Objektorientierter Entwurf

1. Finden von Klassen (Faustregel: Substantive in der Anforderungsbeschreibung)
2. Ermitteln des Verhaltens jeder Klasse
3. Beschreiben der Beziehungen zwischen den Klassen

Beispiel: Rechnungen schreiben

```
R E C H N U N G

Johanna Musterfrau
Hauptstr. 78
85555 Obing

Posten                  Anzahl      Preis    Gesamtpreis
-------------------------------------------------------
Toaster                      3      29.95          89.85
Haartrockner                 1      24.95          24.95
Autostaubsau                 2      19.99          39.98

Rechnungsbetrag: EUR154.78
```

Finden der Klassen

Substantive: Rechnung, Kunde, Posten, Anzahl, Preis, Gesamtpreis, Rechnungsbetrag, Toaster, Fön, Autostaubsauger.

Welche eignen sich für Klassen?

Finden der Klassen

Anzahl, Preis, Gesamtpreis, Rechungsbetrag sind reine Zahlenwerte. Für sie benötigen wir keine Klassen

Toaster, Fön, Autostaubsauger sind Posten, auch für sie brauchen wir keine Klassen.

Es verbleiben: Rechnung, Posten, Kunde.

CRC-Karten

Für jede Klasse kann man eine Karteikarte anlegen, die vermerkt:

- Klassennamen
- Methoden
- Kollaborierende Klassen

Der Klassenname steht oben.

Jede Methode kommt in eine Zeile gefolgt von den kollaborierenden Klassen.

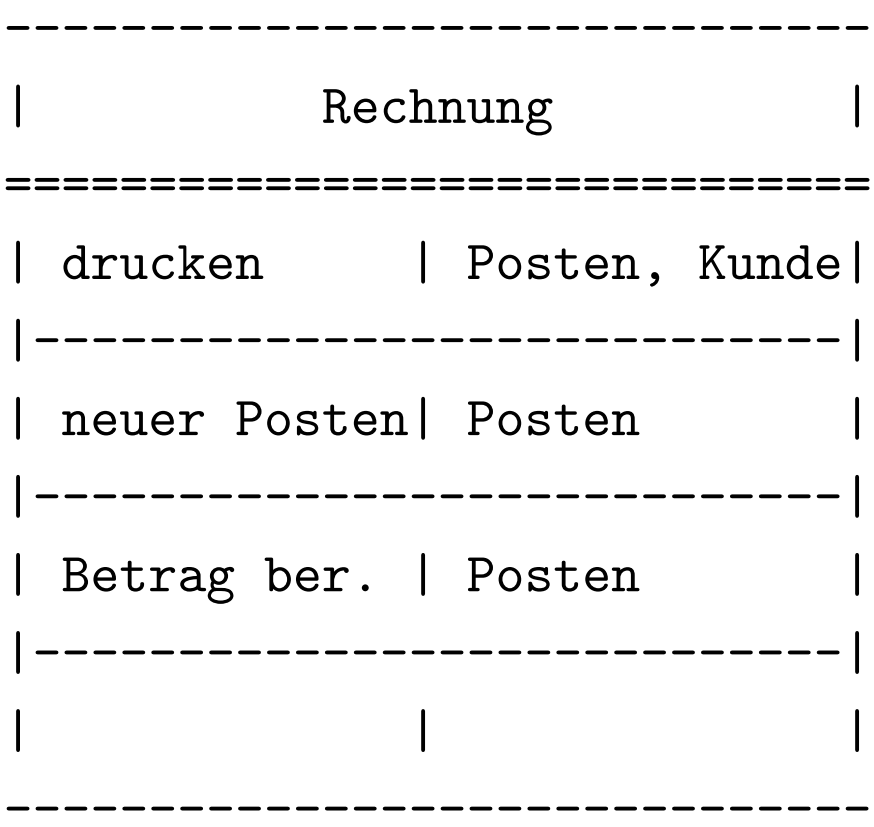
$\rightsquigarrow$ CRC-Karte (*classes*, *responsibilities*, *collaborators*).

