Praktikum 5

In diesem Praktikumsversuch soll eine grafische Oberfläche (*GUI*) mit Hilfe von *JavaFX* für das zuvor entwickelte Bankszenario (P1 bis P4) entwickelt werden.
Im Folgenden werden einige Hinweise angegeben, die bei der Implementierung zu berücksichtigen sind.
In den Aufgaben wird im Wesentlichen nur beschrieben, welche Daten angezeigt und wie sich die Fenster bzw. die *GUI* bei Eingaben/Events verhalten sollen. Die konkrete Implementierung und die Anordnung der *UI*-Elemente wird, falls nicht explizit angegeben, Ihnen überlassen.

Allgemeine Hinweise:

- Im Ilias wurden einige Lehrvideos/Clips und Links zur Recherche angelegt. Nutzen Sie diese und weitere Links bzw. Quellen, die Sie selber recherchieren, um mögliche Probleme und Fragestellungen klären zu können.
- Im Ilias liegt wieder eine *pom.xml* mit einer zusätzlichen Abhängigkeit zu *JavaFX*, die verwendet werden kann. Bei der Lösung kann es erforderlich sein, dass die Anwendung auch via Maven gestartet werden muss (s. *mvn javafx:run*). Diese *pom.xml* gibt als Einstiegspunt die Klasse *ui.FxApplication* an; diese müssen Sie ggf. anpassen.
  Andere Möglichkeiten für eine *JavaFX-Installation* sind ebenfalls erlaubt.
- Für die Deklaration der *JavaFX-Scenes* müssen *.fxml*-Dateien verwendet werden (→ pro *JavaFX-Scene* ist eine *.fxml*-Datei erforderlich). Dies bedeutet, dass möglichst alle *GUI*-Komponenten deklarativ in diesen Dateien definiert werden sollten.
  (*Custom*) *JavaFX-Dialogs* dürfen ohne *.fxml*-Dateien erzeugt werden.
- Alle Fehler, die vom Benutzer verursacht werden, sollen mit Hilfe von *JavaFX-Alerts* angezeigt werden. Mind. erforderlich sind die Fälle, die zuvor als Exceptions definiert wurden (evtl. sind nicht alle Exceptions relevant).
- Pro *JavaFX-Scene* sollte ein *JavaFX-Controller* angelegt werden. Beachten Sie, dass gemeinsame Funktionalität von *Controllern* in eine Oberklasse ausgelagert werden sollte.
- In *JavaFX* können Daten zwischen *JavaFX-Scenes* ausgetauscht werden (z.B. beim Wechsel von zwei *JavaFX-Scenes*).
- Es ist häufig sinnvoll, dass Daten bzw. *Properties* direkt an eine *View*-Komponente gebunden werden (*Property Binding*), so dass bei Aktualisierungen von Werten diese direkt in der *View* angezeigt werden („ohne manuelles Setzen der aktualisierten Werte in der View“). Bei *ListViews* ist insbesondere auch die Klasse *ObservableList<T>* relevant (s. auch die Methode ListView#setItems(*ObservableList<T>*)).
  In diesem Praktikumsversuch ist *Property Binding* nicht verpflichtend.
- Für die Verwendung in diesem Praktikumsversuch dürfen zwar mehrere Instanzen von *PrivateBank* erzeugt werden, es muss allerdings immer das gleiche Verzeichnis (*directoryName*) verwendet werden, so dass lediglich eine einzige Bank simuliert wird.

Aufgabe 1 (Erweiterung der Bank):
Für diesen Praktikumsversuch werden zwei weitere Methoden benötigt, die im Interface Bank zu deklarieren und in der Klasse PrivateBank zu implementieren sind:

1. *void deleteAccount(String account) throws AccountDoesNotExistException, IOException*
2. *List<String> getAllAccounts()*

Die Semantik der beiden Methoden sollte anhand der angegeben Methodensignaturen ersichtlich sein.

