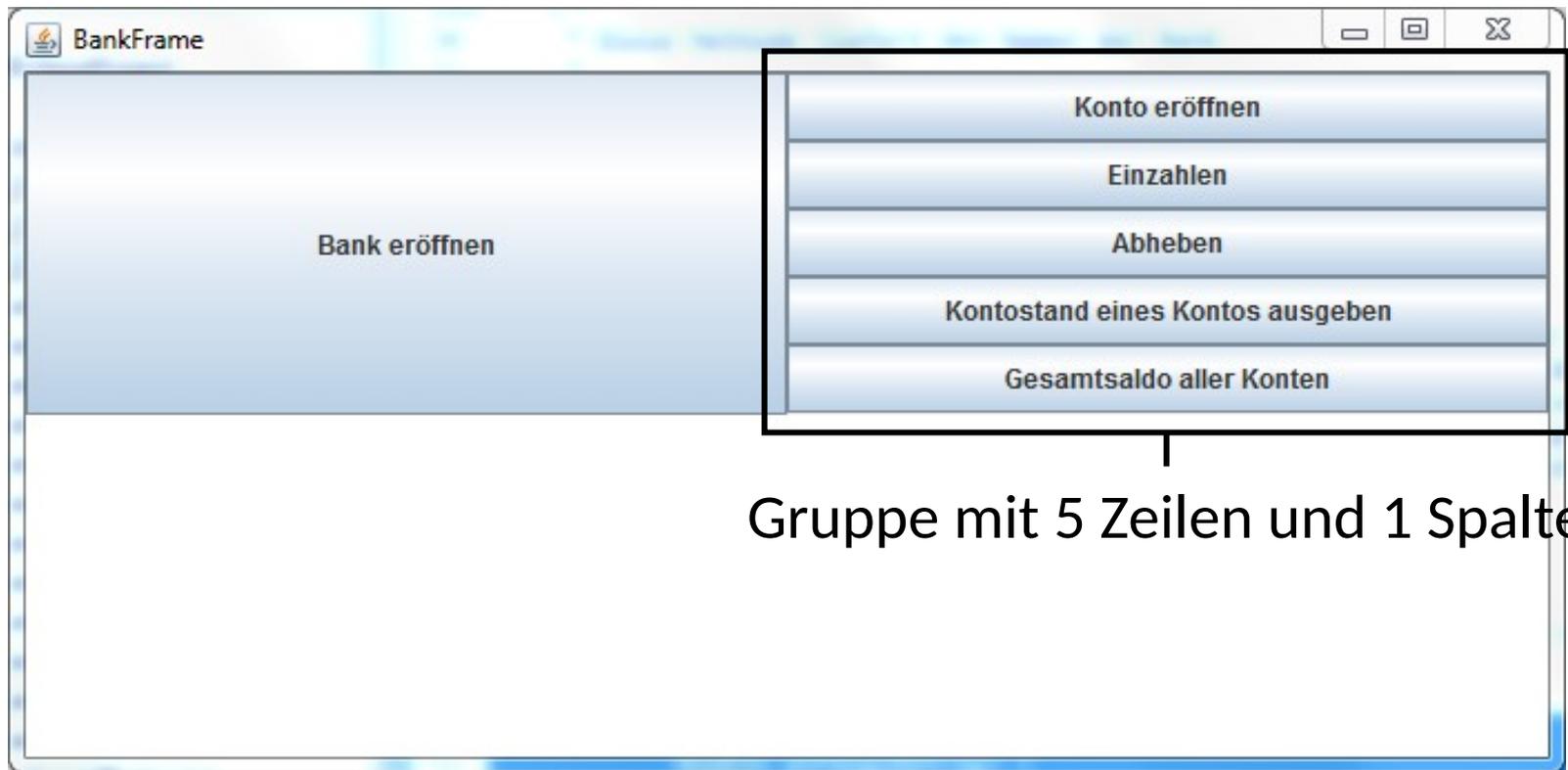
Implementierung wie bisher



Im Folgenden:

