Contact-Manager Extended

Themen:

- Enumeration und ComboBox
- TreeView

Aufgabenstellung

Der Contact-Manager soll um folgende Funktionalität erweitert werden:

- Jeder Kontakt soll einem Typ (Business, Private, None) zugeordnet werden.
  - Tipp: Der Kontakttyp soll im Model der Applikation mit einer Enumeration realisiert werden. In der Datenbank soll ein entsprechendes Attribut mit dem Typ `VARCHAR(20)` hinzugefügt werden.
- Im GUI ist für die Auswahl des Kontakttyps eine ComboBox zu verwenden.
- Die flache Kontaktliste ist durch ein TreeView-Steuerelement zu ersetzen, welches die Kontakte nach Typ gruppiert.

Vorgehensweise

TASK 1: Enumeration ContactType

Fügen Sie im `model`-Package einen Enumerationtyp `ContactType` hinzu. Erstellen Sie die Ausprägungen BUSINESS, PRIVATE, NONE

TASK 2: Contact-Klasse

Erweitern Sie die `Contact`-Klasse um ein Feld zur Zuordnung des Kontakts zu einem Typ:

```
private ObjectProperty<ContactType> type = new SimpleObjectProperty<>();
```

Überarbeiten Sie den Konstruktor sowie alle betroffenen Methoden. Erstellen Sie Getter und Setter für das neue Feld.

TASK 3: Database-Klasse

Ergänzen Sie das `CREATE TABLE`-Statement um ein neues Feld `contactType`.

Tipp: Da sich aktuell in der Datenbank ohnehin nur Testdaten befinden, kann eine bestehende Datenbank einfach gelöscht werden. Beim nächsten Start der Applikation sollte dann die Datenbank in der erweiterten Struktur neu erstellt werden.

TASK 4: ContactRepository-Klasse

Adaptieren Sie erforderlichenfalls die Methoden, sodass das neue Attribut korrekt verarbeitet wird.

TASK 5: ContactView-Klasse

Ersetzen Sie das ListView- durch ein TreeView-Control. Passen Sie auch die Variablen-Namen an.

Erweitern Sie die Detail-Ansicht um eine ComboBox zur Anzeige bzw. Auswahl des Kontakt-Typs.

Tipp: Sie könne die Auswahlmöglichkeiten für die ComboBox beim Initialisieren der View direkt auf Basis der Enumeration definieren:

```
private void init() {
   ...
   cmdContactType.getItems().addAll(ContactType.values());
   ...
}
```

TASK 6: ContactPresenter-Klasse

TreeView

TASK 6.1: TreeView aufbauen

Erstellen Sie eine Methode `buildTreeView()`, die Gruppen für die verschiedenen Kontakttypen anlegt und die Kontakte entsprechend zuordnet.

Beachten Sie:

- Jeder TreeView benötigt einen Root-Knoten. Für die Gruppierung der Kontakte nach Typ wird Dummy-Root-Knoten verwendet werden, der nicht angezeigt wird.
- Zuerst werden die Kontakte nach Typ in einer Map gruppiert.
- Anschließend wird für jeden Typ ein TreeItem angelegt und die zugehörigen Kontakte hinzugefügt.
- Der Root-Knoten wird schließlich dem TreeView zugewiesen und ausgeblendet.

```
    private void buildTreeView() {
        TreeItem<Object> root = new TreeItem<>(null); // Dummy-Root

        // Group contacts by type
        Map<ContactType, List<Contact>> contactsByType = contactList.stream()
                .collect(Collectors.groupingBy(Contact::getType));

        contactsByType.forEach((type, contacts) -> {
            // Create a node for each type
```

```
            TreeItem<Object> typeNode = new TreeItem<>(type);
            contacts.forEach(contact -> {
                // Create a node for each contact
                TreeItem<Object> contactNode = new TreeItem<>(contact);
                typeNode.getChildren().add(contactNode);
            });
            typeNode.setExpanded(true);
            root.getChildren().add(typeNode);
        });

        // set root node and hide it
        view.getTvContacts().setRoot(root);
        view.getTvContacts().setShowRoot(false);
    }
```

TASK 6.2: TreeView aktualisieren

Sorgen Sie dafür, dass die Methode `buildTreeView()` beim Initialisieren der View (`init()`) bzw. neuen Laden der Kontakte (`reloadContactList()`) aufgerufen wird.

Beachten Sie: Die Bindings für das ehemalige ListView-Control entfällt, da der TreeView keine direkte Bindung an die Kontaktliste hat, sondern über die `buildTreeView()`-Methode explizit aufgebaut wird.

TASK 6.3: TreeView-Listener zur Anzeige von Details

Überarbeiten Sie den Listener für die Auswahl eines Kontakts. Beachten Sie: Die TreeView-Elemente sind vom Typ `TreeItem<Object>`, da sie sowohl Gruppen- als auch Kontakt-Knoten enthalten. Ein Anzeigen von Details darf nur dann erfolgen, wenn ein Kontakt-Knoten ausgewählt wurde.

```
    private void addListeners() {
        ...
         if (newValue != null && newValue.getValue() instanceof Contact) {
             ...
         }
         ...
    }
```

TASK 6.4: Suche nach Kontakten

Passen Sie die Methode `searchContacts()` an, sodass im TreeView-Control nach Kontakten gesucht wird. Die Suche soll nur den Namen des Kontakts berücksichtigen. Die Gruppenknoten sollen nicht durchsucht werden.

Tipp: Die Kontakte befinden sich alle auf der gleichen Ebene.

ComboBox

TASK 6.5: ComboBox-Binding

Fügen Sie für die ComboBox ein Binding zum Kontakt-Typ des Modells hinzu.

TASK 6.6: ComboBox-Verhalten

Ergänzen Sie die Methoden `clearFields()` und `setEditMode()` um die Behandlung des Kontakt-Typs.

TASK 6.7: EventHandler

Passen Sie die EventHandler-Methoden `newContact()`, `editContact()`, `saveContacts()` und `deleteContact()` bei Bedarf an, sodass Kontakt-Typ korrekt verarbeitet wird.