

Entwurfsmodell für eine einfache Bankanwendung

Das folgende Klassendiagramm modelliert einen Entwurf für eine einfache Bankanwendung. Jedes Girokonto kostet eine monatliche Gebühr und jedes Sparkonto wird mit einem monatlichen Zinssatz verzinst. Mit der Operation *monatsabschluss()* führt die Bank bei jedem ihrer Konten eine Monatsabrechnung durch. Bei einem Girokonto soll mit der Operation *monatsabrechnung()* die monatliche Gebühr abgezogen werden, bei einem Sparkonto sollen mit der Operation *monatsabrechnung()* die monatlich anfallenden Zinsen gutgeschrieben werden.

Die Aufgaben 1 - 3 beziehen sich auf dieses Entwurfsmodell. Die Aufgaben können jedoch unabhängig voneinander bearbeitet werden. Lösungen zu Aufgabe 1, Teil b) und c) sind in Abb. 1 einzutragen. Alle anderen Lösungen sind auf Extrablätter zu schreiben.

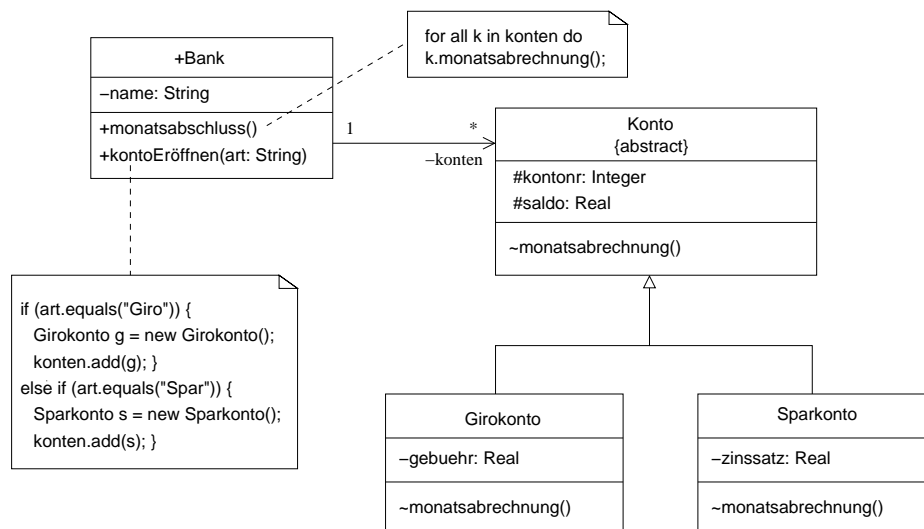


Abbildung 1: Entwurfsmodell für eine Bankanwendung

Aufgabe 1

(12 Punkte)

Die folgenden Teilaufgaben beziehen sich auf das Entwurfsmodell für die Bankanwendung in Abb. 1. Die Teilaufgaben können unabhängig voneinander bearbeitet werden.

- (a) Geben Sie ein bzgl. des Klassendiagramms zulässiges Objektdiagramm (Instanzendiagramm) an, das folgenden Systemzustand beschreibt: Eine Bank mit Namen *NOGO* verwaltet ein Girokonto mit der Kontonummer 111 und ein Sparkonto mit der Nummer 222. Das Girokonto kostet monatlich 4,50 Euro und ist aktuell mit -250,00 Euro belastet. Das Sparkonto wird jährlich mit 3,25% verzinst und besitzt aktuell ein Guthaben von 5100,00 Euro.
(3 Punkte)
- (b) Überprüfen Sie das gegebene Klassendiagramm nach Abhängigkeiten und zeichnen Sie ggf. Abhängigkeitsbeziehungen zwischen Klassen direkt in Abb. 1 ein (mit kurzer Begründung).
(2 Punkte)
- (c) Überprüfen Sie, welche der angegebenen Operationen abstrakt sind. Abstrakte Operationen sind in Abb. 1 entsprechend zu kennzeichnen! Für nicht abstrakte Operationen sind, soweit noch nicht vorhanden, Notizen mit dem jeweiligen Code der Operation anzubringen.
(3 Punkte)
- (d) Erklären Sie anhand des gegebenen Entwurfsmodells der Bankanwendung
 - das Substitutionsprinzip des objektorientierten Ansatzes und
 - das Prinzip der dynamischen Bindung.
(4 Punkte)

Aufgabe 2

(6 Punkte)

Das Klassendiagramm für die Bankanwendung in Abb. 1 befinde sich in einem Paket namens *bank*. Übersetzen Sie das Klassendiagramm in ein Java Programm! Methodenrumpfe und import-Anweisungen können weggelassen werden.

