

## 5 Projekt Bankverwaltung

### 5.1 Festlegen der Schnittstelle

Bevor du mit der Programmierung beginnst, musst du dir einige Gedanken über die Schnittstelle zwischen der grafischen Benutzeroberfläche und der eigentlichen Bankverwaltung machen.

Abbildung 5.1 zeigt eine mögliche GUI, die mit einigen kleinen Änderungen aus Kapitel 3 übernommen wurde.



**Abbildung 5.1:** Mögliche Oberfläche des Projekts *Bankverwaltung* mit den drei Registerkarten *geld*, *suchen* und *neu*

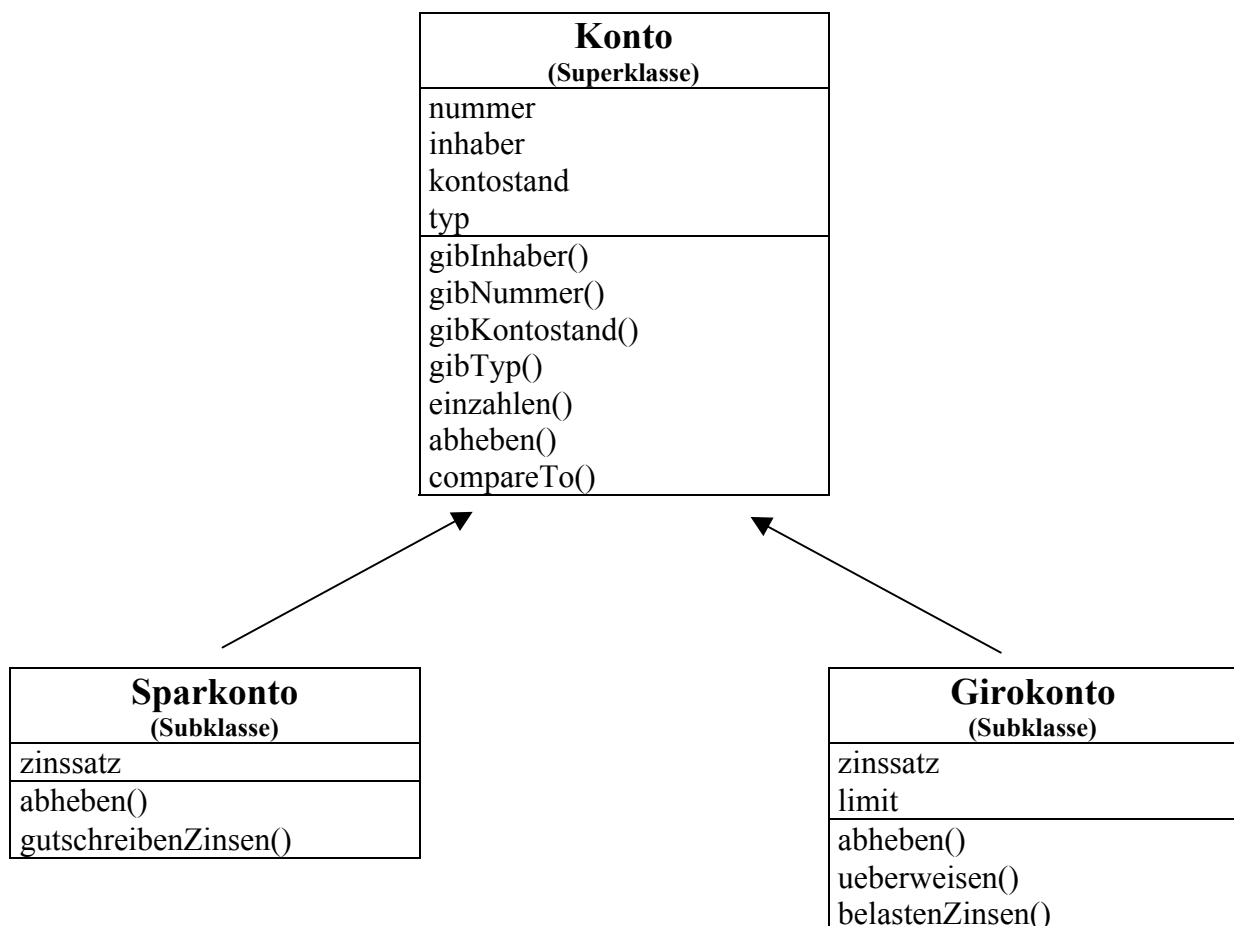
Das gelbe Hauptpanel *mainP* ist unterteilt in das

