h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Programmieren, Algorithmen und Datenstrukturen II

Praktikum 4: Polymorphie, Dynamic cast, Ausnahmebehandlung, Unit-Tests

Sommersemester 2019

Prof. Dr. Arnim Malcherek

Allgemeine Hinweise zum Praktikum:

- Bereiten Sie die Aufgaben unbedingt zu Hause oder in einem freien Labor vor. Das beinhaltet:
  - Entwurf der Lösung
  - Codieren der Lösung in einem Qt-Creator-Projekt
- Die Zeit während des Praktikums dient dazu, die Lösung testieren zu lassen sowie eventuelle Korrekturen vorzunehmen.
- Das Praktikum dient auch zur Vorbereitung der praktischen Klausur am Ende des Semesters. Versuchen Sie also in Ihrem eigenen Interesse, die Aufgaben selbständig nur mit Verwendung Ihrer Unterlagen bzw. Ihres bevorzugten C++-Buches und ohne Codefragmente aus dem Netz zu lösen.
- Die Lösung wird nur dann testiert, wenn
  - sie erklärt werden kann bzw. Fragen zur Lösung beantwortet werden können.
  - das Programm ablauffähig und die Lösung nachvollziehbar ist.
  - die Hinweise oder Einschränkungen aus der Aufgabenstellung befolgt wurden.
- Zur Erinnerung hier noch einmal die Regeln des Praktikums, die schon in der Vorlesung besprochen wurden:
  - Sie arbeiten in 2er Gruppen.
  - Ein Testat gibt es nur zum jeweiligen Termin.
  - Abschreiben und Kopieren ist verboten.
  - Es gibt keine Noten. Die Bewertung ist lediglich erfolgreich / nicht erfolgreich.
  - Das Praktikum ist Zulassungsvoraussetzung für die Klausur. Hierfür müssen alle fünf Praktikumsübungen testiert sein.

Lernziele:

- Sie lernen, Polymorphie und abstrakte Klassen im Softwareentwurf zu berücksichtigen und bei der Implementierung einzusetzen.
- Sie verwenden Ausnahmebehandlung zur Erkennung und Behandlung von Fehlern im Programm.
- Sie können Unit-Tests programmieren und ausführen.

Wir erweitern das bereits bestehende Programm um zusätzliche Funktionalitäten und nutzen dabei Polymorphie aus.

Aufgabe 1

Erweitern Sie Ihr Klassenmodell zunächst um die Attribute `RentalCarReservation::insuranceType`, `HotelBooking::smoke` und `FlightBooking::seatPref` wie im angepassten Klassendiagramm in Abbildung 1 beschrieben (inklusive der Datentypen):

