Refactoring AccountApplication

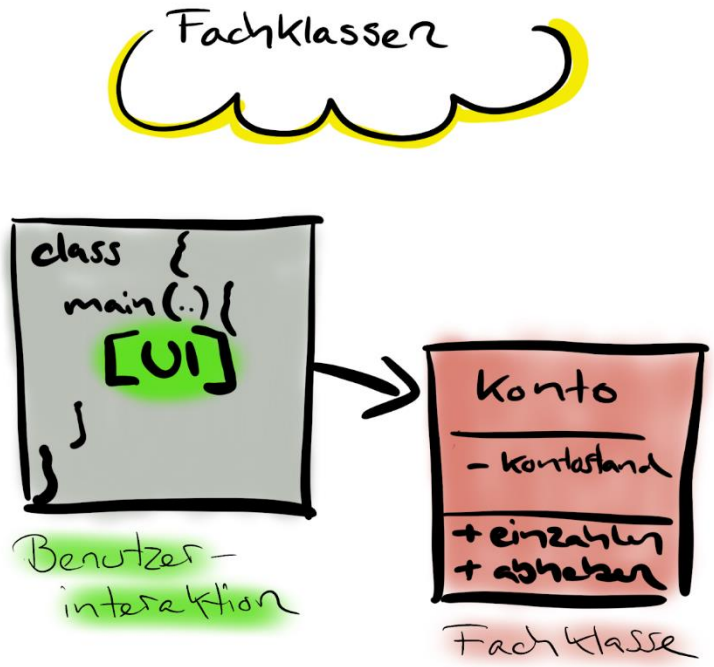
☝ **Machen Sie sich mit dem Konzept der Fachklassen bekannt, bevor Sie weiterfahren!**

Ausgangslage

Wir haben gesehen, dass unsere Klasse AccountApplication eigentlich zu viel macht. Sie kümmert sich um die Benutzerinteraktionen, aber auch um die Verwaltung des Kontostandes. Das sind zwei Aufgabenbereiche.

In der Programmierung wird angestrebt, dass eine Klasse nur **einen** Aufgabenbereich übernimmt (Single-Responsibility-Prinzip, kurz SRP). Somit bietet sich in unserem Beispiel die Trennung von *Benutzerinteraktion* und der *Verwaltung des Kontostandes* an.

- AccountApplication (Beinhaltet die Benutzerinteraktion und die main-Methode)
- Account (Beinhaltet die Fachlogik)



Einführung der Klasse Account

Die **Fachlogik** der `AccountApplication` kann in eine eigene Klasse `Account` ausgelagert werden.

