Hotel Reservierungssystem

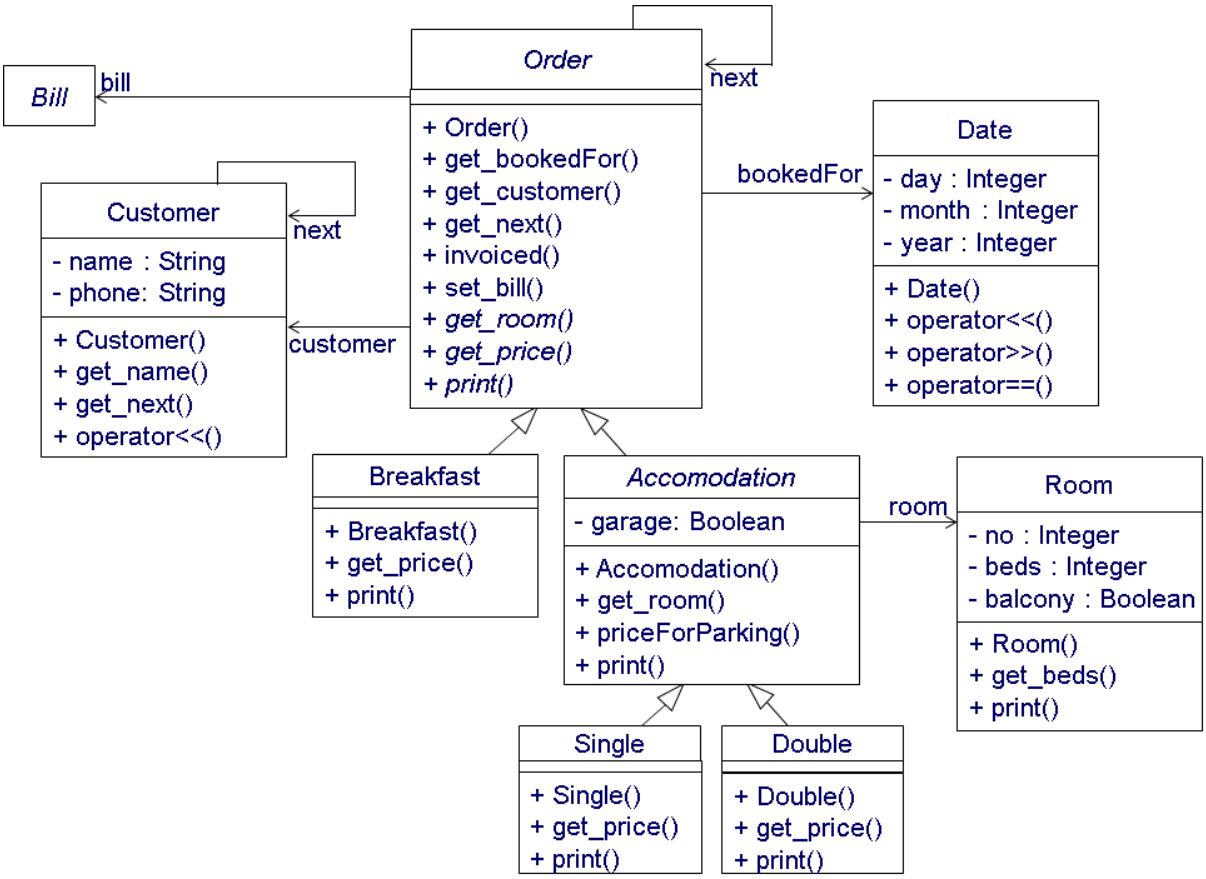
Lernziel:

Einfache Vererbung

Statische und dynamisches Binden

Überblick

Für das kleine Hotel garni "Duisburg Happy Sleeping" soll ein sehr einfaches Reservierungssystem für Einzel- und Doppelzimmer, Garagen und Frühstück erstellt werden.