- hellblaue Navigationspanel *navigationP*,
- dunkelblaue Personpanel *personP*,
- hellrote Geldpanel *geldP* in der Registerkarte *geld*,

- d) hellrote Suchenpanel *suchenP* in der Registerkarte *suchen*,
- e) hellrote Neupanel *neuP* in der Registerkarte *neu*,
- f) hellblaue Buchungspanel *buchP*.

Dieser Vorschlag zeigt bereits, in welche Klassen das Projekt *Bankverwaltung* unterteilt werden muss.

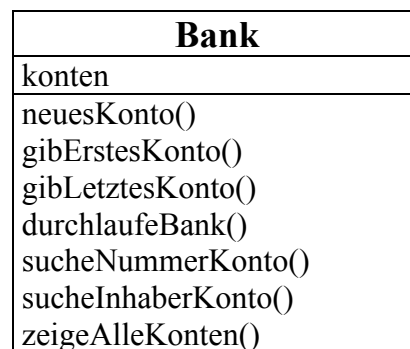
Um lästige Code-Duplizierung zu vermeiden, benötigst du zwei Subklassen *Sparkonto* und *Girokonto*, die von einer abstrakten Superklasse *Konto* erben. Diese Klassen verwenden die Datenfelder *inhaber*, *nummer*, *kontostand* und *typ*, die im *personP* dargestellt werden. Weiterhin besitzen sie die Methoden *einzahlen()*, *abheben()*, *ueberweisen()* und *zinsen()*, die in der Registerkarte *geld* verwendet werden.



**Abbildung 5.2:** *Sparkonto* und *Girokonto* erben von *Konto*

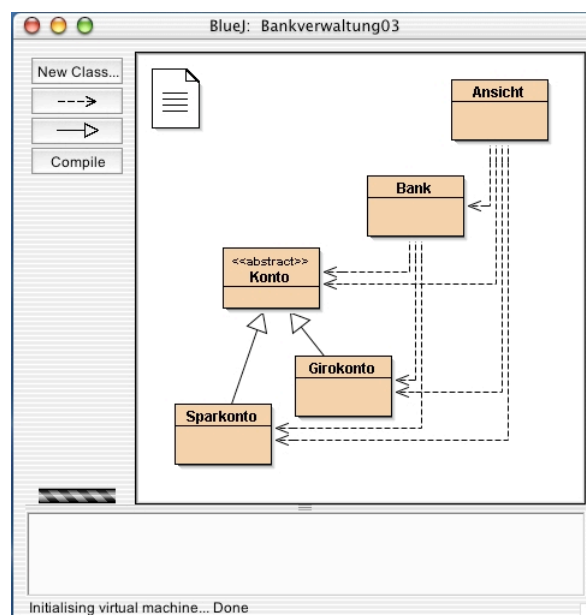
Zur Verwaltung aller Konten benötigst du die Klasse *Bank*. Diese Klasse besitzt als Datenfeld die Sammlung *konten* vom Typ *TreeSet*, die alle Konten in einer sortierten Form sammelt. Weiterhin benötigst du die Methoden

*gibErstesKonto()*, *gibLetztesKonto()* und *durchlaufeBank()*, um mit Hilfe der Buttons im *navigationP* innerhalb der Sammlung zu navigieren. Der Button *ueberweisen* in der Registerkarte *geld* verwendet die Nummer des Empfängerkontos, die mit Hilfe der Methode *sucheNummerKonto()* gesucht wird. Die Registerkarte *suchen* verwendet der Methode *sucheInhaberKonto()*, um nach allen Konten eines bestimmten Kunden zu suchen, und die Registerkarte *neu* verwendet die Methode *neuesKonto()*, um weitere Konten zu eröffnen. Zum Speichern und Drucken brauchst du noch die Methode *zeigeAlleKonten()*, die sämtliche Eigenschaften aller Konten in einem String ausgibt.