Kollaboration

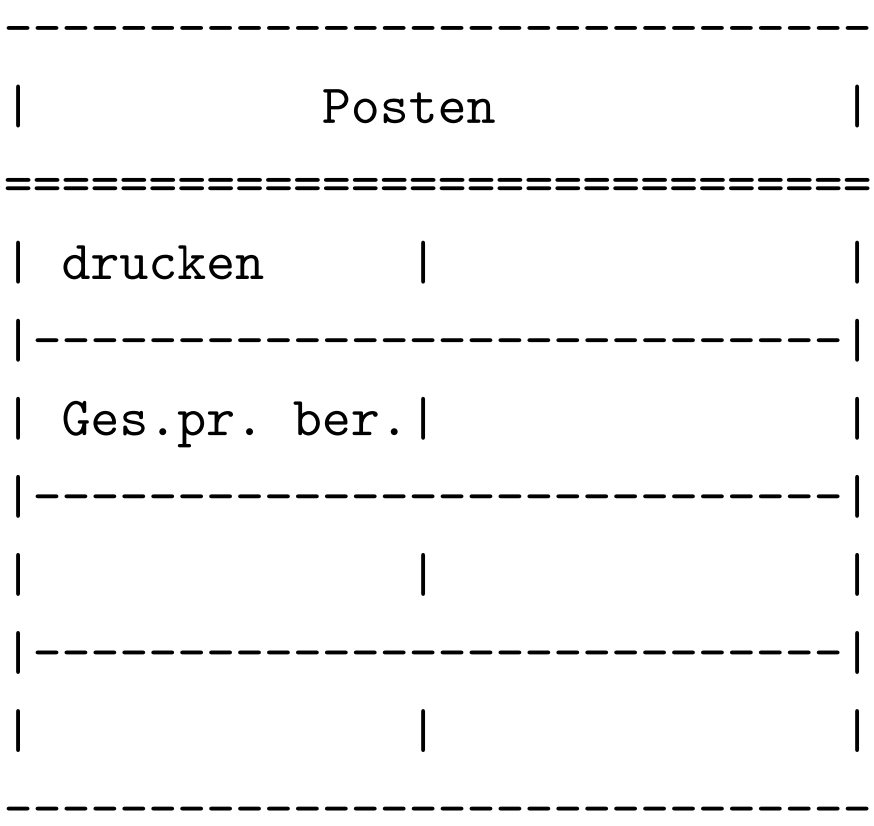
Eine Klasse kollaboriert mit einer Methode, wenn ein Objekt der Klasse

- als Parameter übergeben wird
- als Instanzvariable verwendet wird
- im Methodenrumpf von einer anderen Methode zurückgegeben wird
- im Methodenrumpf neu erzeugt wird