Aufgabe 2 (Realisierung der grafischen Oberfläche):
Für diese Praktikumsaufgabe sollen zwei *JavaFX-Scenes* in *FXML* (inkl. dazugehörige *JavaFX-Controller*) definiert werden. Diese werden im Folgenden genauer erläutert:

Mainview (soll auch beim Start der Applikation angezeigt werden):
In dieser *View* soll eine Liste aller verfügbaren Accounts der *PrivateBank* angezeigt werden. Als *JavaFX*-Komponente soll hierfür eine *ListView* verwendet werden.
In dieser *ListView* soll ein Kontextmenü („Rechtsklick auf ein Element in der Liste“) mit zwei Einträgen in der dazugehörigen definiert werden (dieses kann auch in der *.fxml*-Datei definiert werden):

1. *Auswählen*: In diesem Fall soll die *JavaFX-Scene „Accountview“* (s. unten) geladen werden, um in eine Detail-Ansicht eines Account wechseln zu können. Hierbei soll kein neues Fenster bzw. *Stage* angelegt werden.
   Hierbei wird es erforderlich sein, dass der zuvor ausgewählte Account der neuen *JavaFX-Scene* bekannt gemacht werden muss (z.B. via Datenaustausch zwischen *JavaFX-Scenes*).
2. *Löschen*: Fragen Sie zunächst mit einem *JavaFX*-Dialog ab, ob der Account wirklich gelöscht werden soll (Ja/nein). Im Falle einer Bestätigung soll der Account sowohl aus der *ListView* (z.B. via *Property Binding* oder durch manuelle Aktualisierung), als auch aus der *PrivateBank* entfernt und somit von der Festplatte gelöscht werden.

Zusätzlich soll eine *JavaFX*-Komponente (z.B. via *Button*, *MenuBar*, …) in die *View* integriert werden, die das Erzeugen von neuen Accounts ermöglichen soll. Beim Hinzufügen eines neuen Accounts soll ein modaler Dialog erscheinen, der den Accountnamen abfragt. Anschließend soll dieser Account in der *PrivateBank* hinzugefügt und in der *ListView* angezeigt werden.

Accountview:
In dieser *View* soll unter anderem der Name des aktuell ausgewählten Accounts und der aktuelle Kontostand (*account balance*) angezeigt werden. Bei Veränderungen von Transaktionen, die weiter unten beschrieben werden, muss immer der korrekte Wert des Kontostandes berechnet und angezeigt werden.
Desweiteren soll ein „*Back-Button*“ hinzugefügt werden, mit dem die *JavaFX-Scene* gewechselt werden kann. Hierbei soll wieder zur „*Mainview*“ gewechselt werden (im gleichen Fenster bzw. Stage).

Die primäre *View* dieser *JavaFX-Scene* soll eine Liste aller Transaktionen sein, die mit diesem Account verknüpft sind. Als *JavaFX*-Komponente soll hierfür wieder eine *ListView* verwendet werden.
Für die Anzeige der Transaktionen in der Liste kann die *toString*()-Repräsentation verwendet werden.
Die Anordnung der Listenelemente (hier: Transaktionen) soll nach vier Optionen einstellbar/veränderbar sein (z.B. via *Buttons*, *MenuBar*, *ContextMenu*, …):

1. *Aufsteigende Sortierung* (→ *getTransactionsSorted*)
2. *Absteigende Sortierung* (→ *getTransactionsSorted*)
3. *Anzeige von nur positiven amounts (→ getTransactionsByType)*
4. *Anzeige von nur negativen amounts (→ getTransactionsByType)*

In dieser *ListView* soll zusätzlich ein Kontextmenü („Rechtsklick auf ein Element in der Liste“) mit Einträgen für eine „Löschen-Operation“ angeboten werden:
Fragen Sie zunächst mit einem *JavaFX*-Dialog ab, ob die Transaktion wirklich gelöscht werden soll (Ja/nein). Im Falle einer Bestätigung soll die Transaktion sowohl aus der *ListView* (z.B. via *Property Binding* oder manueller Aktualisierung), als auch aus der *PrivateBank* entfernt und somit von der Festplatte gelöscht werden. Relevant ist hier auch die Anzeige des (aktualisierten) Kontostandes (s. oben).

Zusätzlich soll eine *JavaFX*-Komponente (z.B. via *Button*, *MenuBar*, …) in die *View* integriert werden, die das Erzeugen von neuen Transaktionen ermöglichen soll. In diesem Fall sollen ein *JavaFX-Dialog* aufgerufen werden, über den der Benutzer entweder ein *Payment* oder ein *Transfer* anlegen kann. Die Unterscheidung, ob es sich im Falle eines *Transfers* um *Incoming*- oder *OutgoingTransfer* handelt, sollte möglichst programmatisch entschieden und somit nicht vom Benutzer angegeben werden. Hierfür kann es erforderlich sein, dass Sie einen eigenen *Custom Dialog* erstellen müssen.
Alle Attribute müssen vom Benutzer angegeben werden, ansonsten soll ein Fehler angezeigt und die Eingabe verworfen werden. Relevant ist hier auch die Anzeige des (aktualisierten) Kontostandes (s. oben).