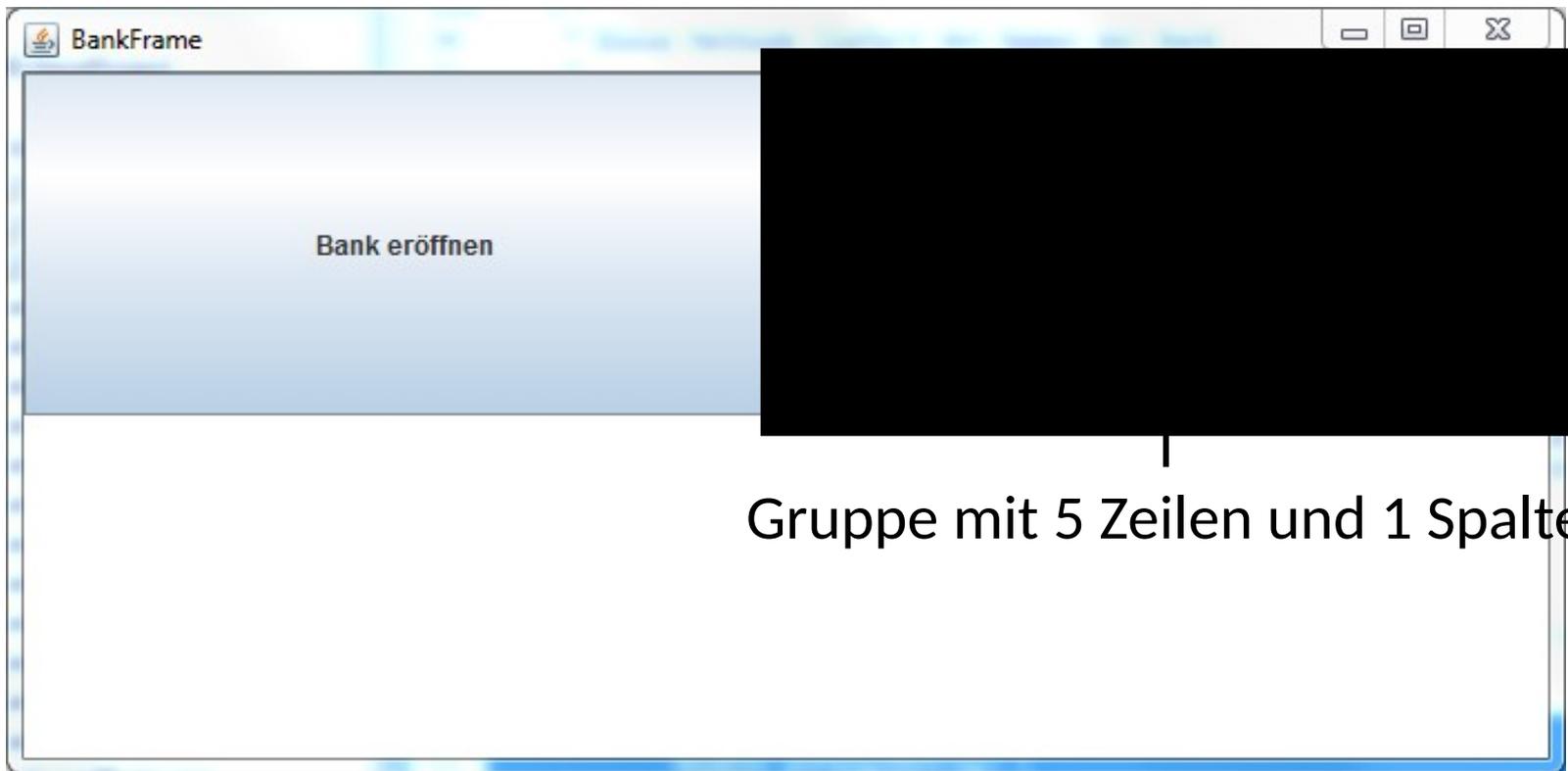
- 1) Layouting**
- 2) Benutzereingaben**

Arrays von Objekten: Bank-GUI (Layouting)