**Abbildung 5.3:** Die Klasse *Bank*

In der Klasse *Ansicht* kannst du nun mit Hilfe der Buttons bzw. Radiobuttons auf die öffentlichen Methoden der oben beschriebenen Klassen zugreifen und somit die entsprechenden Textfelder füllen bzw. löschen.



**Abbildung 5.4:** Das Klassendiagramm des Projekts *Bankverwaltung03*

Abbildung 5.4 zeigt das Klassendiagramm des Projekts *Bankverwaltung*.

## 5.2 Die Klassen *Konto*, *Sparkonto* und *Girokonto*

Die Klasse *Konto* kannst du aus *Kapitel 6: Vererbung* aus dem Skript der 10. Jahrgangsstufe mit kleinen Veränderungen übernehmen.

```
public abstract class Konto implements Comparable
{
    private String inhaber;
    private int nummer;
    protected int kontostand; //somit auch den Subklassen bekannt
    protected String typ; //somit auch den Subklassen bekannt

    ..... weitere Methoden vgl. Abbildung 5.2 .....

    /** Die Konten werden zuerst nach Inhaber, Typ und Nummer sortiert. */
    public int compareTo(Object jenes)
    {
        Konto jenesKonto = (Konto) jenes;
        int vergleich = inhaber.compareTo(jenesKonto.gibInhaber());
        if (vergleich != 0) {
            return vergleich;
        }
        vergleich = typ.compareTo(jenesKonto.gibTyp());
        if (vergleich != 0) {
            return vergleich;
        }
        return (" " + nummer).compareTo(" " + jenesKonto.gibNummer());
    }
}
```

**Abbildung 5.5:** Quelltext der Klasse *Konto*

In der Klasse *Bank* werden die Sparkonten und Girokonten als Typ *Konto* in der Sammlung *konten* vom Typ *TreeSet* zusammengefasst und sortiert. Deswegen werden in der Klasse *Konto*

- das *Comparable*-Interface und die Methode *compareTo()* implementiert,
- die Datenfelder *kontostand* und *typ* als *protected* deklariert, da du in dieser Sammlung auf die Subtypen Spar- und Girokonto zugreifen kannst und somit diese Datenfelder zugänglich sind.

In der Klasse *Sparkonto* sind zwei Konstruktoren implementiert. Der erste Konstruktor wird zur Erzeugung eines neuen Kontos benötigt. Der zweite wird verwendet, um eine bereits gespeicherte Datei zu öffnen.

```
public class Sparkonto extends Konto
{
    private int zinssatz = 5;

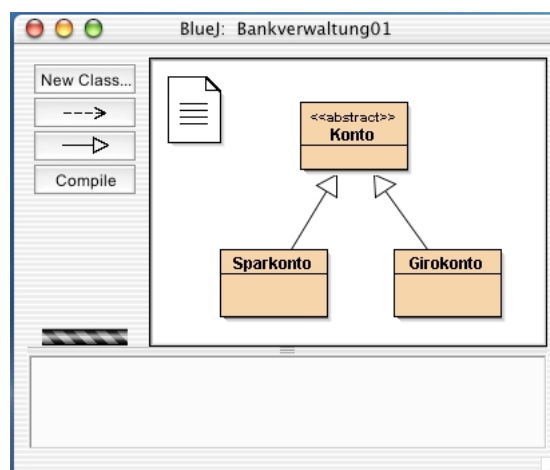
    /** Konstruktor wird zum Erzeugen eines neuen Kontos benoetigt. */
    public Sparkonto(String inhaberH, int nummerH)
    {
        super(inhaberH, nummerH);
        kontostand = 0;
        typ = "Spar";
    }

    /** Konstruktor wird zum Oeffnen der gespeicherten Konten benoetigt. */
    public Sparkonto(String inhaberH, int nummerH, int kontostandH)
    {
        super(inhaberH, nummerH);
        kontostand = kontostandH;
        typ = "Spar";
    }

    ..... weitere Methoden vgl. Abbildung 5.2 .....
}
```

**Abbildung 5.6:** Quelltext der Klasse *Sparkonto*

Analog musst die Klasse *Girokonto* implementieren.