Aufgabe 3

(12 Punkte)

Die folgenden Teilaufgaben beziehen sich auf das Entwurfsmodell für die Bankanwendung in Abb. 1. Die Teilaufgaben können unabhängig voneinander bearbeitet werden.

Es soll eine 2-Schichten-Architektur für die Bankanwendung entwickelt werden, wie sie in Abb. 4 dargestellt ist. Das Paket *bankgui* soll die Benutzerschnittstelle enthalten. Das Paket *bank* enthalte das Klassendiagramm von Abb. 1, das den Anwendungskern bildet.



Abbildung 2: 2-Schichten-Architektur für die Bankanwendung

Die Benutzerschnittstelle soll über zwei Knöpfe verfügen. Mit einem Knopf soll die Ausführung eines Monatsabschlusses bei der Bank ausgelöst werden; mit dem anderen Knopf soll die Anwendung beendet werden.

- (a) Geben Sie den Inhalt des Pakets *bankgui* in Form eines Klassendiagramms an, so dass die beschriebene Funktionalität erfüllt wird. Dazu sollen die folgenden Klassen und Interfaces sowie notwendige aus dem Anwendungskern importierte Klassen verwendet werden.

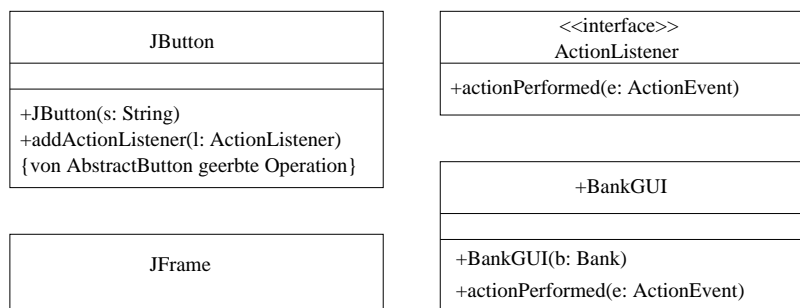


Abbildung 3: Klassen und Interfaces für die Benutzerschnittstelle

Für den Konstruktor der Klasse *BankGUI* braucht der Code zum strukturellen Aufbau der GUI nicht angegeben zu werden. Formulieren Sie jedoch in einer dem Konstruktor zugeordneten Notiz, welche Vorbereitungen für ein wirksames Event-Handling im Rumpf des Konstruktors nötig sind. In dieser Notiz soll auch angegeben werden, wie die Verbindung zum Anwendungskern hergestellt wird, wenn beim Aufruf des Konstruktors der formale Parameter *b* mit einem aktuellen Bankobjekt belegt ist.

Für die Implementierung der Operation *actionPerformed* ist der Code in einer der Operation zugeordneten Notiz anzugeben. Der Zugriff auf die auslösende Quelle eines Events *e* kann mit dem Methodenaufruf *e.getSource()* erfolgen. (8 Punkte)

- (b) Erweitern Sie die in Abb. 4 angegebene Architektur um eine Klasse *BankSimulation* mit der Methode *main(args: String[])* und geben Sie den Code der *main* Methode an, so dass nach deren Ausführung ein *BankGUI*-Objekt mit einem Bankobjekt verbunden ist und die Benutzerschnittstelle sichtbar ist. Welche Abhängigkeiten sind dadurch für die Klasse *BankSimulation* entstanden? (4 Punkte)

Aufgabe 4

(10 Punkte)

Am Eingang eines Schwimmbads befindet sich ein Drehkreuz, das mit einer Lichtanzeige verbunden ist. Das Drehkreuz und die Lichtanzeige werden durch folgendes Klassendiagramm modelliert.