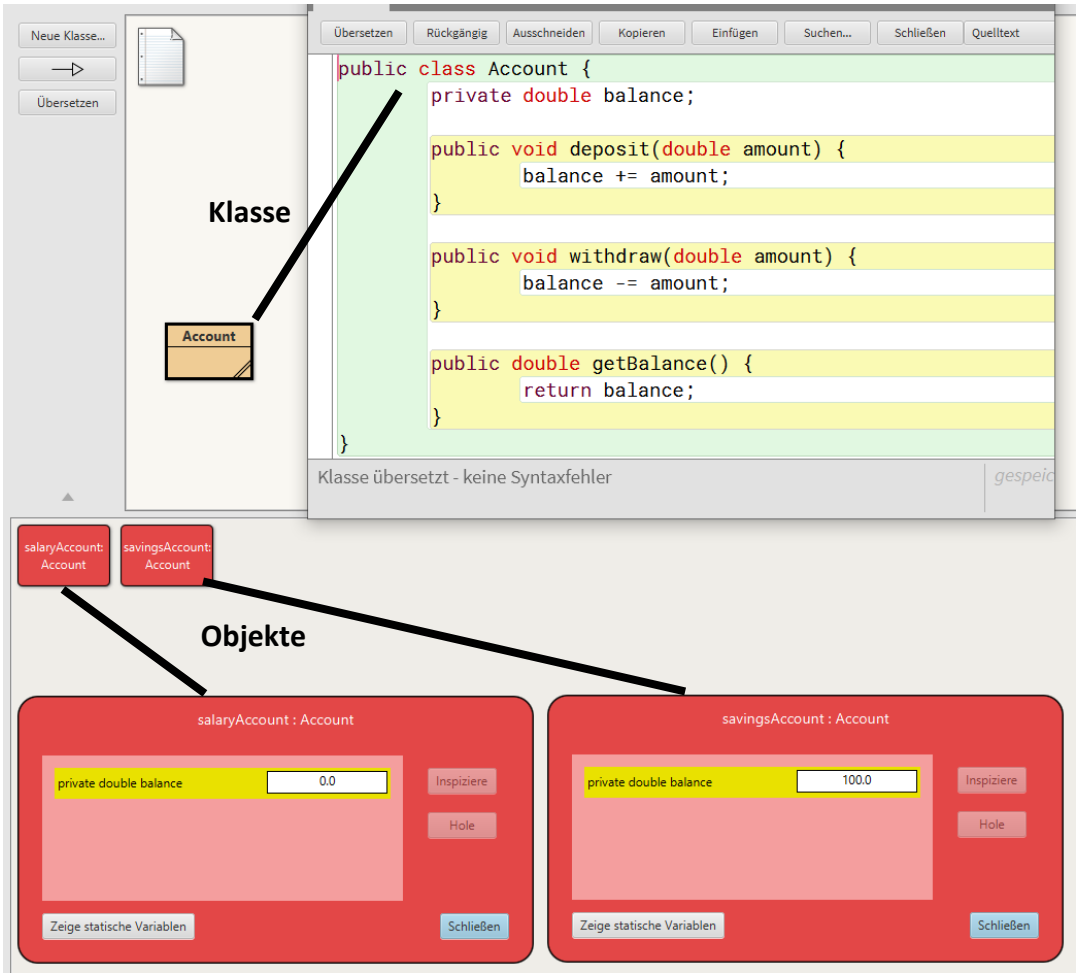
```
public class Account {
	private double balance;

	public void deposit(double amount) {
		balance += amount;
	}

	public void withdraw(double amount) {
		balance -= amount;
	}

	public double getBalance() {
		return balance;
	}
}
```

Wie wir gesehen haben, hat dies nun auch den Vorteil, dass wir daraus viele Kontoobjekte erstellen können, die komplett eigenständig einen Kontostand verwalten können. Somit wird ermöglicht, theoretisch mehrere Konti anzulegen.



Beispiel: Mehrere Objekte der Klasse Account

```
// neues `Account` Objekt gespeichert in der Variable `sparkonto`
Account sparkonto = new Account();
// neues `Account` Objekt gespeichert in der Variable `lohnkonto`
Account lohnkonto = new Account();

sparkonto.deposit(10); // dem Sparkonto 10 Franken einzahlen
sparkonto.deposit(20); // dem Sparkonto 20 Franken einzahlen

lohnkonto.withdraw(20); // dem Lohnkonto 20 Franken abheben

System.out.println(sparkonto.getBalance()); // => 30;
System.out.println(lohnkonto.getBalance()); // => -20;
```

Aufgabe

Bauen Sie Ihr Programm nun so um, dass es aus zwei Klassen besteht (die ursprüngliche Klasse und die Klasse Account).

- Erstellen und implementieren Sie die Klasse `Account`.
- Löschen Sie in der ursprünglichen Klasse die Variable `double balance;`
- Legen Sie dafür ein Objekt der Klasse `Account` an.
- Jetzt erscheinen Fehler im Quellcode. Überall dort müssen Sie das Programm anpassen und mit dem neuen Objekt der Klasse `Account` arbeiten.
- Die Methoden `deposit` und `withdraw` in der Hauptklasse können entfernt werden. Arbeiten Sie direkt mit den Methoden des `Account`-Objekts.