**Abbildung 5.7:** Klassendiagramm des Projekts *Bankverwaltung01*

### 5.3 Die Klasse Bank

Die Klasse *Bank* gleicht weitgehend der Klasse *Telefonbuch* aus Kapitel 2: „Entwurf eines einfachen Telefonbuchs“. Die Methoden *neuesKonto()*, *gibErstesKonto()*, *gibLetztesKonto()*, *durchlaufeBank()*, *sucheInhaberKonto()* und *zeigeAlleKonten()* werden aus der Klasse *Telefonbuch* übernommen und deren Bezeichnungen angepasst. Neu zu implementieren ist lediglich die Methode *sucheNummerKonto()*.

```
/** Sucht nach dem Konto, dessen Kontonummer uebergeben wird. */
public Konto sucheNummerKonto(int kontonummer)
{
    ArrayList gefundeneKonten = new ArrayList();

    Iterator it = konten.iterator();
    while(it.hasNext()) {
        Konto konto = (Konto) it.next();
        int nummer = konto.gibNummer();
        if (nummer == kontonummer) {
            gefundeneKonten.add(konto);
        }
    }
    //in gefundeneKonten gibt es nur dieses Konto
    Konto konto = (Konto) gefundeneKonten.get(0);
    return konto;
}
```

Abbildung 5.8: Quelltext der Methode *sucheNummerKonto()*

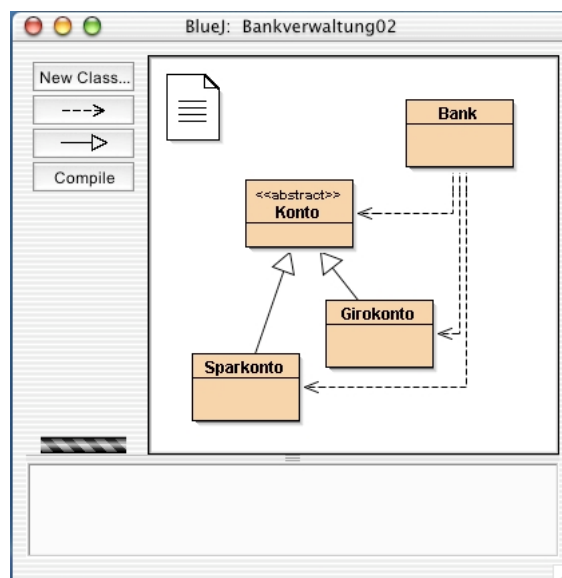


Abbildung 5.9: Klassendiagramm des Projekts *Bankverwaltung02*

## 5.4 Die Klasse *Ansicht*

Die Klasse *Ansicht* kannst du aus dem Projekt *Telefonbuch* übernehmen. Bei meinem Vorschlag (vgl. Abbildung 5.1) musst du eine weitere Registerkarte *geld* eingefügen. Die Schnittstelle zwischen der Klasse *Ansicht* und den Klassen *Bank*, *Konto*, *Sparkonto* und *Girokonto* wurden in den Abbildungen 5.2 und 5.3 dargestellt.

Abbildung 5.4 zeigt das Klassendiagramm des Projekts *Bankverwaltung03*.

Dieser Entwurf der Bankverwaltung zeigt noch viele Mängel in der Benutzerführung. Es verlangt vom Benutzer eine bestimmte Reihenfolge in der Eingabe von Daten. Die folgende Liste lässt sich beliebig verlängern. Hierzu liefert jedoch das Buch *The JFC Swing Tutorial, A Guide to Constructing GUIs* wertvolle Hinweise zur Programmierung einer sinnvollen Swing-Oberfläche.

1. Es lässt sich keine neue Textdatei *Bank* anlegen. Man kann nur eine bereits vorhandene wählen.
2. Wird auf der Registerkarte *geld* ein Betrag eingezahlt, stiftet das Textfeld *auf Kontonummer* Verwirrung, da es nicht benötigt wird.
3. Nach dem Einzahlvorgang sollte der Betrag aus dem entsprechenden Textfeld wieder entfernt werden.
4. Beim Überweisungsvorgang muss das Quellkonto im *personP* dargestellt werden.
5. Die Nummer des Empfängerkontos muss man aus einem eventuell vorher gemachten Ausdruck entnehmen.
6. Für die Überweisung wäre ein Bild aller vorhandenen Kontoeinträge nützlich.
7. ....