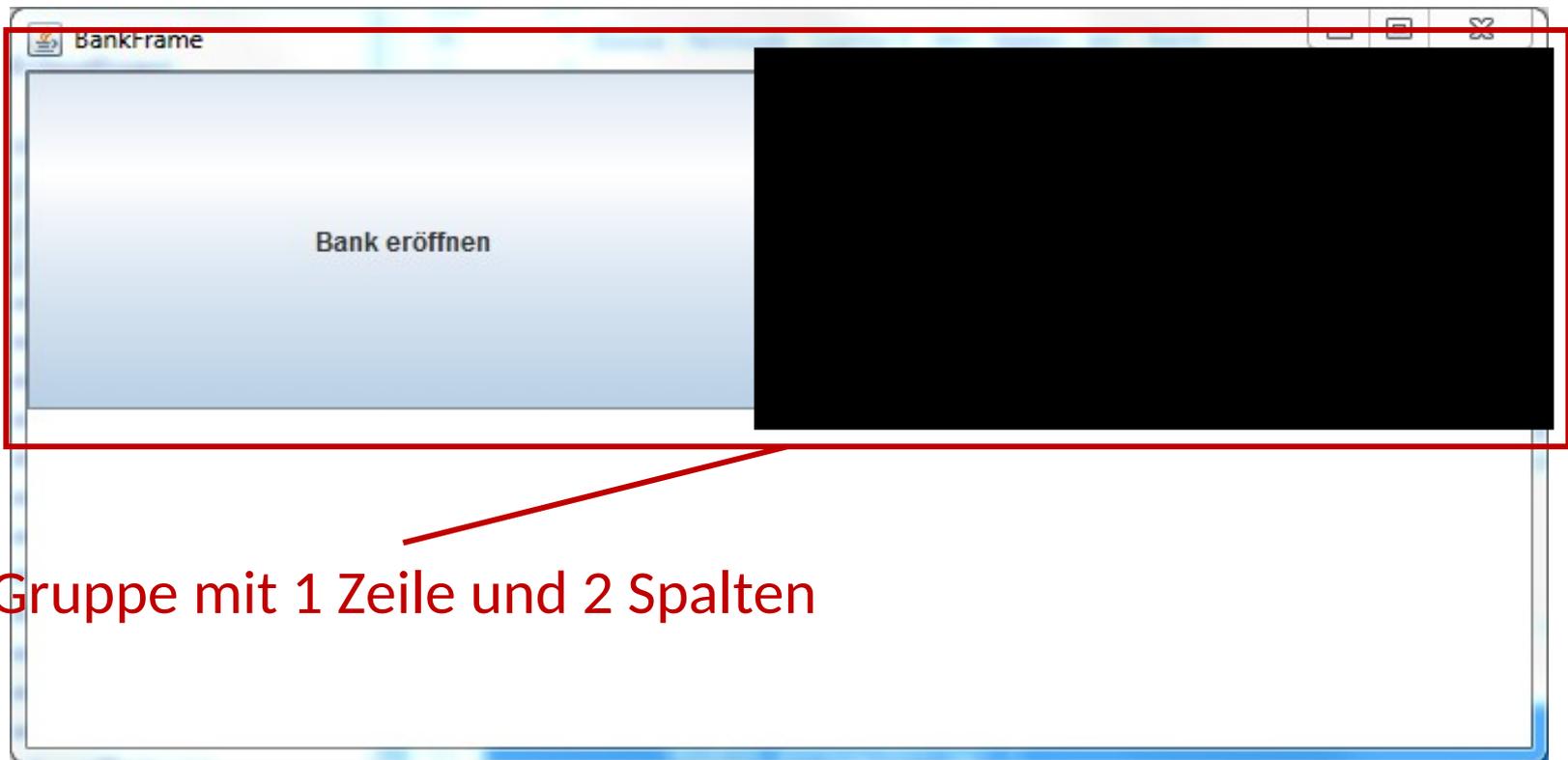
Gruppe mit 5 Zeilen und 1 Spalte

Arrays von Objekten: Bank-GUI (Layouting)



Gruppe mit 5 Zeilen und 1 Spalte

Arrays von Objekten: Bank-GUI (Layouting)