Mögliche Parameter der (Member-)Funktionen, deren Typen und Rückgabewerte sind aus Platzgründen weggelassen. Das Reservierungssystem verwaltet die feste Anzahl von Zimmern (Klasse Room) des Hotels mit zugehörigen (optionalen) Garagen sowie die beliebige Anzahl möglicher Kunden/Kundinnen (Klasse Customer). Buchungen (Klasse Order) einer Kundin/eines Kunden für eine Unterkunft (Klasse Accomodation) in einem Zimmer können als Einzel- (Klasse

Single, Preis 49€) oder als Doppel-Belegung (Klasse Double, Preis 69€) erfolgen mit optionalem Frühstück (Klasse Breakfast, Preis 9,90€) und optionaler Garage (als Attribut garage, Preis 15€). Zusätzliches Frühstück soll für zusätzliche Personen als Gäste zusätzlich gebucht, Garagen allein sollen nicht gebucht werden können. Rechnungen (Klasse Bill) für die Buchungen der jeweiligen Kundinnen/Kunden werden in Zusatzaufgabe erstellt.

Teilaufgabe globale Variable und Klassen

Deklarieren (nicht definieren!) Sie fünf Klassen mit Namen Bill, Customer, Date, Order und Room. Definieren Sie eine konstante Variable Hotel als Zeichenkette mit dem Hotelnamen "Duisburg Happy Sleeping", eine konstante Variable numberOfRooms als natürliche Zahl mit dem Wert 4, ein Feld mit Namen rooms mit numberOfRooms Zeigern vom Typ Room, einen Zeiger customers initialisiert mit NULL vom Typ Customer und einen Zeiger orders initialisiert mit NULL vom Typ Order.

Teilaufgabe Klasse Date

Definieren Sie eine Klasse mit Namen Date mit

- drei privaten natürlichzahligen Attributen day, month und year.
- einem überladenen öffentlichen (Standard-) Konstruktor mit drei Default-Parametern für das Datum 09.01.2015, der die drei Attribute initialisiert.
- einem befreundeten überladenen Operator >> zur Eingabe eines Datums von einem Eingabedatenstrom.
- einem befreundeten überladenen Operator << zur Ausgabe eines Datums auf einen Ausgabedatenstrom. Das Datum soll immer jeweils zweistellig für den Tag und den Monat und vierstellig für das Jahr ausgegeben werden wie in den folgenden Beispielen: 08.02.2015, 13.02.2015, 01.11.2015, 31.12.2015).
- einem befreundeten überladenen Operator == zum Vergleich zweier Objekte dieser Klasse mit einem Booleschen Rückgabewert.

Teilaufgabe Klasse Room

Definieren Sie eine Klasse mit Namen Room mit

- zwei privaten natürlich-zahligen Attributen mit Namen no für die Raumnummer und beds für die Anzahl der Betten.
- einem privaten Booleschen Attribut mit Namen balcony das angibt, ob das Zimmer einen Balkon hat.
- einem öffentlichen Konstruktor mit drei Parametern zur Initialisierung der drei Attribute eines Objekts.
- einer öffentlichen Methoden mit Namen get_beds, die die Anzahl der Betten als Funktionswert zurück liefert.
- einer öffentlichen Methode mit Namen print, in der die Zeichenkette "room no.", die Raumnummer, mit einem Komma getrennt die Anzahl der Betten und

"bed(s)" sowie zusätzlich ", balcony" ausgegeben wird, falls das Zimmer einen Balkon besitzt (siehe Beispiele unten).