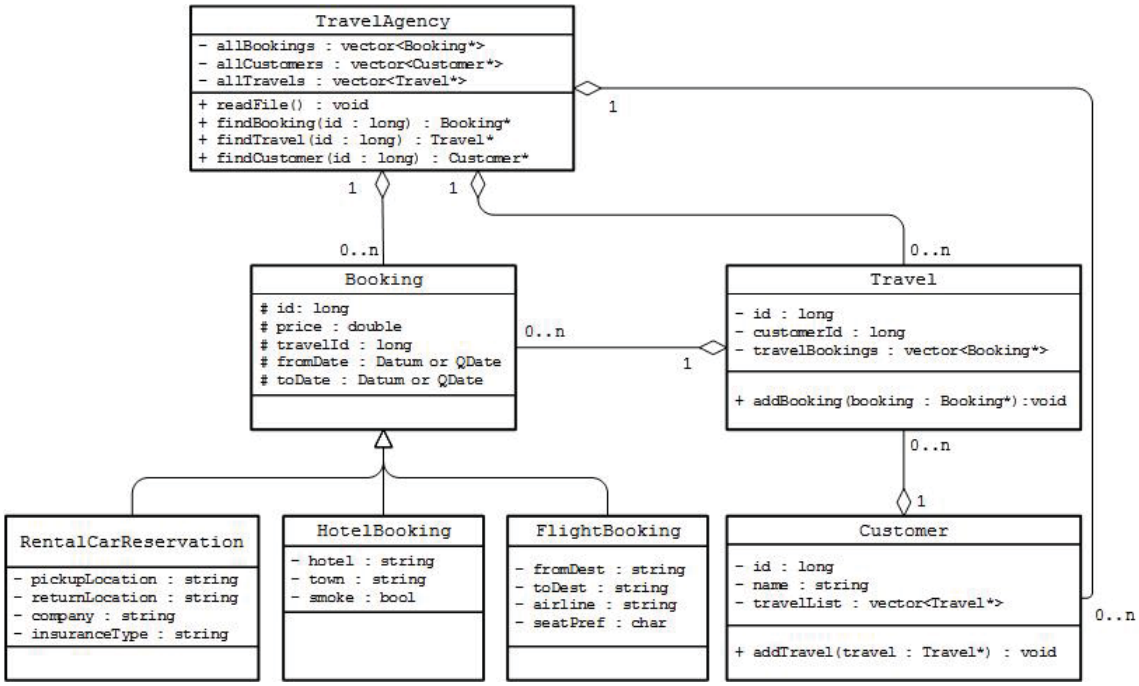

Abbildung 1: Klassendiagramm

Ändern Sie jetzt Ihre Methode `readFile()` so, dass Sie die neuen Attribute aus der Datei *bookings_praktikum4.txt* verarbeiten können. Die Beschreibung der Datei lautet jetzt:

```
F|35|277.01|20150602|20150603|7|4|Uli Stein|STR|WAW|Lufthansa|A
R|36|167.01|20150602|20150603|7|4|Uli Stein|CHOPIN FLUGH.|CHOPIN FLUGH.|Europcar|Diebstahlschutz
H|37|256.02|20150602|20150603|7|4|Uli Stein|Residence 1898|Warschau|1
F|38|288.93|20150603|20150603|7|4|Uli Stein|WAW|STR|Lufthansa|A
```

Spalte 1 beschreibt den Typ der Buchung. Die Spalten 9-12 hängen vom Typ ab. Im Einzelnen:

Spalte 1:	Typ der Buchung (`F` für Flight, `H` für Hotel, `R` für Rental Car)
Spalte 2:	ID der Buchung (Attribut `id`)
Spalte 3:	Preis der Buchung (Attribut `price`)
Spalte 4:	Start-Datum der Buchung im Format JJJJMMTT (Attribut `fromDate`)
Spalte 5:	End-Datum der Buchung im Format JJJJMMTT (Attribut `toDate`)
Spalte 6:	ID der Reise (Booking::travelId und Travel::id)
Spalte 7:	ID des Kunden (Customer::id und Travel::customerId)
Spalte 8:	Name des Kunden (Customer::name)
Spalte 9:	falls Typ = 'F': Startflughafen (`FlightBooking::fromDest`)
	falls Typ = 'R': Abholstation des Autos (`RentalCarReservation::pickupLocation`)
	falls Typ = 'H': Hotelname (`HotelBooking::hotel`)
Spalte 10:	falls Typ = 'F': Zielflughafen (`FlightBooking::toDest`)
	falls Typ = 'R': Rückgabestation des Autos (`RentalCarReservation::returnLocation`)
	falls Typ = 'H': Name der Stadt (`HotelBooking::town`)
Spalte 11:	falls Typ = 'F': Fluglinie (`FlightBooking::airline`)
	falls Typ = 'R': Autoverleihfirma (`RentalCarReservation::company`)
	falls Typ = 'H': Raucher- oder Nichtraucherzimmer (`HotelBooking::smoke`) (Wert 1 oder 0)
Spalte 12:	falls Typ = 'F': Sitzplatzpräferenz (A für „Aisle“ oder W für „Window“) (`FlightBooking::seatPref`)
	falls Typ = 'R': Gewünschte Versicherung (`RentalCarReservation::insurance`)
	falls Typ = 'H': leer

Aufgabe 2

Deklarieren Sie in der Klasse `Booking` eine virtuelle Methode `showDetails()`. Der Rückgabetyp dieser Methode sei vom Typ `string`. Machen Sie gleichzeitig die Klasse `Booking` abstrakt.

Implementieren Sie die virtuelle Methode `showDetails()` in den Klassen `HotelBooking`, `RentalCarReservatio` und `FlightBooking`. Der Rückgabewert sei ein `string` mit dem Inhalt:

- *Raucherzimmer* oder *Nichtraucherzimmer* für `HotelBooking`
- *Gang* oder *Fenster* für `FlightBooking`
- *Vollkasko* bzw. anderer Wert aus Datei für `RentalCarReservation`