Gruppe mit 1 Zeile und 2 Spalten

Arrays von Objekten: Bank-GUI (Layouting)

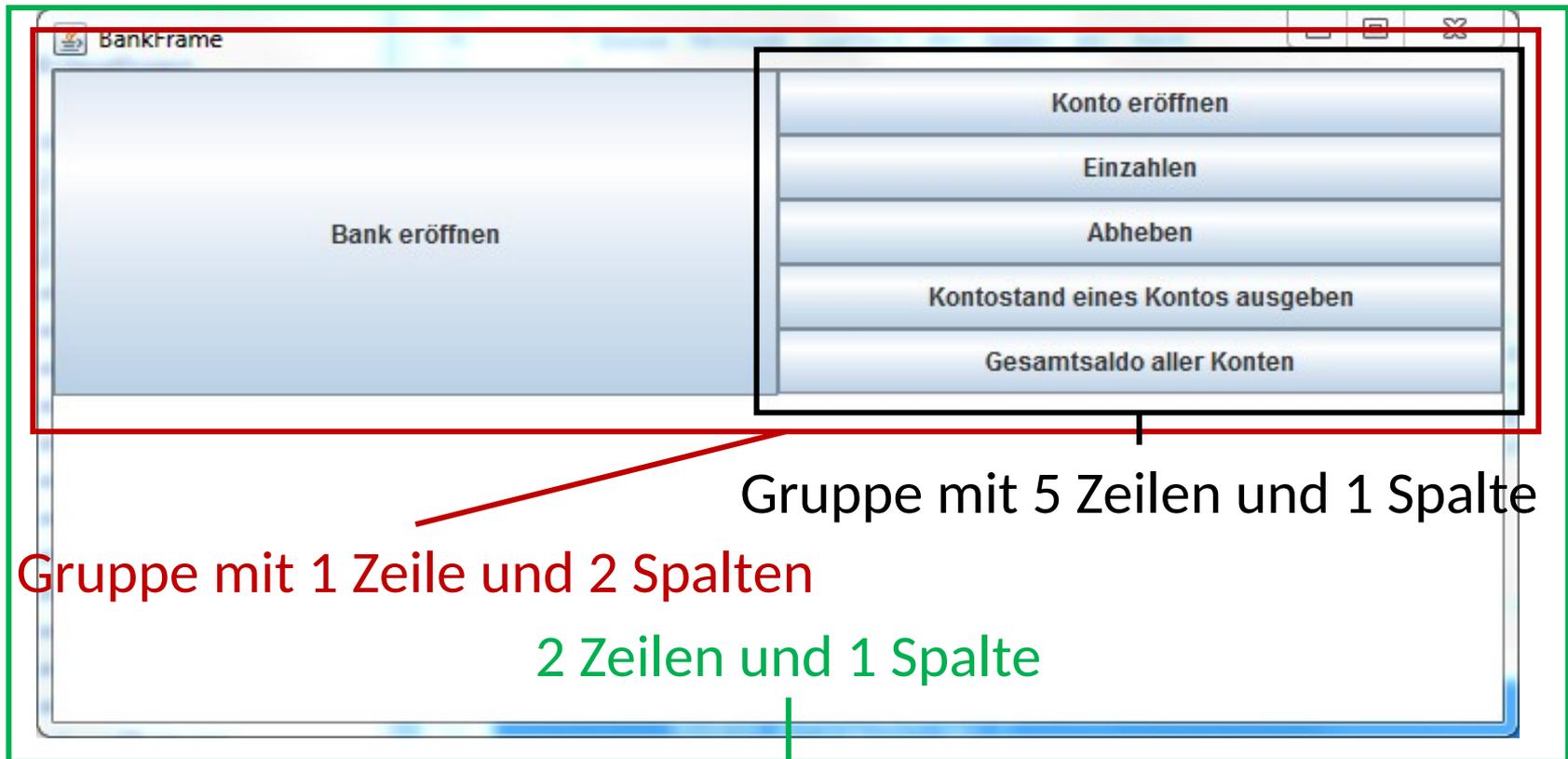


Gruppe mit 1 Zeile und 2 Spalten

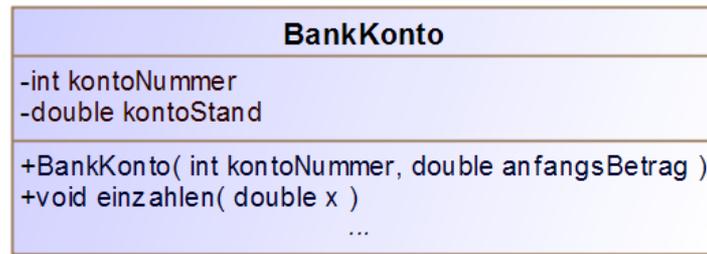
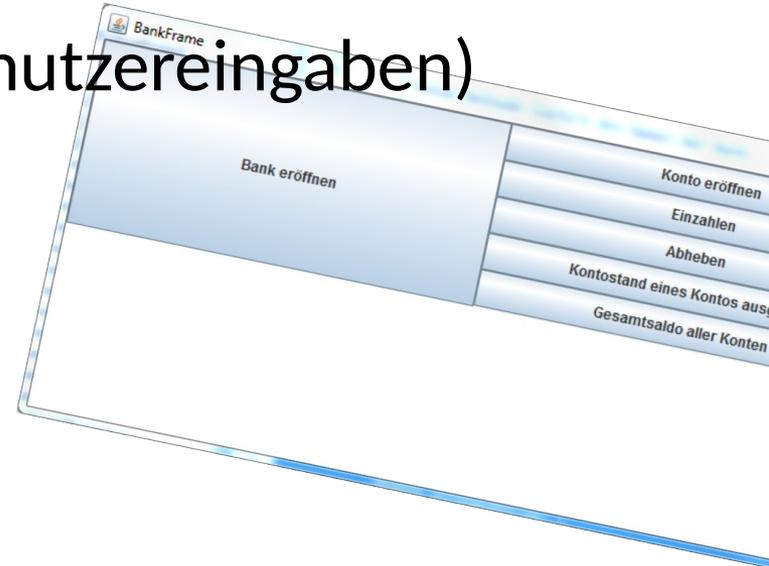
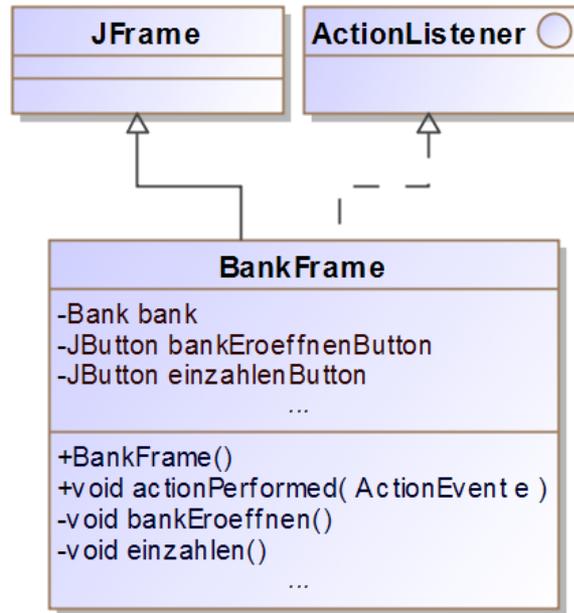
Arrays von Objekten: Bank-GUI (Layouting)



Arrays von Objekten: Bank-GUI (Layouting)



Arrays von Objekten: Bank-GUI (Benutzereingaben)



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 kontoNr,
                               double betrag) {
    if (this.anzahlEroeffneterKonten <
        this.konten.length) {
        this.konten[this.anzahlEroeffneterKonten] =
            new BankKonto(kontoNr, betrag);
        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`