Teilaufgabe Klasse Customer

Definieren Sie eine Klasse mit Namen Customer mit

- zwei privaten Attributen mit Namen name für den Namen und phone für die Telefonnummer beide vom Typ Zeichenkette.
- einem privaten Attribut mit Namen next vom Typ Zeiger auf eine/einen Customer.
- einem öffentlichen Konstruktor mit zwei Zeichenketten-Parametern sowie einem Zeiger vom Typ Customer zur Initialisierung der drei Attribute. Der Name muss immer zwingend angegeben werden, der zweite Parameter für die Telefonnummer soll als Defaultparameter eine leere Zeichenkette "" sein, der dritte Defaultparameter soll als Defaultwert den Nullzeiger NULL haben und dem Aufbau einer Liste von beliebig vielen Kunden/Kundinnen dienen.
- zwei öffentliche Methoden mit Namen get_name und get_next, die die jeweiligen Werte der privaten Attribute als Wert der Funktion zurück liefern.
- einen befreundeten überladenen Operator << zur Ausgabe einer Referenz auf ein Objekt der Klasse als zweiten Parameter, in dessen Rumpf der Name gefolgt von ", phone: " und die Telefonnummer der Kundin/des Kunden auf den als ersten Parameter übergebenen Ausgabedatenstrom geschrieben werden (siehe Beispiele unten).

Teilaufgabe globale Funktion searchCustomer

Implementieren Sie eine Funktion

```
Customer* searchCustomer(string name);
```

die in der Liste der Kunden/Kundinnen (globale Variable customers) den ersten gefunden Kunden mit dem als Parameter übergebenen Namen sucht und einen Zeiger auf diesen zurück gibt oder NULL, falls der Name nicht gefunden wurde.

Teilaufgabe Klasse Order

Definieren Sie eine abstrakte Klasse mit Namen Order mit

- einem privaten Attribut mit Namen bookedFor vom Typ Date.
- einem privaten Zeiger mit Namen bill vom Typ Bill.
- einem privaten Zeiger mit Namen customer vom Typ Customer.
- einem privaten Zeiger mit Namen next vom Typ Order.
- einem öffentlichen Konstruktor mit drei Parametern. Der erste Parameter soll ein Objekt vom Typ Date sein, der zweite ein Zeiger auf eine Kundin/einen Kunden, der dritte ein Defaultparameter mit dem Wert NULL auf eine

Buchung vom Typ Order zum Aufbau einer Liste von beliebig vielen Buchungen. Die zugehörigen drei Attribute sollen mit den Werten initialisiert werden, die zugehörige Rechnung bill mit NULL.

- zwei öffentlichen Methoden mit Namen get_bookedFor und get_customer, die die jeweiligen Werte der privaten Attribute als Funktionswert zurück liefern.
- einer öffentliche Methode mit Namen invoiced, die true zurück liefern soll, wenn diese Buchung bereits in Rechnung gestellt ist, also der Zeiger bill auf ein Objekt vom Typ Bill verweist, sonst false.
- einer öffentlichen Methode mit Namen set_bill mit einem Zeiger-Parameter auf eine Rechnung, die den Wert des gleichnamigen Attributs setzt und damit diese Buchung mit einer gestellten Rechnung verknüpft.
- einer öffentlichen virtuellen Methode mit Namen get_room ohne Parameter, die immer NULL als einen Zeiger auf ein Objekt der Klasse Room zurück liefert.
- einer öffentlichen abstrakten virtuellen Methode mit Namen get_price ohne Parameter und mit einem double als Rückgabewert.
- einer öffentlichen abstrakten virtuellen Methode mit Namen print ohne Parameter und ohne Rückgabewert.

Teilaufgabe Klasse Breakfast

Definieren Sie eine Klasse mit Namen Breakfast abgeleitet von der Klasse Order mit

- einem öffentlichen Konstruktor mit einem Datum und einem Zeiger auf eine Kundin/einen Kunden als Parameter sowie als drittem Defaultparameter einem Zeiger vom Typ Order mit Defaultwert NULL (ein Frühstück ist eine zusätzliche optionale Buchung in der Liste aller Buchungen).
- einer öffentlichen virtuellen Methode mit Namen get_price, die als Wert 9,90€ für ein Frühstück zurück liefert (siehe Beispiel unten).
- einer öffentlichen virtuellen Methode mit Namen print, die das Datum für das Frühstück, rechtsbündig fix in 10 Zeichen mit 2 Nachkommastellen den über obige Funktion aufgerufenen Preis und dahinter " EUR: Breakfast"ausdruckt (siehe Beispiel unten).