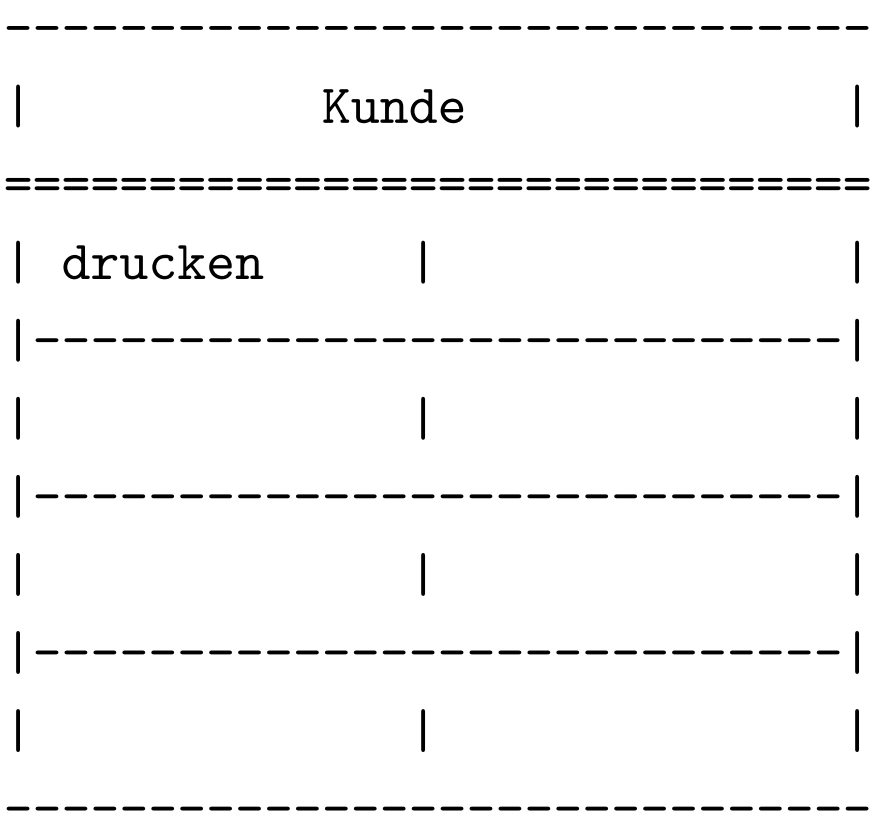
Beispiel: Rechnung



Beispiel: Posten

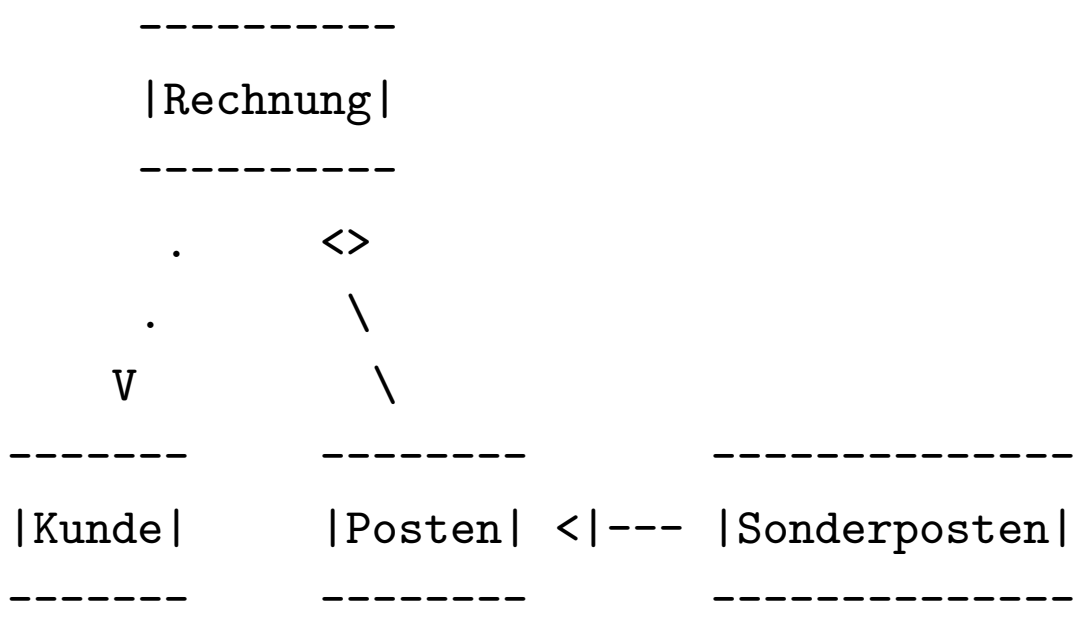


Beispiel: Kunde



Beziehungen zwischen den Klassen

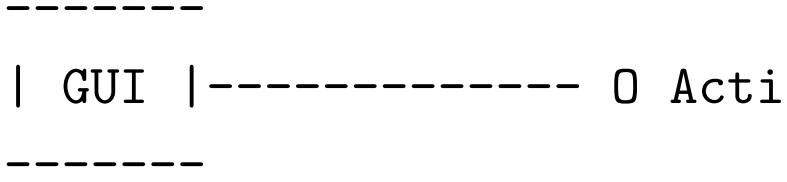
- Vererbung (“ist ein”). Beispiel: Van-Pkw. UML: `---|>`.
- Aggregation (“hat ein”). Beispiel: Polygon-Strecke, Rechnung-Posten. UML: `<>---`.
- Benutzung. Beispiel: Rechnung-Kunde. UML: `- - - >`.



Schnittstellen

Sonderfall: Implementierung einer Schnittstelle. Beispiel: GUI-ActionListener, Bankkonto-Comparable.

In UML werden Schnittstellen als Kreise dargestellt und mit den implementierenden Klassen verbunden.