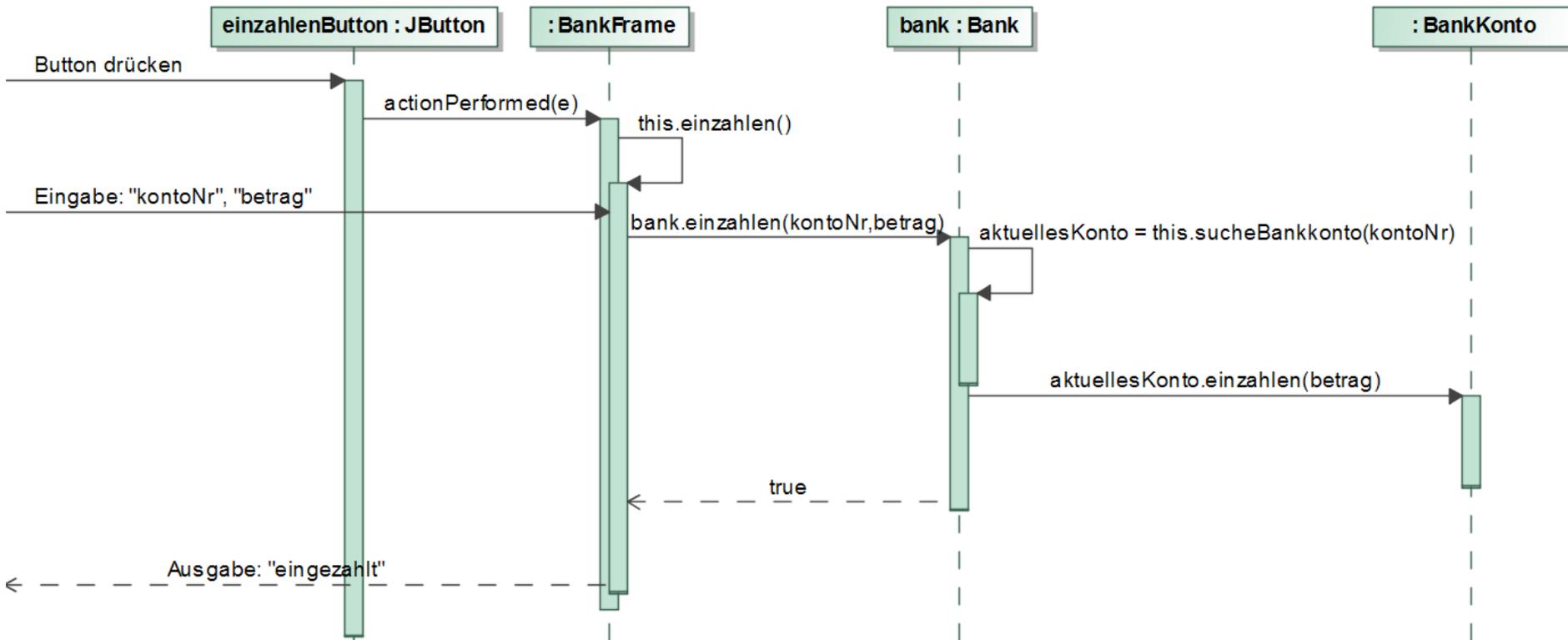
```
public boolean einzahlen(int kontoNr,  
                        double betrag) {  
    BankKonto konto = this.sucheBankkonto(kontoNr);  
    if (konto != null) {  
        konto.einzahlen(betrag);  
        return true;  
    }  
    return false;  
}
```

Arrays von Objekten: Button “Einzahlen” (III)

- Einzahlen auf ein Konto
d.h. Methode `einzahlen` in der Klasse `Konto`

```
public void einzahlen(double x) {  
    this.kontoStand = this.kontoStand + x;  
}
```

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;
}
```

Arrays von Objekten: Verwendung von partiellen Arrays

Was passiert bei der Ausführung des Programmstücks?

```
for (int i = 0; i < this.konten.length; i++) {  
    BankKonto aktuellesKonto = this.konten[i];  
    gesamtSaldo = gesamtSaldo  
        + aktuellesKonto.getKontoStand();  
}
```

- a) Das Programm terminiert nicht.
- b) Der Wert von `gesamtSaldo` wird falsch berechnet.
- c) Das Programm wird mit einem Fehler abgebrochen.