Abbildung 4: Klassendiagramm für Drehkreuz mit Lichtanzeige

Durch das Drehkreuz können Badegäste, die im Besitz einer gültigen Eintrittskarte sind, in das Schwimmbad eintreten. Bevor das Drehkreuz passiert werden kann, muss die Eintrittskarte in einen Schlitz an der Seite des Drehkreuzes eingegeben werden. Es gibt einen Anwendungsfall *Am Drehkreuz eintreten*, der durch folgendes Primär- und Sekundärszenario beschrieben wird.

Primärszenario (Drehkreuz passieren):

- Der Badegast gibt seine Eintrittskarte ein.
- Das Drehkreuz überprüft die Eintrittskarte, veranlasst, dass an der Lichtanzeige das grüne Licht eingeschaltet wird und gibt die Eintrittskarte wieder aus.
- Der Badegast entnimmt die Karte und das Drehkreuz entsperrt sich.
- Der Badegast passiert die Schranke, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das grüne Licht wieder ausgeschaltet wird; anschließend sperrt sich das Drehkreuz wieder.

Sekundärszenario (Karte ungültig):

- Der Badegast gibt seine Eintrittskarte ein.
- Das Drehkreuz überprüft die Eintrittskarte, veranlasst, dass an der Lichtanzeige das rote Licht eingeschaltet wird und gibt die Eintrittskarte wieder aus.
- Der Badegast entnimmt die Karte, woraufhin das Drehkreuz veranlasst, dass an der Lichtanzeige das rote Licht wieder ausgeschaltet wird.

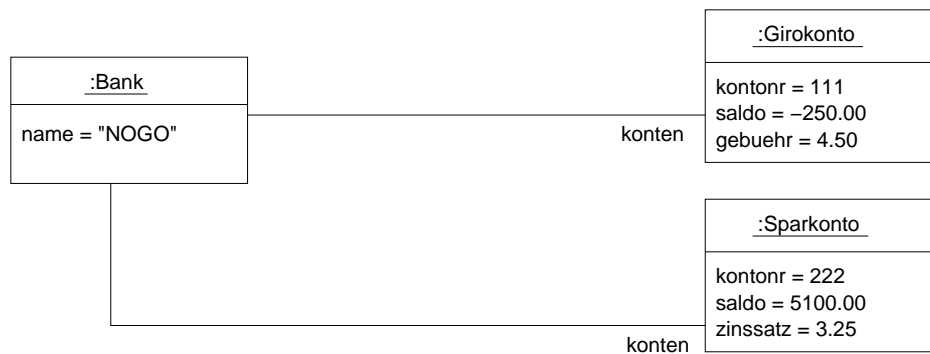
Geben Sie zwei Sequenzdiagramme an, die die Interaktionen zwischen einem Badegast, dem Drehkreuz und der Lichtanzeige

(a) gemäß des oben beschriebenen Primärszenarios und (6 Punkte)