Attribute

Ein Attribut einer Klasse ist ein Wert, der einem Objekt der Klasse zugeordnet ist.

Die Gesamtheit der Attribute bestimmt den Zustand eines Objekts.

Attribute können in Javaauf zwei Arten realisiert werden:

- Instanzvariablen mit get-set-Methoden. Beispiel: Kontostand.
- Berechung aus anderen Instanzvariablen. Beispiel: Kontostand in EUR, falls intern als ganzzahliger Cent-Betrag gespeichert.

Ob ein berechneter Wert Attribut ist oder nicht, richtet sich nach dem Sinn und der Anwendung.

Grenzfall: Flächeninhalt eines Rechtecks.

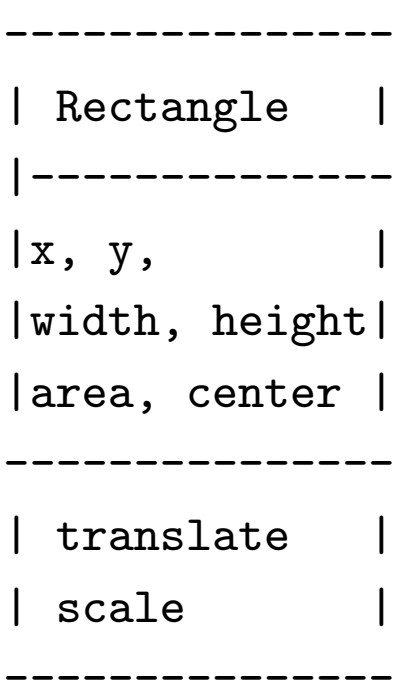
Klassen in UML

Klassen werden in UML als dreigeteilte Rechtecke dargestellt.

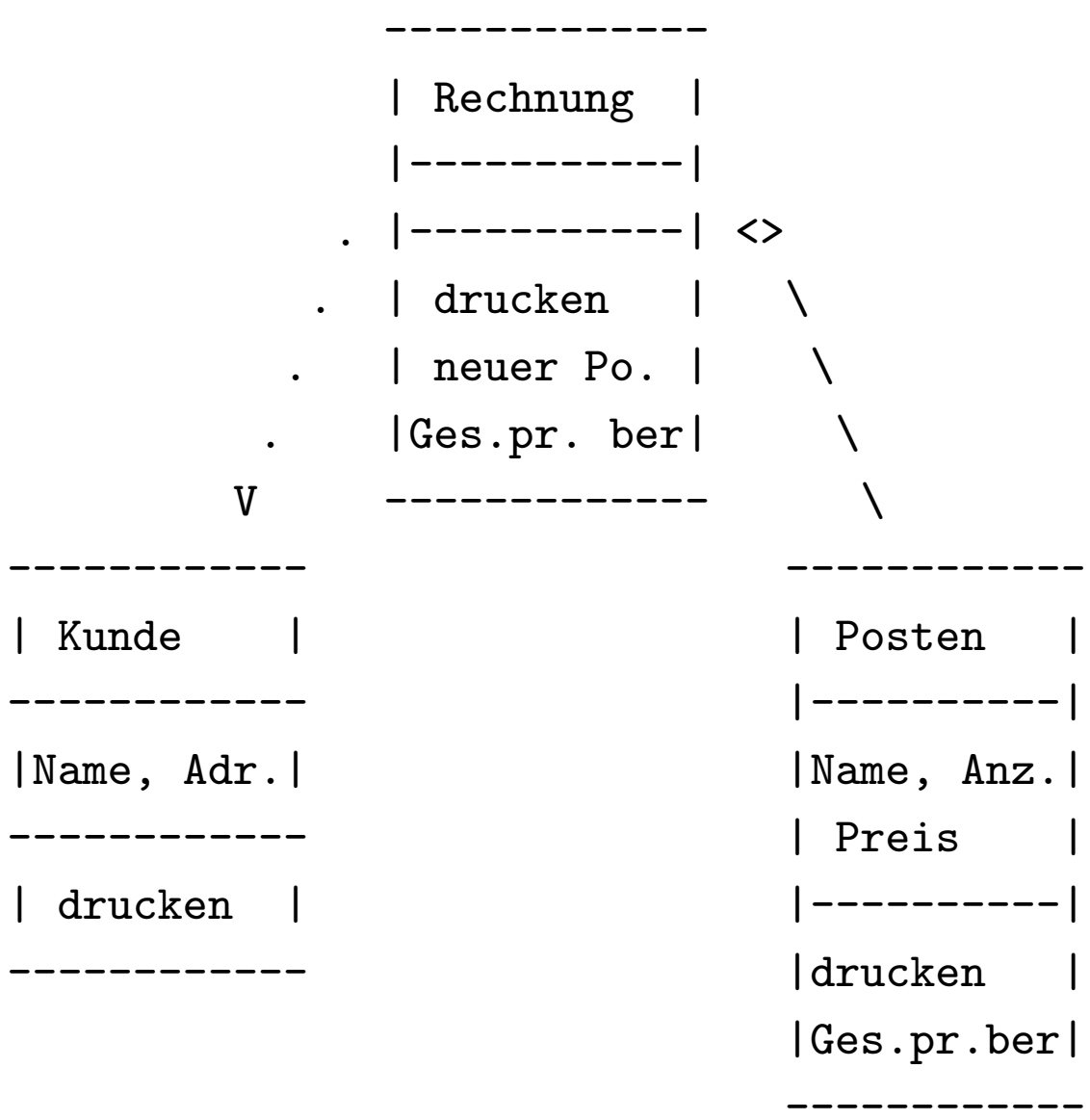
Der oberste Teil enthält den Klassennamen.