Teilaufgabe Klasse Accomodation

Definieren Sie eine abstrakte Klasse mit Namen Accomodation abgeleitet von der abstrakten Klasse Order mit

-
- einem privaten Booleschen Attribut mit Namen garage.
- einem privaten Zeiger mit Namen room vom Typ Room.
- einem öffentlichen Konstruktor mit fünf Parametern zur Initialiserung der Attribute. Die ersten drei Parameter sollen zwingend ein Datum für die Übernachtung, ein Zeiger auf eine Kundin/einen Kunden und ein Zeiger auf ein (noch freies) Zimmer sein. Die letzten beiden Parameter sollen Defaultparameter für keine Garage als Defaultwert sowie ein Zeiger vom Typ Order mit Defaultwert NULL sein.

- eine geschützte Methode mit Namen priceForParking ohne Parameter, die im Fall einer gemieteten Garage15.00€, sonst 0.00€ als double zurück liefert.
- einer öffentlichen virtuellen Methode mit Namen get_room ohne Parameter, die den gleichnamigen Zeiger zurück liefert.
- eine öffentliche virtuelle Methode mit Namen print, die das Datum für die Übernachtung, rechtsbündig fix in 10 Zeichen mit 2 Nachkommastellen den über die Funktion get_price ermittelten Preis für die Übernachung und dahinter " EUR: " ausdruckt. Danach soll der dafür reservierte Raum ausgegeben werden sowie ", garage", falls eine Garage hinzugebucht wurde (siehe Beispiel unten).

Teilaufgabe Klasse Single

Definieren Sie eine Klasse mit Namen Single abgeleitet von der abstrakten Klasse Accomodation mit

- einem öffentlichen Konstruktor mit fünf Parametern zur Initialiserung der Attribute. Die ersten drei Parameter sollen zwingend ein Datum für die Übernachtung, ein Zeiger auf eine Kundin/einen Kunden und ein Zeiger auf ein (noch freies) Zimmer sein. Die letzten beiden Parameter sollen Defaultparameter für keine Garage als Defaultwert sowie ein Zeiger vom Typ Order mit Defaultwert NULL sein.
- eine öffentliche virtuelle Methode mit Namen get_price, die als Wert 49,00€ für eine Single-Übernachtung zurück liefert (siehe Beispiel unten).
- eine öffentliche virtuelle Methode mit Namen print, die die print-Funktion der direkten Oberklasse aufruft sowie die Zeichenkette " (as single)" dahinter ausdruckt (siehe Beispiel unten, Doppelzimmer können auch für eine Single-Übernachtung gebucht werden).

Teilaufgabe Klasse Double

Definieren Sie eine Klasse mit Namen Double abgeleitet von der abstrakten Klasse Accomodation mit

- einem öffentlichen Konstruktor mit fünf Parametern zur Initialiserung der Attribute. Die ersten drei Parameter sollen zwingend ein Datum für die Übernachtung, ein Zeiger auf eine Kundin/einen Kunden und ein Zeiger auf ein (noch freies) Zimmer sein. Die letzten beiden Parameter sollen Defaultparameter für keine Garage als Defaultwert sowie ein Zeiger vom Typ Order mit Defaultwert NULL sein. Falls die Zahl der Betten in dem Raum nicht mindestens 2 ist, soll eine Textmeldung "additional bed required" ausgegeben werden.
- eine öffentliche virtuelle Methode mit Namen get_price, die als Wert 69,00€ für eine Double-Übernachtung zurück liefert (siehe Beispiel unten).
- eine öffentliche virtuelle Methode mit Namen print, die die print-Funktion der direkten Oberklasse aufruft sowie die Zeichenkette " (as double)" dahinter ausdruckt (siehe Beispiel unten, Einzelzimmer können mit einem Zusatzbett auch für eine Double-Übernachtung gebucht werden).