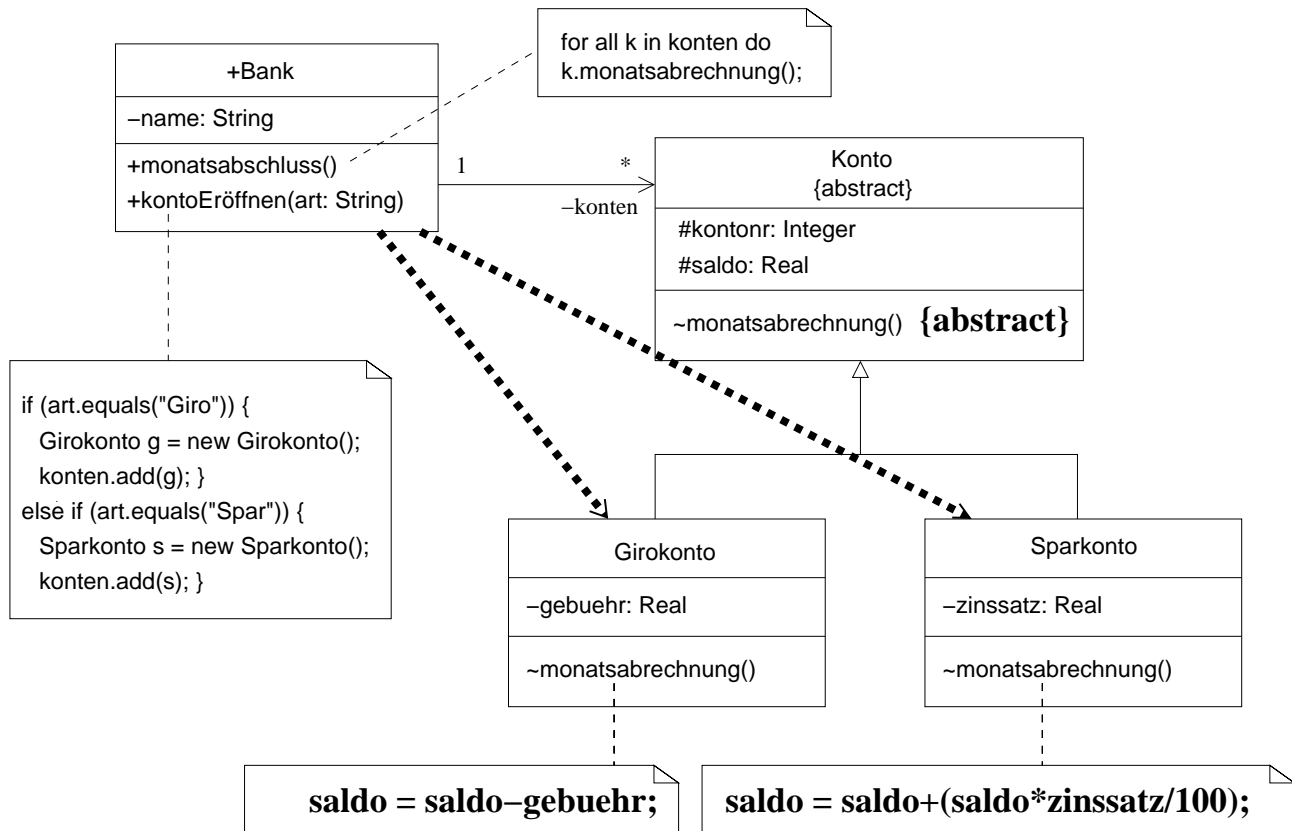
(b) gemäß des oben beschriebenen Sekundärszenarios zeigen. (4 Punkte)

Die Sequenzdiagramme sollen genau darstellen, wann das Drehkreuz und die Lichtanzeige sich in einer Aktivierungsphase befinden. Achten Sie auch darauf, die Art der Nachrichten (synchron oder asynchron) gemäß der UML Syntax richtig darzustellen.

Lösung zu Aufgabe 1 a):



Lösungen zu Aufgaben 1 b) und 1 c):



Lösung zu Aufgabe 1 d):

- Substitutionsprinzip: Die Klassen *Girokonto* und *Sparkonto* sind Subklassen der Klasse *Konto*. Deshalb kann an jeder Stelle, wo ein Konto erwartet wird ein Girokonto oder ein Sparkonto eingesetzt werden (z.B. bei den *konten* einer Bank).
- Dynamische Bindung: Im Code der Operation *monatsabschluss* wird für alle Konto-Objekte (die unter *konten* zu finden sind) die Operation *monatsabrechnung* aufgerufen. Zur Laufzeit wird festgestellt, welchen aktuellen Typ das gerade betrachtete Konto hat (Girokonto oder Sparkonto). Der in der entsprechenden Klasse implementierte Code der Operation *monatsabrechnung* wird dann ausgeführt.