Der mittlere Teil enthält die Attribute.

Der untere Teil enthält die Methodennamen.



Beispiel


Rechnung
drucken
neuer Po.
Ges.pr. ber
<>
Kunde
Name, Adr.
drucken
Posten
Name, Anz.
Preis
drucken
Ges.pr.ber

Redundanz

Merke:

Klassen, die durch `---<>` (Aggregation) gekennzeichnet sind, werden nicht als Attribute gelistet und auch nicht mit `- - ->` (Verwendung) bezeichnet.

Methodendokumentation

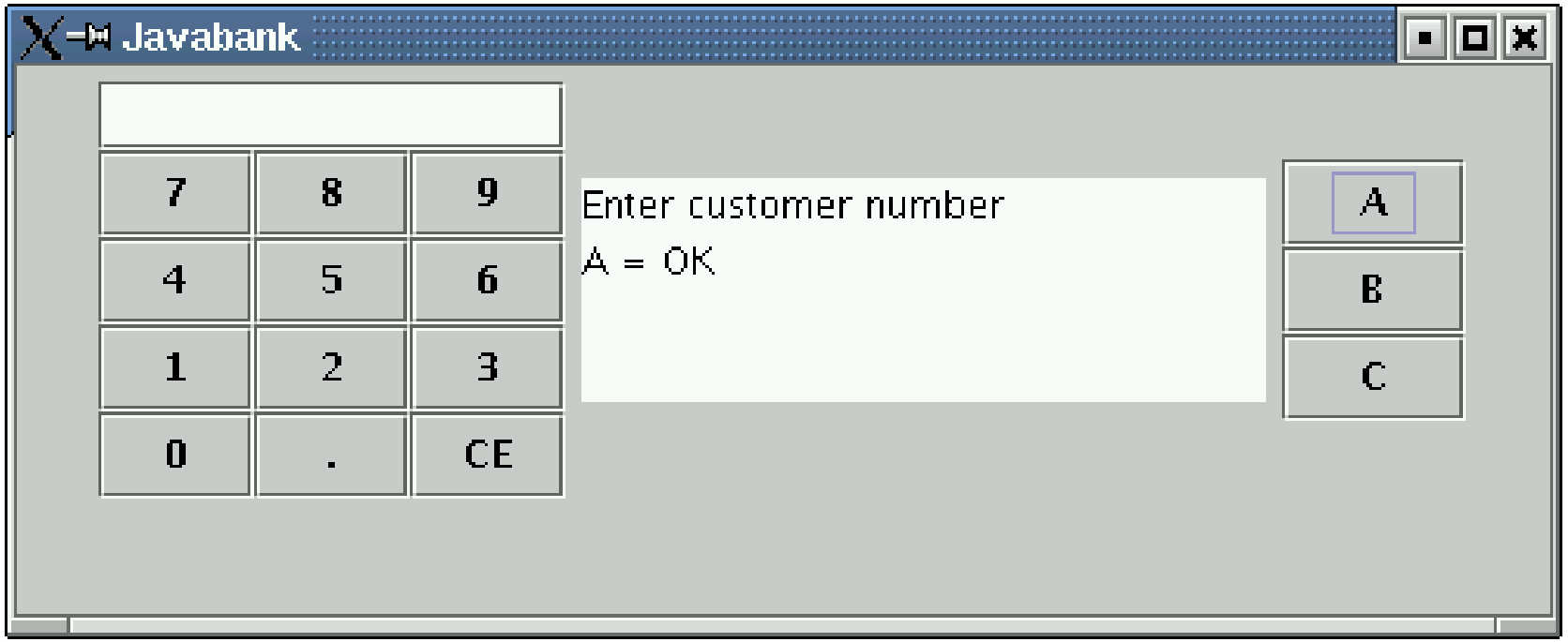
Für jede Klasse eine Datei mit leeren Methodenrümpfen und javadoc-Kommentaren.

```
package rechnung;

public class Rechnung {
    private Kunde kunde;
    private Posten[] diePosten;

    /** Hinzufuegen eines neuen Postens.
        @param posten hinzugefuegter Posten.
    */
    public void neuerPosten(Posten posten) {}

    /** Druckt die Rechnung. */
    public void print()     {}
```