Teilaufgabe globale Funktion searchFreeRoom

Implementieren Sie eine Funktion

Room* searchFreeRoom(Date date, unsigned int beds);

In dieser soll ein freies Zimmer mit mindestens beds Betten am Datum date gesucht und als Zeiger zurück gegeben werden. Ist kein Zimmer mit ausreichend Betten an diesem Datum mehr frei, soll der Nullzeiger zurück gegeben werden.
Hinweise: das globale Feld room enthält numberOfRooms Zeiger auf die numberOfRooms Zimmer des Hotels (verwenden Sie bei der Programmierung immer diese konstante Variable numberOfRooms, nie den direkten Wert 4). Sie können in einer (äusseren) Schleife für jedes Zimmer prüfen, ob für dieses (einfacher Vergleich der jeweiligen Zeiger auf Zimmer im Feldelement und im Buchungsobjekt) in der globalen Liste orders aller Buchungen eine Übernachtung zu diesem Datum (verwenden Sie den oben definierten Operator == für zwei Objekte Date beim Vergleich) vorliegt (hierfür ist eine zweite innere Schleife erforderlich).

Teilaufgabe Funktion main

Schreiben Sie eine Funktion main:

- Erzeugen Sie in einer ersten Schleife numberOfRooms/2 neue Zimmer auf dem Heap mit fortlaufender Zimmernummer 1, 2, ... mit jeweils 1 Bett und ohne Balkon und weisen den jeweiligen Zeiger auf diese jeweils den ersten numberOfRooms/2 Feldelementen von room zu.
- Erzeugen Sie in einer zweiten Schleife numberOfRooms/2 neue Zimmer auf dem Heap mit fortlaufender Zimmernummer numberOfRooms/2+1, numberOfRooms/2+2, ... mit jeweils 2 Betten und mit Balkon, und weisen den jeweiligen Zeiger auf diese den jeweils zweiten numberOfRooms/2 Feldelementen von room zu.
- Geben Sie in einer dritten Schleife alle numberOfRooms Zimmer des Hotels hotel aus.

Beispiel:

```
Duisburg Happy Sleeping
room no. 1, 1 bed(s)
room no. 2, 1 bed(s)
room no. 3, 2 bed(s), balcony
room no. 4, 2 bed(s), balcony
```

Geben Sie in einer Schleife als Menüauswahl aus:

```
HOTEL RESERVATION SYSTEM
Duisburg Happy Sleeping
e end
n new customer
p print all customers
a accomodation booking
b breakfast booking
s show all orders
your choice:
```

Implementieren Sie alle Menüpunkte:

- **`n`** neue Kundin/neuer Kunde: der Name und die Telefonnummer sollen eingegeben werden, ein neues Objekt auf dem Heap erzeugt und in die Liste **`customers`** eingefügt werden. Bevor eine Reservierung eingegeben werden kann, muss jeweils die Kundin/der Kunde im System hinterlegt sein.

  Beispiele:

  ```
  your choice: n
  name of customer: Elvis Presley
  phone number: +1 555 12345 678

  your choice: n
  name of customer: Lady Gaga
  phone number: +1 555 987654321

  your choice: n
  name of customer: Joe Cocker
  phone number: +1 555 111 222 333
  ```

- **`p`** Ausgabe alle Kundinnen/Kunden: die Liste **`customers`** soll auf dem Bildschirm ausgegeben werden.