Lösung zu Aufgabe 2:

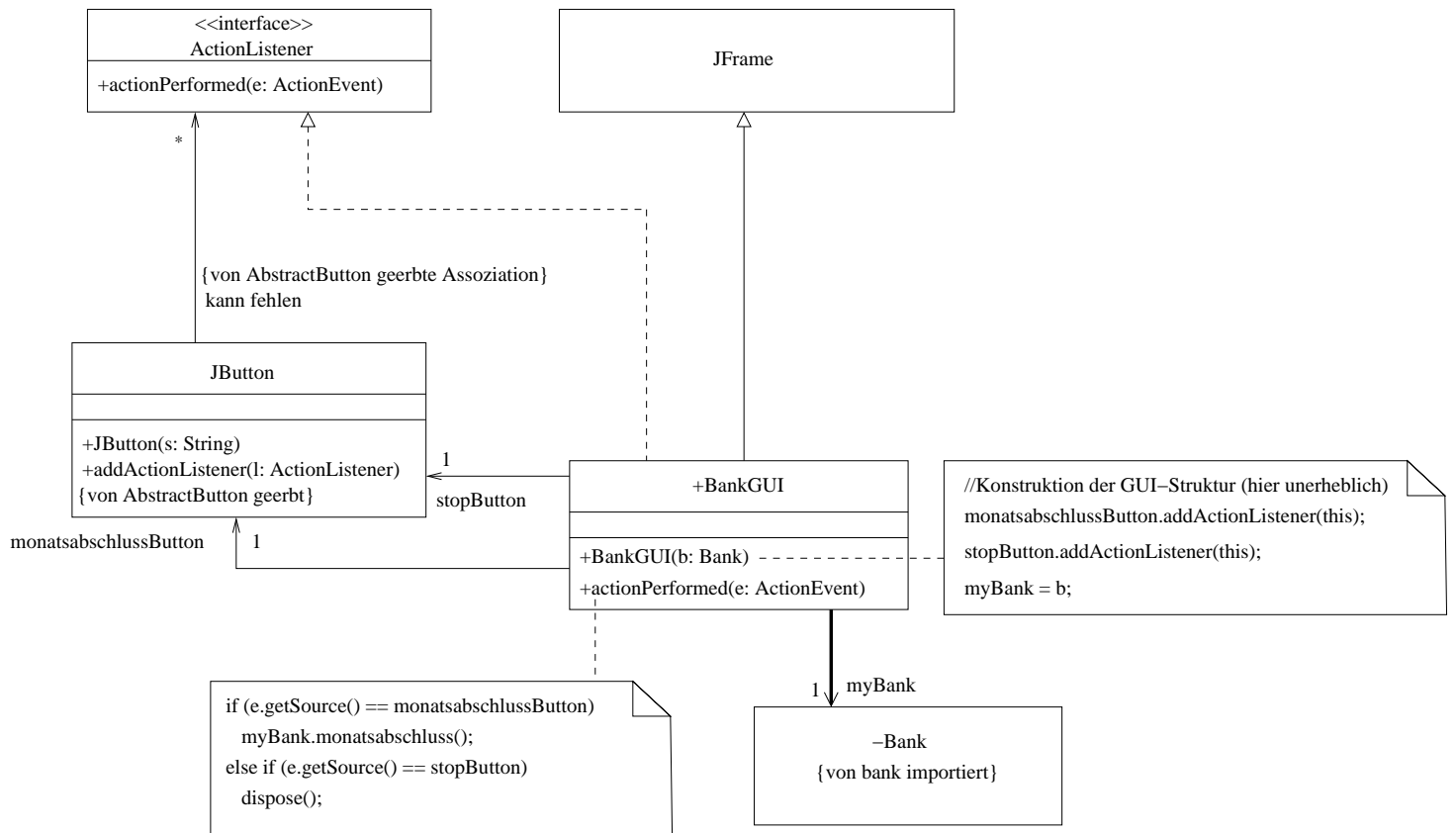
```
package bank;
import java.util.*;  \\kein Abzug, wenn import fehlt
public class Bank {
    private String name;
    private Set<Konto> konten = new HashSet<Konto>();
    \\Set ohne <Konto> auch ok, List auch ok
    public void monatsabschluss() {
        ...
    }
    public void kontoEroeffnen(String art) {
        ...
    }
}

abstract class Konto {
    protected int kontonr;
    protected double saldo;  //float auch ok
    abstract void monatsabrechnung();
    \\ "abstract" kann fehlen, da nicht im Klassendiagramm vorgegeben
}

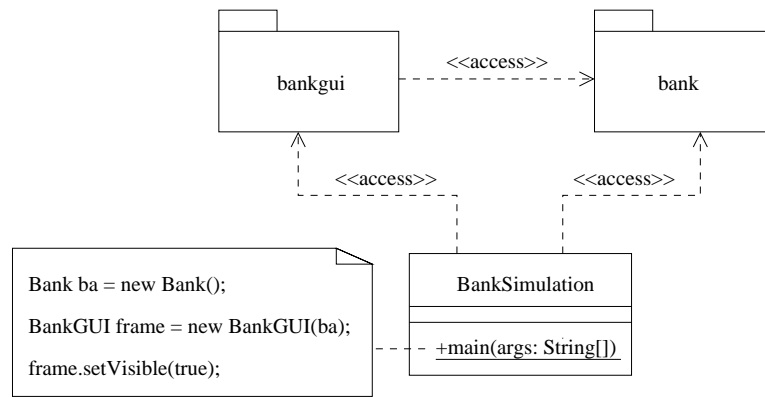
class Girokonto extends Konto {
    private double gebuehr;
    void monatsabrechnung() {
        ...
    }
}

class Sparkonto extends Konto {
    private double zinssatz;
    void monatsabrechnung() {
        ...
    }
}
```