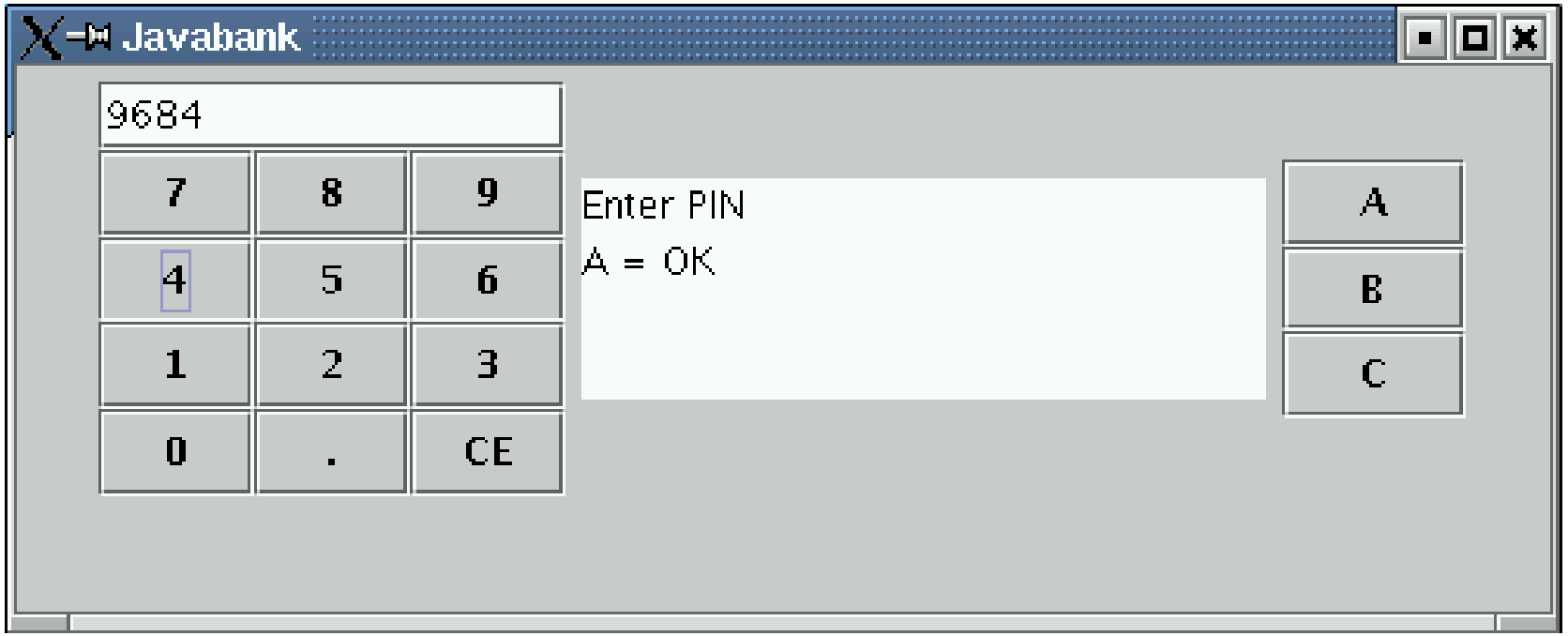
Methodendokumentation

```
    /** Berechnet den Rechungsbetrag.
        @return den Rechungsbetrag
    */
    public double berechneBetrag()
    {return 0;}
}
```