Aufgabe 3

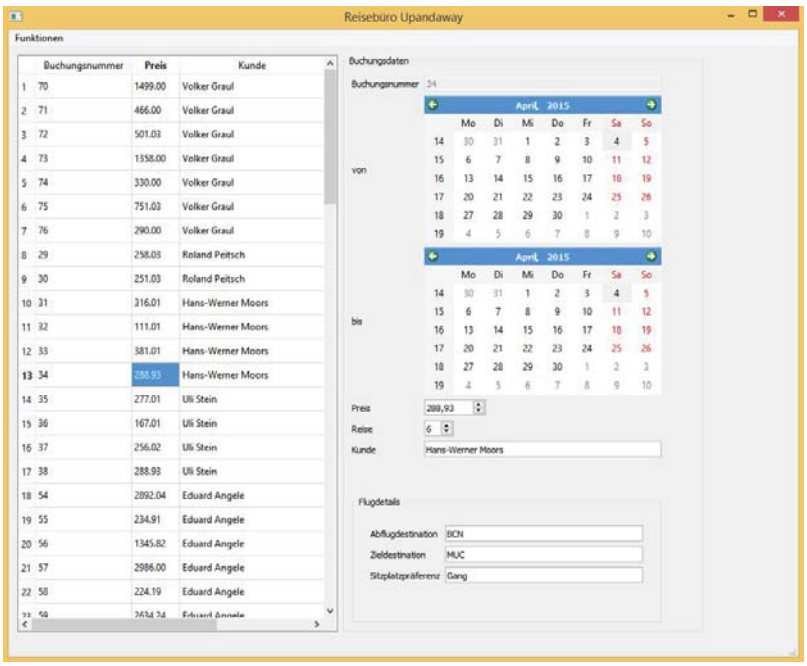

Abbildung 2: Beispiel-UI Flugbuchung

Erweitern Sie Ihr Detail-UI um die weiteren Felder der verschiedenen Buchungstypen. Einige Felder sind für alle Buchungstypen gleich, andere hängen vom Buchungstyp ab. Die Anforderung an Ihr

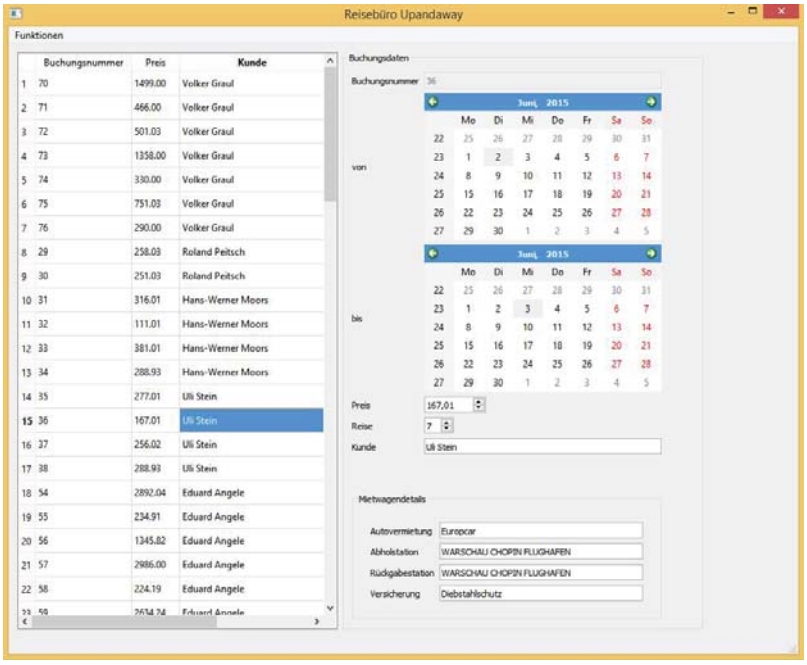

Abbildung 3: Beispiel-UI Mietwagenbuchung

UI ist, dass bei einem Doppelklick in die Tabelle alle Details der entsprechenden Buchung angezeigt werden (inklusive des Resultats der Methode `showDetails()`).

In den Beispielen (siehe Abb. 2, Abb. 3 und Abb. 4) habe ich für die unterschiedlichen Detaildaten ein `StackedWidget` und `Labels` und `LineEdits` verwendet. Sie können aber auch andere Lösungen und Widgets wählen (Popups, Tab-UIs etc.).
Tipp: Eventuell benötigen Sie einen `dynamic_cast`.

