

Übung 02: Vererbung, Exceptions

Abgabetermin: 19. 3. 2009, 8:15

Name: _____

Matrikelnummer: _____

Informatik: G1 (Prähofer) G2 (Prähofer) G3 (Würthinger) G4 (Prähofer)

WIN: G1 (Khalil) G2 (Khalil) G3 (Schwinger)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 1	24	Prosabeschreibung, Java-Programm, Ausgabe eines Testlaufs	Java-Programm	<input type="checkbox"/>	

Übung 02: Bankkonten

(24 Punkte)

In dieser Übungsaufgabe sollen Sie unterschiedliche Bankkonten als Klassen realisieren. Implementieren Sie eine Basisklasse `Account` und Spezialisierungen für Girokonten (Klasse `TransferAccount`), Sparkonten (Klasse `SavingsAccount`) und Kreditkonten (Klasse `CreditAccount`).

Basisklasse Account:

Jedes Konto hat eine eindeutige Kontonummer und einen aktuellen Kontostand. Deklarieren Sie entsprechende Felder und Zugriffsmethoden. Die Kontonummer muss eindeutig sein und sollte daher vom Programm generiert und unveränderlich sein. Überlegen Sie die Sichtbarkeitsattribute für die Felder und ob auch eine `set`-Methode definiert werden soll (d.h. ob das Feld von außen gesetzt werden kann).

Hinweise:

- Zur Vereinfachung können Sie für die Geldbeträge den Typ `int` verwenden.
- Verwenden Sie ein `static int`-Feld `nextId`, um die Kontonummern zu generieren.

Man kann von einem Konto Geld abheben und auf ein Konto Geld einzahlen. Definieren Sie dazu entsprechende Methoden. Es soll möglich sein, die letzte Behebung oder Einzahlung rückgängig zu machen. Realisieren Sie dazu eine Methode `void undoLastOperation()`.

Definieren Sie weiters eine Methode `public String toString()`, um eine String-Repräsentation des Kontos zu generieren.

Spezialisierungen TransferAccount, SavingsAccount und CreditAccount:

Leiten Sie dann Unterklassen `TransferAccount` für Girokonten, `SavingsAccount` für Sparkonten und `CreditAccount` für Kreditkonten ab. Dabei sind folgende Zusätze und Einschränkungen zu beachten:

Girokonten:

- Haben eine maximale Überziehung. Über diese Grenze darf das Konto nicht belastet werden.

Sparkonten:

- Dürfen nur im Haben sein (Kontostand > 0).

Kreditkonten:

- Kreditkonten werden anfänglich mit einem Soll (Kontostand < 0) angelegt. Man kann darauf nur einzahlen (Kredit zurückzahlen) und nicht weiter beheben.
- Auf das Kreditkonto kann nicht mehr eingezahlt werden, als der ausstehende Kredit ausmacht (Kontostand darf nicht > 0 werden).

Prüfen Sie alle diese Bedingungen in den Methoden. Die Methoden sollen bei Verletzung von Bedingungen `Exceptions` auslösen. Definieren Sie sich dazu (eine oder mehrere) `Exception`-Klassen.

Enumerationen:

Versuchen Sie in der Aufgabe Enumerations zu verwenden. Wo könnten diese sinnvoll sein?

Test:

Realisieren Sie eine Testklasse mit Testmethoden, in der Sie die einzelnen Methoden testen. Die Tests sollen so gestaltet sein, dass

- eine bestimmte Methode eines Testobjekts aufgerufen wird
- dann das erwartete Ergebnis überprüft wird
- und bei Fehlschlägen möglichst genau das beobachtete Fehlverhalten beschrieben wird.

Testen Sie dann auch, ob bei Verletzung der entsprechenden Bedingungen die Exceptions richtig ausgelöst werden. Das heißt:

- eine Methode eines Testobjekts soll so aufgerufen werden, dass eine Exception erwartet wird
- wird die Exception wirklich geworfen, arbeitet die Funktion wie erwartet und das Programm ist korrekt
- wird die Exception aber nicht geworfen, also das Programm nach der Methode normal fortgesetzt, liegt ein Fehler vor

Konstruieren Sie dann folgenden speziellen Testfall:

1. Von einem Sparkonto wird ein Betrag behoben.
2. Es wird versucht, diesen Betrag auf ein Kreditkonto einzuzahlen. Dies scheitert aber, weil der ausstehende Kredit kleiner als der Betrag ist. Es wird eine Exception ausgelöst.
3. Fangen Sie diese Exception und machen Sie die Behebung vom Sparkonto rückgängig (Methode `undoLastOperation`).

Anleitungen

Gehen Sie bei der Aufgabe folgend vor:

- Verfassen Sie zuerst eine kurze Beschreibung der Klassen, der Operationen der Klassen und aller Bedingungen, die gelten müssen.
- Gestalten Sie dann die öffentlichen Schnittstellen der Klassen und beschreiben Sie die Methoden durch Kommentare. Beschreiben Sie insbesondere auch, welche Bedingungen gelten müssen und fügen Sie `throws`-Klauseln an.
- Realisieren Sie die einzelnen Methoden.
- Schreiben Sie die Testklasse mit den Testmethoden. Testen Sie zuerst die normalen Fälle, bei denen keine Exceptions ausgelöst werden. Testen Sie dann, dass bei den entsprechenden Fehlerfällen die Exceptions richtig ausgelöst werden.

Achten Sie bei dieser Aufgabe ganz besonders auf:

- die Gestaltung der Schnittstelle der allgemeinen Basisklasse und der Spezialisierungen,
- das Überschreiben der Methoden in den Spezialisierungen mit Verwendung von `Super`-Aufrufen,
- die Verwendung von Exceptions zur Signalisierung von Verletzungen der Bedingungen bei den unterschiedlichen Konten und
- die Gestaltung der Tests mit der Überprüfung, dass die Exceptions richtig ausgelöst werden.