Praktikum 3

Aufgabe 1 (Abstraktion einer Bank):
In dieser Aufgabe soll eine Abstraktion einer Bank hinzukommen, die Konten und Transaktionen verwalten und verarbeiten kann.
Zentraler Bestandteil dieser Abstraktion ist das Interface *Bank*, das in den Praktikumsunterlagen zur Verfügung gestellt wird. Dieses repräsentiert eine Schnittstelle auf eine Menge von Operationen, die in allen möglichen Ausprägungen von Banken zur Verfügung stehen sollten.
Kopieren Sie dieses zunächst in Ihr Projekt (innerhalb des Java-Packages *bank*) und schauen Sie sich den Inhalt der Datei genauer an und lesen Sie insbesondere die *Javadoc*-Kommentare. Relevant sind auch die Exceptions, die als Fehlererkennung und -behandlung angegeben wurden.

Legen Sie nun eine Klasse *PrivateBank* im Java-Package *bank* an, die das Interface *Bank* implementieren und folgende Klassenattribute beinhalten soll:

1. *name* (Typ: String): Dieses Attribut soll den Namen der privaten Bank repräsentieren.
2. *incomingInterest* (Typ: double): Dieses Attribut gibt die Zinsen bzw. den Zinssatz (positiver Wert in Prozent, 0 bis 1) an, die bei einer Einzahlung (Deposit) anfallen. Dieses Attribut soll identisch zu dem gleichnamigen Attribut der Klasse *Payment* sein.
3. *outgoingInterest* (Typ: double): Dieses Attribut gibt die Zinsen bzw. den Zinssatz (positiver Wert in Prozent, 0 bis 1) an, die bei einer Auszahlung (Withdrawal) anfallen. Dieses Attribut soll identisch zu dem gleichnamigen Attribut der Klasse *Payment* sein.
4. *accountsToTransactions* (Typ: *Map*<*String*, *List*<*Transaction*>>): Dieses Attribut soll Konten mit Transaktionen verknüpfen, so dass jedem gespeicherten Konto 0 bis n Transaktionen zugeordnet werden können. Der Schlüssel steht für den Namen des Kontos.
   Beispiel: *„Konto 1“ -> [Transaktion 1, Transaktion2]*
   „Konto Adam“ -> [Transaktion 3]
   „Konto Eva“ -> []
   …

Überlegen Sie generell bei allen Klassenattributen, welche Sichtbarkeit sinnvoll wäre (private, public, …).

Implementieren Sie außerdem Getter- und Settermethoden für die ersten drei Attribute (s. oben).
Im nächsten Schritt soll ein Konstruktor für die ersten drei Attribute implementiert werden. Die Map *accountsToTransactions* soll hingegen nicht im Konstruktor, sondern direkt bei der Definition des Klassenattributs mit *new* erzeugt werden.
Implementieren Sie zusätzlich einen Copy-Konstruktor, der die ersten drei Attribute kopieren soll.
Überschreiben Sie nun die *toString*() und *equals*(*Object*)-Methoden der Klasse *Object*. Für die Überprüfung mit *equals*(*Object*) sollen alle Klassenattribute verwendet und geprüft werden.

Im nächsten Schritt sollen alle Methoden, die durch das Interface *Bank* gefordert werden, in der Klasse *PrivateBank* implementiert werden. Achten Sie insbesondere auch auf die Exceptions, die bei einigen Methoden angegeben wurden und implementieren Sie diese. Alle Exceptions sollen im Java-Package *bank.exceptions* abgelegt werden.

Hinweise zu einigen Methoden (die restlichen Methoden sollten mit Hilfe der Methodensignatur und Javadoc-Dokumentation problemslos implementiert werden können):

1. *addTransaction*(*String*, *Transaction*): Im Falle von *Payments* sollen die Attribute *incomingInterest* und *outgoingInterest* mit den Werten aus der *PrivateBank* überschrieben werden.