Beispiel:

```
LIST OF CUSTOMERS
Joe Cocker, phone: +1 555 111 222 333
Lady Gaga, phone: +1 555 987654321
Elvis Presley, phone: +1 555 12345 678
```

- **`a`** Buchung einer Übernachtung: zuerst soll der Name eingegeben und die Kundin/der Kunde in der Liste gesucht werden (Fehlermeldung, falls nicht vorhanden, und keine Buchung). Dann sollen das Datum eingegeben werden und ob ein Einzel- oder Doppelzimmer gebucht werden soll. Falls an diesem Datum noch ein Zimmer mit 1 oder mit 2 Betten frei ist, soll gefragt werden, ob mit oder ohne Frühstück und mit oder ohne Garage. Danach soll ein geeignetes neues Single- oder Double-Objekt auf dem Heap erzeugt (mit oder ohne Garage im Konstruktor angeben) und in die Buchungsliste eingefügt werden. Falls mit Frühstück gebucht wird, sollen bei einem **`Single`** ein und bei einem **`Double`** zwei weitere Objekte vom Typ **`Breakfast`** erzeugt und in die Liste eingetragen werden.

Beispiele:

```
your choice: a
for which customer name: Lady Gaga
for which date: 9 1 2015
d for double room, s for single room: d
b for breakfast, w for without: b
g for garage, w for without: g

your choice: a
for which customer name: Angela Merkel
sorry customer not found

your choice: a
for which customer name: Elvis Presley
for which date: 9 1 2015
d for double room, s for single room: d
b for breakfast, w for without: w
g for garage, w for without: w

your choice: a
for which customer name: Joe Cocker
for which date: 9 1 2015
d for double room, s for single room: d
sorry no free room at this date

your choice: a
for which customer name: Joe Cocker
for which date: 9 1 2015
d for double room, s for single room: s
b for breakfast, w for without: b
g for garage, w for without: w
```

```
your choice: a
for which customer name: Joe Cocker
for which date: 10 1 2015
d for double room, s for single room: s
b for breakfast, w for without: b
g for garage, w for without: g
```

Beachten Sie, dass keine drei Doppelzimmer vorhanden sind und eine entsprechende Meldung **`"sorry no free room at this date"`** ausgegeben werden soll. Beachten Sie auch, dass durch die Reihenfolge erst Einzel-, dann Doppelzimmer bei der Buchung eines Einzelzimmers erst diese gesucht und gefunden werden, und falls alle belegt sind, auch freie Doppelzimmer für eine Person gebucht werden.

- **`b`** Frühstück für zusätzliche Gäste buchen: für einen vorher einzugebenden Kunden können für ein einzugebendes Datum weitere zusätzliche Frühstücke hinzu gebucht werden, für die jeweils ein neues Objekt vom Typ **`Breakfast`** auf dem Heap erzeugt und ebenfalls in die Liste aller Buchungen **`orders`** eingefügt wird.

Beispiel:

```
your choice: b
for which customer name: Joe Cocker
for which date: 9 1 2015
```

- **`s`** Ausgabe alle Buchungen: hier soll eine Ausgabe aller gespeicherten Buchungen erfolgen.

