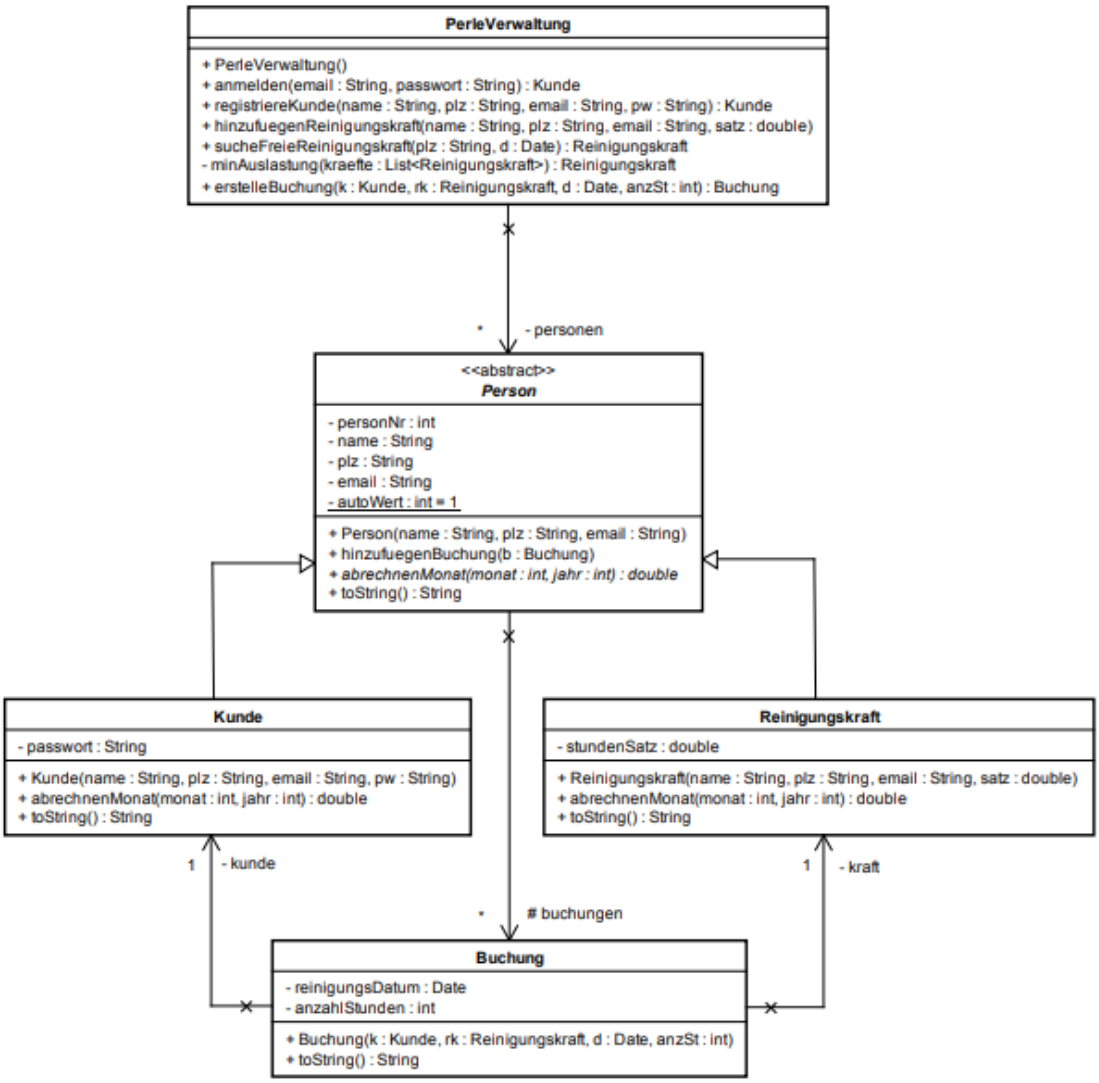
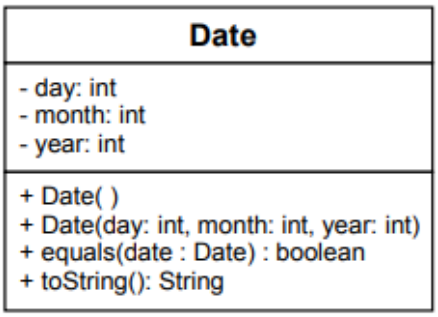
Aufgabenblatt: UML – Realisation von Assoziazionen

(1.) Objektorientierte Entwicklung eines Client/Server-Systems (2018-B)
Das Unternehmen plant eine Online-Plattform, um Kunden selbstständig tätige Reinigungskräfte zu vermitteln. Die Anwendung soll folgende Anforderungen erfüllen:
– Kunden müssen sich auf dem Online-Portal anmelden.
– Ein Kunde kann eine Reinigungskraft für ein Datum und eine Anzahl von Stunden buchen.
– Für jede erfolgreiche Buchung erhält das Unternehmen eine Vermittlungsgebühr.
– Die Abrechnung der Gebühren erfolgt monatlich



(a.) Überführen Sie die im Klassendiagramm dargestellten Klassen Person und Reinigungskraft in Anweisungen einer objektorientierten Programmiersprache und implementieren Sie die aufgeführten Methoden.
Hinweise: Bei der Umsetzung der Methode abrechnenMonat() ist folgendes zu berücksichtigen:
– Die Vermittlungsgebühr beträgt 20 % auf die Leistungshöhe und ist vom Kunden und der Reinigungskraft zu gleichen Teilen zu tragen.
– Umfasst die Buchung weniger als 3 Stunden, erhält die Reinigungskraft zusätzlich eine Fahrtkostenpauschale von 10 EURO, die vom Kunden zu entrichten ist.
Die Dokumentation der Klasse List und Datum für (c.) finden Sie anschließend.
Anmerkung: List entspricht ArrayList in Java

Kurzbeschreibung der Klasse `Date`



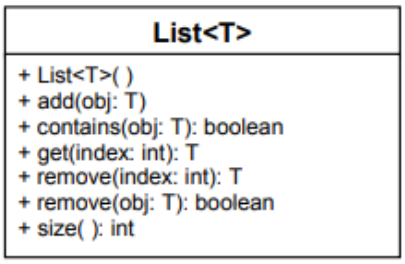
`Date()`
erzeugt ein `Date`-Objekt mit dem aktuellen Systemdatum.

`Date(day: int, month: int, year: int)`
erzeugt ein `Date`-Objekt mit den Werten von `day`, `month` und `year`.

`equals(date: Date): boolean`
liefert `true`, wenn das Datum des `Date`-Objekts und das des Parameters `date` gleich ist.

`toString(): String`
liefert eine String-Repräsentanz des `Date`-Objekts im Format `"dd.mm.yyyy"`.

Die Container-Klasse `List`



Mithilfe der Klasse `List` können Objekte im Arbeitsspeicher in einer linearen Liste verwaltet werden.

`List` repräsentiert eine generische Liste mit Elementen des Typs `T` in festgelegter Reihenfolge, auf die sowohl wahlfrei als auch sequenziell zugegriffen werden kann.

Kurzbeschreibung der Klasse `List`:

`add(obj: T)`
Hängt das Objekt `obj` vom Typ `T` am Ende der Liste an.

`contains(obj: T): boolean`
Liefert `true`, wenn das Objekt `obj` in der Liste enthalten ist, sonst `false`.

`get(index: int): T`
Liefert das Listenelement an der Position `index` zurück bzw. **`null`**, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

`remove(index: int): T`
Entfernt das Listenelement an der Position `index`. Liefert das entfernte Element zurück bzw. **`null`**, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

`remove(obj: T): boolean`
Entfernt das Objekt `obj` aus der Liste. Falls `obj` mehrmals in der Liste enthalten ist, wird nur das erste Vorkommen entfernt. Der Rückgabewert ist `true`, falls das Objekt gefunden und entfernt wurde, sonst `false`.

`size(): int`
Liefert die Anzahl der Elemente in der Liste zurück.

(b.) Überführen Sie die im Klassendiagramm (Material 3) dargestellte Klasse PerleVerwaltung in Anweisungen einer objektorientierten Programmiersprache und implementieren Sie den Konstruktor sowie die Methoden anmelden() und registriereKunde(). Hinweise: Die Dokumentation der Klasse List finden Sie in Material 4. Die Methode anmelden() muss prüfen, ob es sich bei einer Person um ein Kunden-Objekt handelt.
(c.) Die Methode sucheFreieReinigungskraft(plz:String, d:Date):
Reinigungskraft ermittelt aus der Liste der Personen alle Reinigungskräfte, bei denen die übergebene PLZ identisch ist und zu dem übergebenen Datum noch keine Buchungen existieren. Zurückgegeben wird die Reinigungskraft, für die im laufenden Monat die wenigsten Stunden gebucht sind. Die Methode liefert null zurück, wenn keine freie Reinigungskraft gefunden wurde.
Entwickeln Sie den Algorithmus dieser Methode und zeichnen Sie ihn in Form eines Struktogramms. Alternativ/ und: Kodieren Sie die Methode!

Aufgabenblatt: UML – Realisation von Assoziationen - Lösungen

(a.)

```
public abstract class Person {
   private int personNr;
   private String name;
   private String plz;
   private String email;
   private static int autoWert = 1;
   protected List < Buchung > buchungen;
   public Person(String name, String plz, String email) {
      this.personNr = autoWert++;
      this.name = name;
      this.plz = plz;
      this.email = email;
      buchungen = new List < Buchung > ();
   }
   public void hinzufuegenBuchung(Buchung b) {
      buchungen.add(b);
   }
   public abstract double abrechnenMonat(int monat,
      int jahr);
   public String toString() {
      return name + " (Personennummer: " + personNr + "), " +
         email + ", PLZ " + plz;
   }
}
public class Reinigungskraft extends Person {
   private double stundenSatz;
   public Reinigungskraft(String name, String plz,
      String email, double satz) {
      super(name, plz, email);
      this.stundenSatz = satz;
   }
   public double abrechnenMonat(int monat, int jahr) {
      double betrag = 0;
      for (Buchung b: buchungen) {
         if (b.getReinigungsDatum().getMonth() == monat &&
            b.getReinigungsDatum().getYear() == jahr) {
            betrag += b.getAnzahlStunden() * stundenSatz * 0.9;
            if (b.getAnzahlStunden() < 3) {
               betrag += 10;
            }
         }
      }
      return betrag;
   }
   public String toString() {
      return "Reinigungskraft " + super.toString() +
         ", Stundensatz " + stundenSatz + " EURO";
   }
}
```

(b.)

```
public class PerleVerwaltung {
   private List < Person > personen;

   private PerleVerwaltung() {
      personen = new List < Person > ();
   }
   public Kunde anmelden(String email, String passwort) {
      Kunde gesucht = null;
      for (Person p: personen) {
         if (p instanceof Kunde) {
            Kunde k = (Kunde) p;
            if (k.getEmail().equals(email) &&
               k.getPasswort().equals(passwort)) {
               gesucht = k;
               break;
            }
         }
      }
      return gesucht;
   }
   public Kunde registriereKunde(String name, String plz,
      String email, String pw) {
      Kunde k = new Kunde(name, plz, email, pw);
      personen.add(k);
      return k;
   }
}
```

(c.)