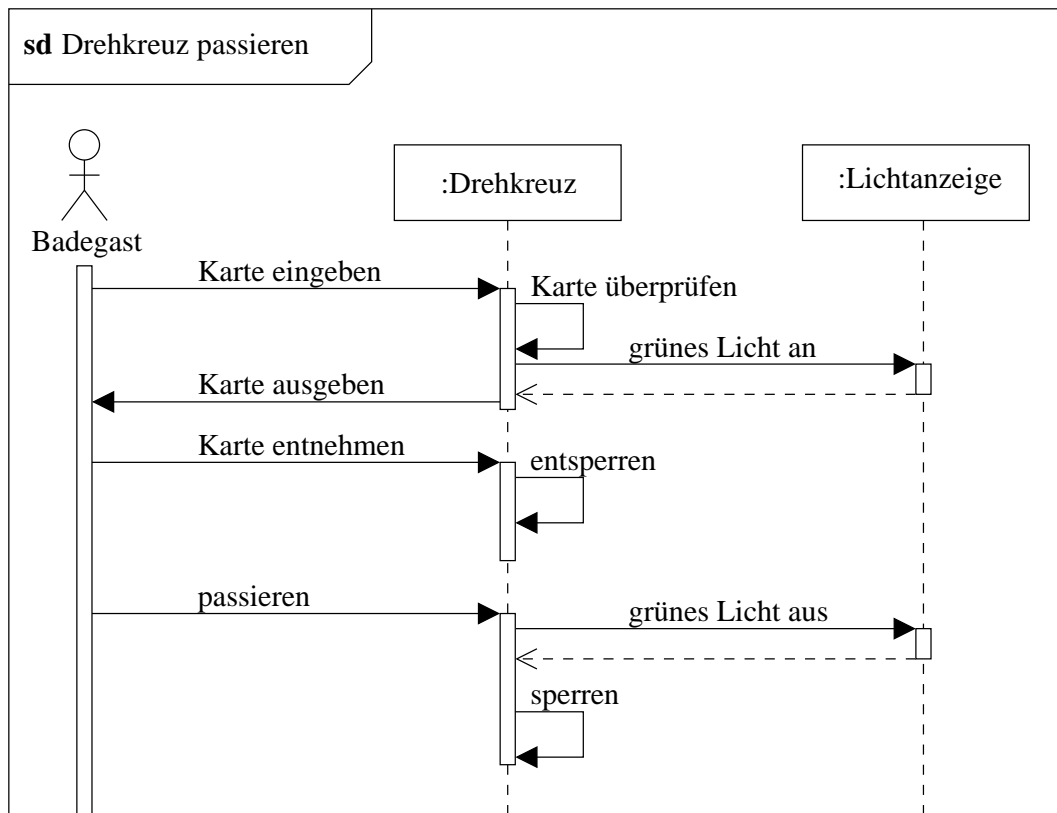
Lösung zu Aufgabe 3 a):



Lösung zu Aufgabe 3 b):



Lösung zu Aufgabe 4 a):



Lösung zu Aufgabe 4 b):