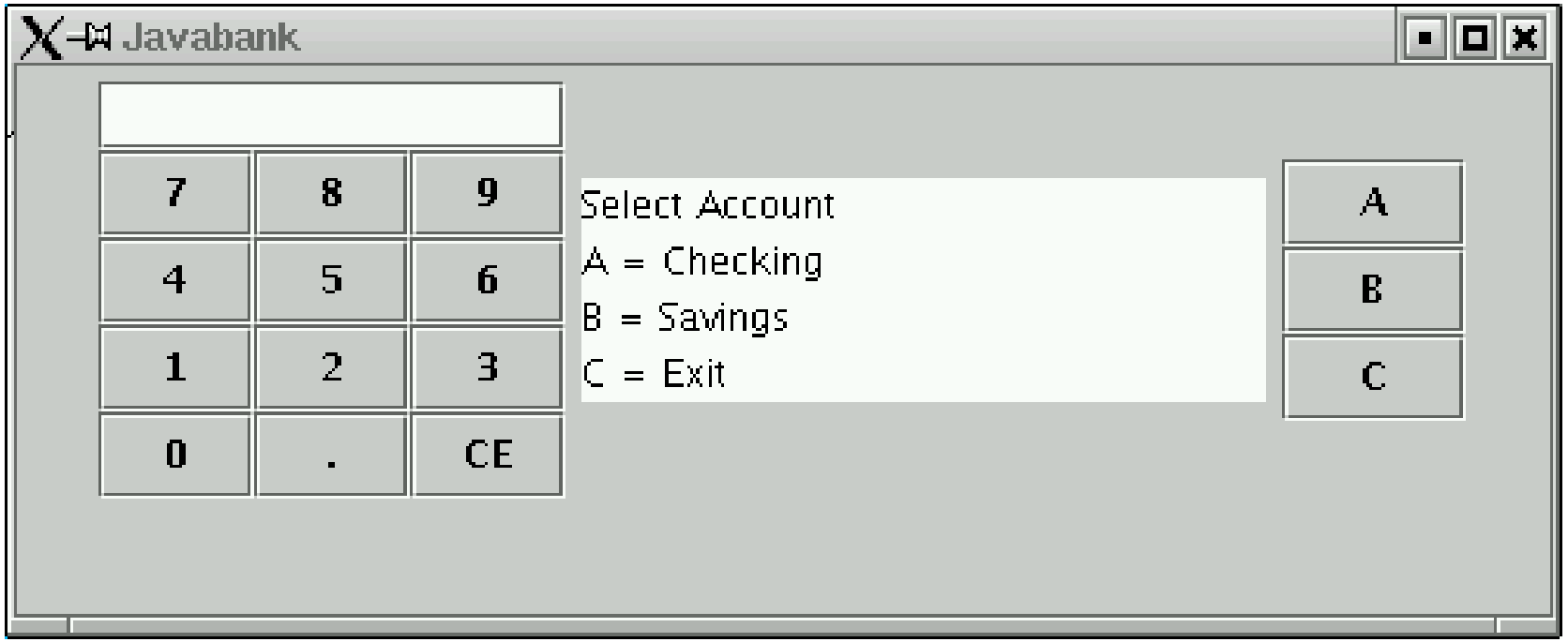
Fallstudie: Geldautomat



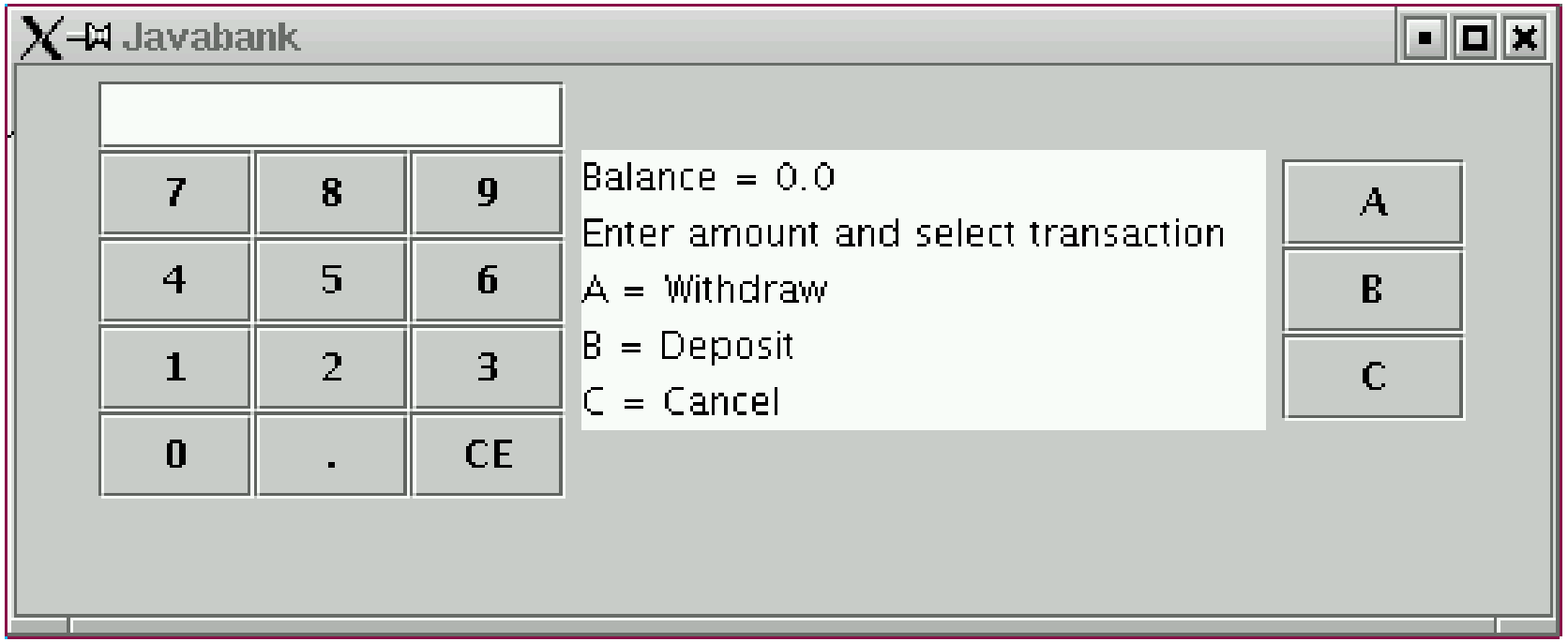
Fallstudie: Geldautomat



Fallstudie: Geldautomat



Fallstudie: Geldautomat



Analyse

Der Automat gehört einer Bank, die eine Anzahl von Kunden hat.

Jeder Kunde hat ein Spar- und ein Girokonto, eine Kundennummer und eine PIN.

Der Automat hat ein Zahlenfeld, zwei Textfelder und drei Knöpfe A, B, C.

Kunden geben ihre Nummer ein und anschließend die PIN.

Stimmen sie überein, so wird über die Knöpfe ein Konto ausgewählt oder abgebrochen.

Es wird der Kontostand angezeigt und die Möglichkeit geboten, abzuheben, einzuzahlen oder abzubrechen.

Fortsetzung der Fallstudie an der Tafel (siehe auch Buch Ch.14).

UML-Zustandsdiagramme

Oft wechselt ein Objekt gemäß der Eingaben oder Ereignisse durch mehrere Zustände.

Beispiel: Geldautomat hat vier Zustände (Kundennummer erwarten, PIN erwarten, Konto wählen, Transaktion erwarten.)

Die Übergänge zwischen den Zuständen lassen sich in einem UML-Zustandsdiagramm darstellen (*state chart*).

Abgerundete Rechtecke symbolisieren Zustände.

Mit Aktionen beschriftete Pfeile `----->` symbolisieren Zustandsübergänge.