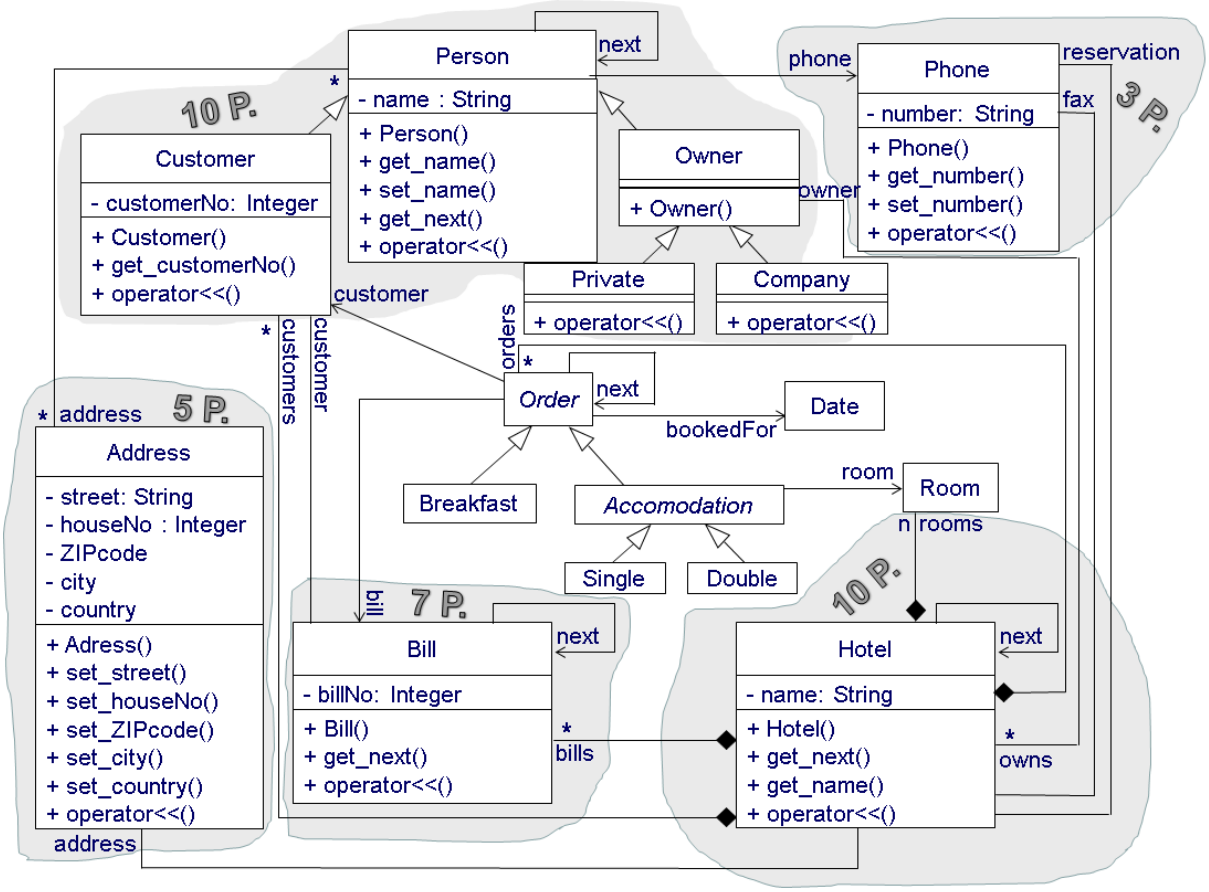
Beispiel:

```
your choice: s

LIST OF ORDERS
10.01.2015      64.00 EUR: room no. 1, 1 bed(s), garage (as single)
10.01.2015       9.90 EUR: Breakfast
09.01.2015      49.00 EUR: room no. 1, 1 bed(s) (as single)
09.01.2015       9.90 EUR: Breakfast
09.01.2015       9.90 EUR: Breakfast
09.01.2015      69.00 EUR: room no. 4, 2 bed(s), balcony (as double)
09.01.2015      84.00 EUR: room no. 3, 2 bed(s), balcony, garage (as double)
09.01.2015       9.90 EUR: Breakfast
09.01.2015       9.90 EUR: Breakfast
```

Aufgabe Hotelkette

Die Buchungssoftware aus Aufgabe (Hotel Reservierungssystem) war für die kleinen Hoteleigentümer so hilfreich und erfolgreich, dass die Software in dieser Zusatzaufgabe erweitert werden soll auf mehrere (baugleiche) Hotels/eine Hotelkette.



Folgende Erweiterungen sollen auf Wunsch der Kunden hinzu programmiert werden: 7 Punkte: Schreiben Sie eine Klasse für Rechnungen (Klasse **`Bill`**) mit mindestens den angegebenen Attributen, Memberfunktionen und Operatoren. Rechnungen werden über den Konstruktor erstellt, sollen eine eindeutige Rechnungsnummer haben (verwenden Sie dazu ein statisches Klassenattribut, das die jeweils letzte Rechnungsnummer speichert), werden für einen Kunden ausgestellt und enthalten als Rechnungsposten alle noch nicht bezahlten Bestellungen für ein einzelnes Hotel (also nicht für alle Hotels einer Hotelkette). Erweitern Sie das Menü in der **`main`**-Funktion um die Erstellung der Rechnung für einen bekannten Kunden und geben Sie die Rechnung aus.