2. *getAccountBalance(String):* Die Problemstellung bei der Methode *getAccountBalance*(*String*) bezieht sich nur auf Transfer-Objekte, weil dort zum Zeitpunkt der Berechnung des Kontostandes unklar ist,
   ob der Betrag vom Konto abgezogen („Sendung“) oder hinzuaddiert („Empfang“) werden soll. Um diese Unterscheidung durchführen zu können, sollten die Attribute *sender* und *recipient* der Klasse Transfer verwendet werden.
   Beispiel: *Aktuelles Konto ist „Account Adam“ und enthält eine Transaktion in Form eines Transfers t1. Für dieses Konto soll nun der Kontostand berechnet werden.*
   Fall 1) t1.sender == „Account Adam“ → Betrag muss abgezogen werden (negativer Wert), da „Sendung“
   Fall 2) t1.sender == „Account Eva“ → Betrag muss hinzuaddiert werden (positiver Wert), da „Empfang“

Sie sollen die Logik für *getAccountBalance(String)* in diesem Praktikumsversuch auf zwei unterschiedliche Vorgehensweisen implementieren.
(Wichtig: Für die Folgepraktika soll nur Variante 1 verwendet werden!)

Variante 1 (Unterscheidung mit Hilfe von Vererbung):
Legen Sie zwei neue Klassen *IncomingTransfer* und *OutgoingTransfer* im Java-Package *bank* an, die beide von der Klasse *Transfer* erben sollen. Überlegen Sie, wie bzw. ob Sie die Methode *calculate*() sinnvoll überschreiben können.
Anschließend kann die Berechnung des Kontostandes in der Methode *PrivateBank#getAccountBalance*(*String*) ohne weitere Unterscheidungen implementiert werden.

Variante 2 (Unterscheidung mit Hilfe von instanceof und Type-Casting):
Legen Sie eine Kopie der Klasse *PrivateBank* an und nennen Sie diese Klasse *PrivateBankAlt*. In dieser Klasse soll die Methode *PrivateBankAlt#getAccountBalance*(*String*) mit Hilfe des Java-Operators *instanceof* (und *Type-Casting*) implementiert werden. Sie müssen im Falle von *Transfer*-Objekten überprüfen, ob es sich um eine „Sendung“ oder einen „Empfang“ handelt. Arbeiten Sie sich hierfür selbstständig in *instanceof* und *Type-Casting* ein.

Bei der Überprüfung der Klassenattribute *amount* (*Transfer*-Objekte) und *incomingInterest*/*outgoingInterest* (*Payment*-Objekte) soll im Fehlerfall eine Exception des Types *TransactionAttributeException* geworfen werden.

Aufgabe 2 (Dokumentation mit Hilfe von Javadoc):
Erweitern Sie die Dokumentation aus Praktikum 2 mit korrektem *Javadoc* (gilt für Klassendefinitionen, Klassenattribute und Methoden) und verwenden Sie sinnvolle *Javadoc-Tags*, um eine möglichst aussagekräftige Dokumentation des gesamten Programmcodes zu gewährleisten.

Aufgabe 3 (Testen der Funktionalität):
In dieser Aufgabe soll die in Aufgabe 1 erstellte Funktionalität mit Hilfe einer *main*-Methode getestet werden.
Erstellen Sie hierfür zunächst eine Klasse *Main* im *default package* (oder verwenden Sie die alte Klasse wieder). Die Klasse *Main* soll eine *main*-Methode enthalten und somit als Einstiegspunkt für folgende Tests fungieren:

Erzeugen Sie Objekte der Klassen *PrivateBank* (u.a. mit *incomingInterest* und *outgoingInterest* mit Werten größer 0) und *PrivateBankAlt* und testen Sie, ob die implementierten Methoden des Interfaces *Bank* korrekt funktionieren.
Testen Sie die Fehlerfälle, bei denen *Exceptions* geworfen werden sollen.
Testen Sie zusätzlich mit zwei Objekten der Klasse *PrivateBank*, ob die *PrivateBank#equals* Methode korrekt funktioniert.

Aufgabe 4 (Erweiterung der UML-Klassendiagramme):
Erweitern Sie die bisher erstellten UML-Klassendiagramme um die Elemente, die in diesem Praktikumsversuch dazugekommen sind. Achten Sie insbesondere auch auf mögliche Beziehungen zwischen den Klassen und Interfaces und stellen Sie diese UML-konform dar.