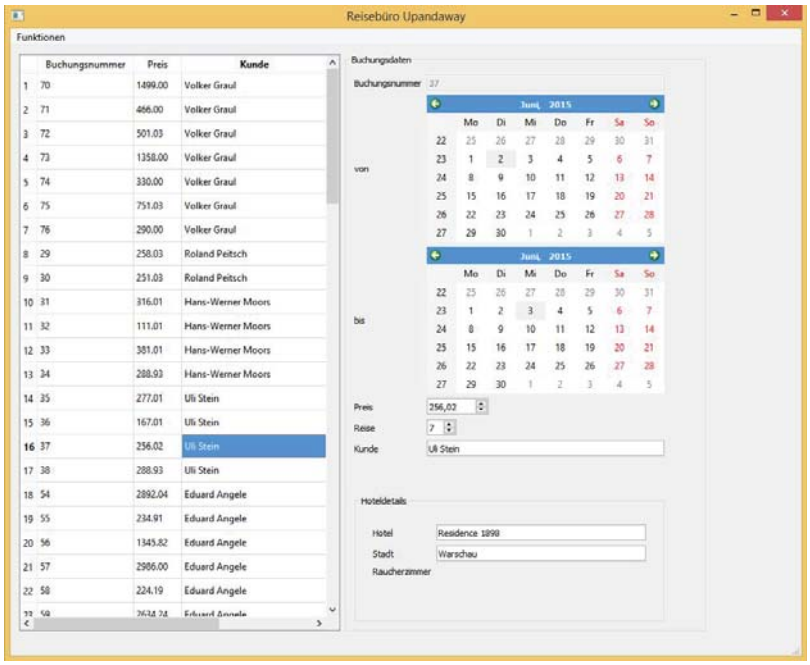

Abbildung 4: Beispiel-UI Hotelbuchung

Aufgabe 4

Bauen Sie in Ihr Programm eine Ausnahmebehandlung ein, die in der Lage ist, zu erkennen, dass in der Datei *bookings_praktikum4.txt* in einer Zeile ein Attribut fehlt. In diesem Fall soll der Leseprozess abgebrochen werden und eine Fehlermeldung ausgegeben werden, die die Zeilennummer der fehlerhaften Zeile enthält. Um dies zu testen, wird der jeweilige Praktikumsbetreuer in Ihrer Datei *bookings_praktikum4.txt* ein beliebiges Attribut löschen (siehe Beispiel in Abb. 5).

```
H|90|813.42|20151012|20151016|18|7|Wolfgang Pohl|Four Seasons Residence Club|Carlsbad|1
F|91|256.42|20151016|20151016|18|Wolfgang Pohl|SAN|ATL|Delta Airlines|W
H|92|430.31|20151016|20151018|18|7|Wolfgang Pohl|Hilton|Atlanta|1
F|93|215.78|20151018|20151018|18|7|Wolfgang Pohl|ATL|YYZ|Air Canada|W
```

Abbildung 5: Beispiel für Dateiveränderung

Der catch Befehl soll so in Ihrem Programm untergebracht sein, dass es möglich ist, die Datei *bookings_praktikum4.txt* nach Ausgabe einer Fehlermeldung (siehe Beispiel in Abb. 6) zu reparieren und das Programm fortzusetzen, ohne es neu starten zu müssen. Die Datei soll aber nur ganz oder gar nicht eingelesen werden, also nicht in Teilen. Was schon eingelesen wurde, wenn der Fehler erkannt wird, muss verworfen werden.

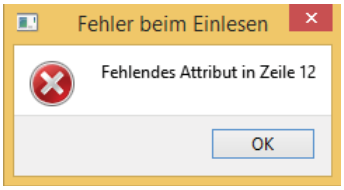

Abbildung 6: Beispiel für Fehlermeldung

Nach dem Reparieren der Datei wieder auf *Datei einlesen* klicken ergibt wieder die ursprüngliche Erfolgsmeldung in Abbildung 7.

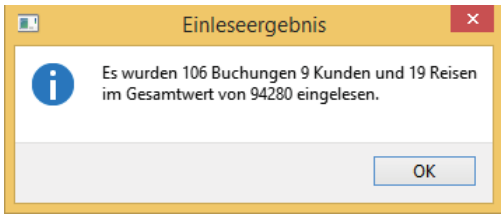

Abbildung 7: Meldung nach erfolgreichem Einlesen

Aufgabe 5

Ergänzen Sie die folgenden Unit-Tests:

- Eingelesen wurden 5 Mietwagenbuchungen der Firma Avis.
- Eingelesen wurden 3 Flugbuchungen mit der Fluglinie United Airlines.
- Eingelesen wurden 31 Buchungen mit einem Wert von mindestens 1000 €.

Aufgabe 6

Fügen Sie Ihrem Programm die Möglichkeit zum Speichern aller Buchungen (der eingelesenen sowie der neu angelegten) in einer Datei hinzu. Der Name der Datei soll über Dialog abgefragt werden. Ihr Programm aus Praktikum 3 müssen Sie dazu noch so erweitern, dass für neue Buchungen auch die Zusatzattribute abgefragt werden.