

Übung 02: Inheritance, Exceptions

Hand in by:

19/MAR 2009, 08:15

Name: _____

Student Nr.: _____

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

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

Assignment	Points	to hand in written form	to hand in electronically	corr.	Points
Übung 2	24	Prosabeschreibung, Java-Programm, Ausgabe eines Testlaufs	Java-Programm	<input type="checkbox"/>	

Übung 02: Bank Account

(24 Punkte)

In this exercise, you have to implement different bank accounts in terms of classes. Implement a base class `Account` and sub-classes for checking accounts (class `TransferAccount`), savings accounts (class `SavingsAccount`) and credit accounts (class `CreditAccount`).

Base-Class Account:

Each account has a unique account number and a current account balance. Define the appropriate attributes and access methods for this. The account number must be unique and should therefore be generated by the program and should be immutable. Furthermore, define the appropriate visibility of attributes, and whether a set-method is to be defined (i.e., whether the attribute(s) shall be modifiable in that way through other classes).

Notes:

- For the sake of simplicity, you may use the type `int` for representing the amount of money in the accounts.
- Use a static `int`-Feld `nextId` to generate the account numbers.

One can withdraw from and deposit money onto accounts - define appropriate methods for this.

It should be possible to reverse the last withdraw of deposit with a dedicated method `void undoLastOperation()`.

Define a method `public String toString()` to generate a string representation of the account.

Specialisations for TransferAccount, SavingsAccount and CreditAccount:

Based on the `Account` class subclasses `TransferAccount` for current accounts, `SavingsAccount` for savings accounts and `CreditAccount` for credit accounts shall be realized inheriting from the base class. Notice the following additions and limitations for the respective account types.

Checking Account:

- Have a maximum overdraft. Over this limit, the account will not be charged.

Savings Account:

- Have to be positive (balance > 0).

Credit Account:

- Credit accounts are initially set with a negative amount (balance < 0). You can only add to it (to pay back credit) and may not make any further withdrawal.
- Not more than the initial credit may be paid onto the account (account must not be > 0).

Check all of these conditions in the respective methods. The methods shall trigger exceptions if these conditions stated are violated. For this, appropriate classes indicating these violations should be defined.

Enumerations:

Try to use enumeration in this assignment. Where could this be useful?

Testing:

Implement a test class with test methods in which the individual methods are tested. The tests should be designed so that

- a specific method of a test object is invoked
- then the expected result will be verified
- and in case of a failure, the reasons for that failure should be reported as accurately as possible.

Also test whether the relevant exceptions are triggered correctly in case of violation of account constraints. This means:

- a method of a test object should be invoked expecting a certain Exception
- if the exception is thrown as expected, the respective method works correctly
- if the exception is not thrown as expected and thus the program continues normally, the according method is not working correctly and the test should report an error.

Construct the following specific test case:

1. A certain amount is withdrawn from a savings account
2. It is tried to deposit this amount onto a credit account. This fails, however, because the outstanding credit is less than the amount intended to deposit. There an exception is thrown.
3. Catch this exception and reverse the withdrawal from the savings account (Method `undoLastOperation`).

Guidance:

Proceed in the following for solving this assignment:

- Firstly, write a brief description of the classes, the methods of these classes and all conditions, which must be considered.
- Design the public interfaces of the classes and describe the methods by comments. Describe in particular, what conditions must be fulfilled and add the respective throws clauses to the class definition.
- Realize the individual methods.
- Write a test class containing the respective test methods. First, focus on the normal test cases where no exceptions are triggered. Then, focus on the test cases where exceptions are expected to be triggered.

Particularly, pay attention to:

- the design of the interface of the general base class and the specializations,
- overriding methods in the specializations and usage of super-calls,
- the use of exceptions to signal violations of conditions in the various account types and
- the design of tests with the intentions that the triggering exceptions is correct.