Beispiel:

```
Duisburg Happy Sleeping
BillNo:   1
Customer: Ulrich Radtke, phone: 0203 379 5555
-------------------------
14.03.2015      69.00 EUR: room no. 3, 2 bed(s), balcony (as double)
14.03.2015       9.90 EUR: Breakfast
14.03.2015       9.90 EUR: Breakfast
13.03.2015      69.00 EUR: room no. 3, 2 bed(s), balcony (as double)
13.03.2015       9.90 EUR: Breakfast
13.03.2015       9.90 EUR: Breakfast
-------------------------
               177.60 EUR
=========================
```

- 10 Punkte

: Neben Kunden (Klasse **`Customer`**), die nun geändert von einer Klasse **`Person`** abgeleitet werden sollen, sollen auch Hotelbesitzer (Klasse **`Owner`**) betrachtet werden, die wiederum Privatpersonen (Klasse **`Private`**) oder Juristische Personen/Firmen (Klasse **`Company`**) sein können. Implementieren Sie die Klassen mindestens mit allen angegebenen Attributen, Memberfunktionen und Operatoren, und erweitern Sie das Menü in der **`main`**-Funktion um alle diese jeweils eingeben und ausgeben zu können.

- 5 Punkte:

Schreiben Sie eine Klasse für Adressen (Klasse Address) mit mindestens den angegebenen Attributen, Memberfunktionen und Operatoren. Erweitern Sie das Menü in der main-Funktion um Adressen von Kunden, Besitzern und Hotels jeweils eingeben und ausgeben zu können.

- 3 Punkte:

  Schreiben Sie eine Klasse für Telefonnummern (Klasse Phone) mit mindestens den angegebenen Attributen, Memberfunktionen und Operatoren. Erweitern Sie das Menü in der main-Funktion um Telefonnummern von Kunden, Besitzern und Hotels (Reservierung und Fax) jeweils eingeben und ausgeben zu können.

- 10 Punkte:

Schreiben Sie eine Klasse **`Hotel`** mit mindestens den angegebenen Attributen, Memberfunktionen und Operatoren - alle Hotels dürfen der Einfachheit halber gleich gebaut sein, also gleich viele Einzel- und Doppelzimmer haben wie das Hotel "Duisburg Happy Sleeping" in der obigen (Hotel Reservierungssystem) Aufgabe.

Erweitern Sie das Menü in der **`main`**-Funktion um einen Menüpunkt, in dem ein neues Hotel zur Liste der Hotels hinzugefügt werden kann, sowie einen weiteren Menüpunkt, bei dem jeweils alle Hotelnamen angezeigt werden und eins dieser Hotels ausgewählt werden kann, für das alle nachfolgend aufgerufenen Menüpunkte dann jeweils gelten sollen. Ändern Sie alle vorhandenen Funktionalitäten einschließlich der Suche nach einem Kunden oder nach einem freien Zimmer entsprechend so ab, dass nur das jeweils ausgewählte Hotel berücksichtigt wird.

Beispiel

```
your choice: h
list of hotels:
1 Duisburg Happy Sleeping
2 Essen Happy Sleeping
3 Bochum Happy Sleeping
choice of hotel: 2

HOTEL RESERVATION SYSTEM
Essen Happy Sleeping
e end
h hotel choice
```

Sie dürfen statt der Listen mit **`next`**-Zeigern und **`get_next`**-Memberfunktionen auch alternativ Template-Klassen wie **`<vector>`** oder **`<list>`** verwenden sowie zugehörige Iteratoren.