**Protokoll Übung 1**

Gruppe: **Schwabakla**

– Robert Dziuba, Tobias Klatt & Inga Schwarze

1. **Properties implementieren**

Aufgabenteil A:

Wir schreiben eine Klasse Appointment, die folgende Attribute als StringProperties erhält: datum, titel, startzeit, endzeit, kategorie, notiz. Die Klasse Zeitrechner ist unsere Kontrollklasse, die die Zeiten überprüft. Neue Objekte von StringProperties werden mit SimpleStringProperty initialisiert.

Properties haben den Vorteil, dass wir im späteren Verlauf in unserem Programm mit diesen arbeiten können und das ermöglicht uns, diese Attribute zu „beaobachten“ (> Observable) und miteinander zu „verbinden“ (> Binding). Dadurch kann der Wert des Attributes an eine andere Property gebunden werden und wird somit direkt und ständig synchronisiert (bei jeder Änderung).

Wir haben drei Konstruktoren: einen Standartkonstruktor (leer), einen Kopierkonstruktor und einen, der ein Appointment initialisiert und alle Attribute setzt.

Wir haben Getter und Setter, die wir zwingend für diesen ADT brauchen. Wir setzen also unsere Attribute damit fest.

Getter: Setzen

Setter: Holen

Mit diesen können wir die Werte verändern. > Inhalt der Properties verändern.

Property: Hier werden die Properties direkt übergeben. Da wir Sachen herausgeben, nennen wir sie Getter – StringProperty. > Gesamte Klasse Property mit Inhalt wird zurückgegeben. Das ist später nützlich, da diese Klasse z.B. das Binding enthält.

To String Methode: Um den ADT auszugeben. In der Konsole erscheint dann eine Übersicht unseres Termineintrags.

Unsere Setter sorgen dafür, dass die eingegebenen Daten überprüft werden. Wir haben dazu Methoden geschrieben. Datum: testeDatumString() überprüft, ob ein Datum vorliegt und ob dieses im richtigen Format (testeDatumFormat()) ist. Außerdem überprüfen wir ob unser Datum im Wertebereich liegt (testeDatumWertebereich()). Analog dazu testen wir auch die Zeit: testeZeitString(), testeZeitFormat(), testeZeitWertebereich().

Aufgabenteil B:

Bei den Settern kontrollieren wir zusätzlich, ob die Endzeit tatsächlich hinter der Startzeit liegt und andersherum (Aufgabenteil B). Die Methoden heißen testeZeitFenster().

Es wird direkt eine Exception geworfen, wenn die Startzeit nicht vor der Endzeit liegt.

1. **Editierbare ListView**

Aufgabenteil A:

Wir schreiben eine neue Klasse ObservableContactDetails, die von unserer Klasse ContactDetails erbt. Wir haben fünf Attribute vom Typ StringProperty: vorname, nachname, adresse, telefonnummer, mail. Die Klasse hat dieselben Eigenschaften wie unsere ContactDetails, bis auf die Tatsache, dass es sich nun um SimpleStringProperties handelt, damit wir später mit Binding arbeiten können.

Drei Konstruktoren: Standartkonstruktor, der ruft unseren normalen Konstruktor auf. Der normale K. ruft zusätzlich den K. von ContactDetails auf. Der Kopierk. Ruft den normalen K. auf. Die Getter geben die Werte für unsere Attribute, die Setter setzen die Attribute für die Property. Auch hier haben wir wieder die Möglichkeit, die Klasse Property/ das Objekt zurückzugeben. > StringProperty vornameProperty > return vorname;

Außerdem wie gewöhnlich eine ToString-Methode und eine Main-Methode zum Ausführen, um einen Überblick über die Funktion der Klasse zu bekommen.

Aufgabenteil B:

Aufgabenteil C:

Aufgabenteil D:

1. **Editierbare Tabelle**

Aufgabenteil A:

Idee: Wir trennen unsere Klassen nach dem MVC Prinzip. Das bedeutet, wir haben folgende Klassen:

View: Für die grafische Oberfläche, ruft Control auf

Control: Für die Logik des Programms, füllt die View

TableCellFactory: Reagiert auf z.B. Klicks.

Als erstes schreiben wir die Klasse View: Sie ruft ein Objekt von Control auf. Control ruft die Methode fuelleTabelle() auf. Wir übergeben der Methode eine Pane. Hier wird dann der Inhalt angezeigt.

Wir schreiben die Klasse Control: Bei Starten dieser Klasse füllen wir das Adressbuch mit Dummy-Daten. Wir rufen dann die Methode fuelleTabelle() auf, die die Pane übergeben bekommt. In dieser Methode kreieren wir uns ein Array von ObservableContactDetails, in das wir alle unsere Kontakte packen.

Aufgabenteil B:

Aufgabenteil C:

Aufgabenteil E: