Name: _____

Abiturprüfung 2022

Informatik, Leistungskurs

Aufgabenstellung:

In der Turing-Schule werden häufig unterschiedliche Schulveranstaltungen (z. B. Schulmusical, Schultheater) aufgeführt. In der Vergangenheit bildeten sich vor dem Einlass für eine Veranstaltung teils lange Warteschlangen.
Um mehr Planungssicherheit für alle Personen zu gewährleisten, sollen für eine Veranstaltung die Sitzplätze nun im Vorfeld reserviert werden können. Die Schülerinnen und Schüler des Informatikprojektkurses möchten daher ein Buchungssystem für diese Veranstaltungen entwickeln.
Es hat sich bewährt, dass die Aula für jede Veranstaltung individuell bestuhlt wird. Dabei werden die Stühle immer in Sitzplatzreihen angeordnet. Bei einer Veranstaltung werden pro Sitzplatzreihe immer gleich viele Stühle aufgestellt.
Eine Person kann für eine Veranstaltung auch mehrere Sitzplätze reservieren. Falls eine Veranstaltung bereits ausgebucht ist, kann sich eine Person – für mehrere Platzwünsche auch mehrfach – auf die Warteliste setzen lassen. Für eine Person wird – nur zum Zweck der schnellen Erreichbarkeit bei Veranstaltungsabsagen – die Telefonnummer gespeichert.

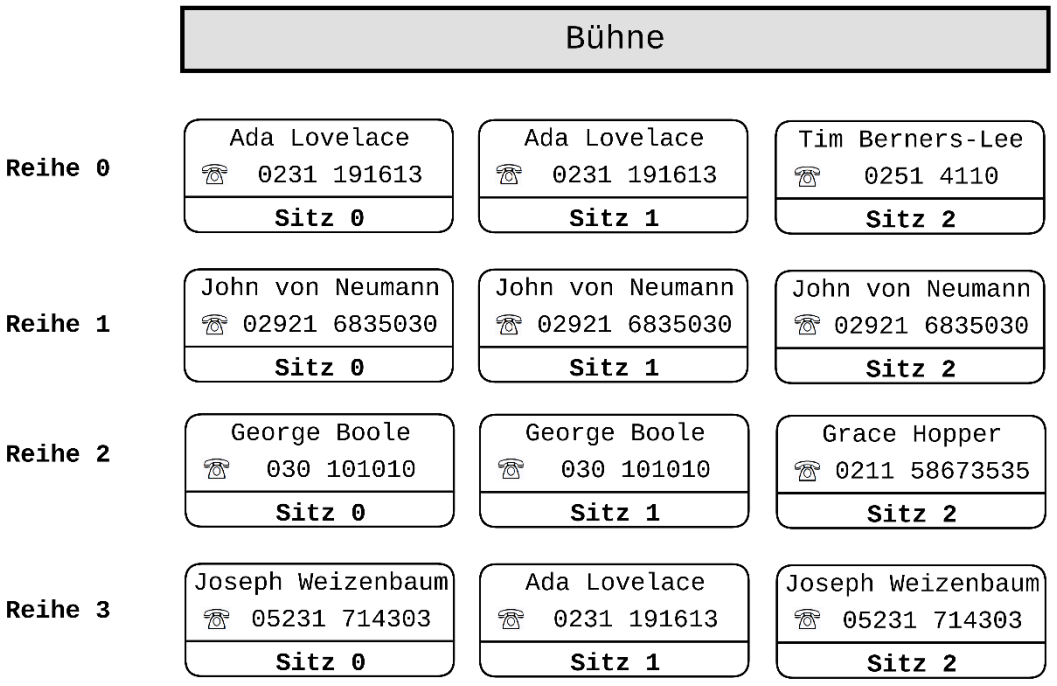

Abbildung 1: Darstellung der Reservierungen der Sitzplätze für eine Veranstaltung mit vier Sitzplatzreihen und jeweils drei Plätzen pro Reihe mit Beispieldaten

Name: _____

Eine erste Modellierung für die Verwaltung der Veranstaltungen ist im folgenden Implementationsdiagramm dargestellt. Die Dokumentationen der Klassen befinden sich im Anhang.

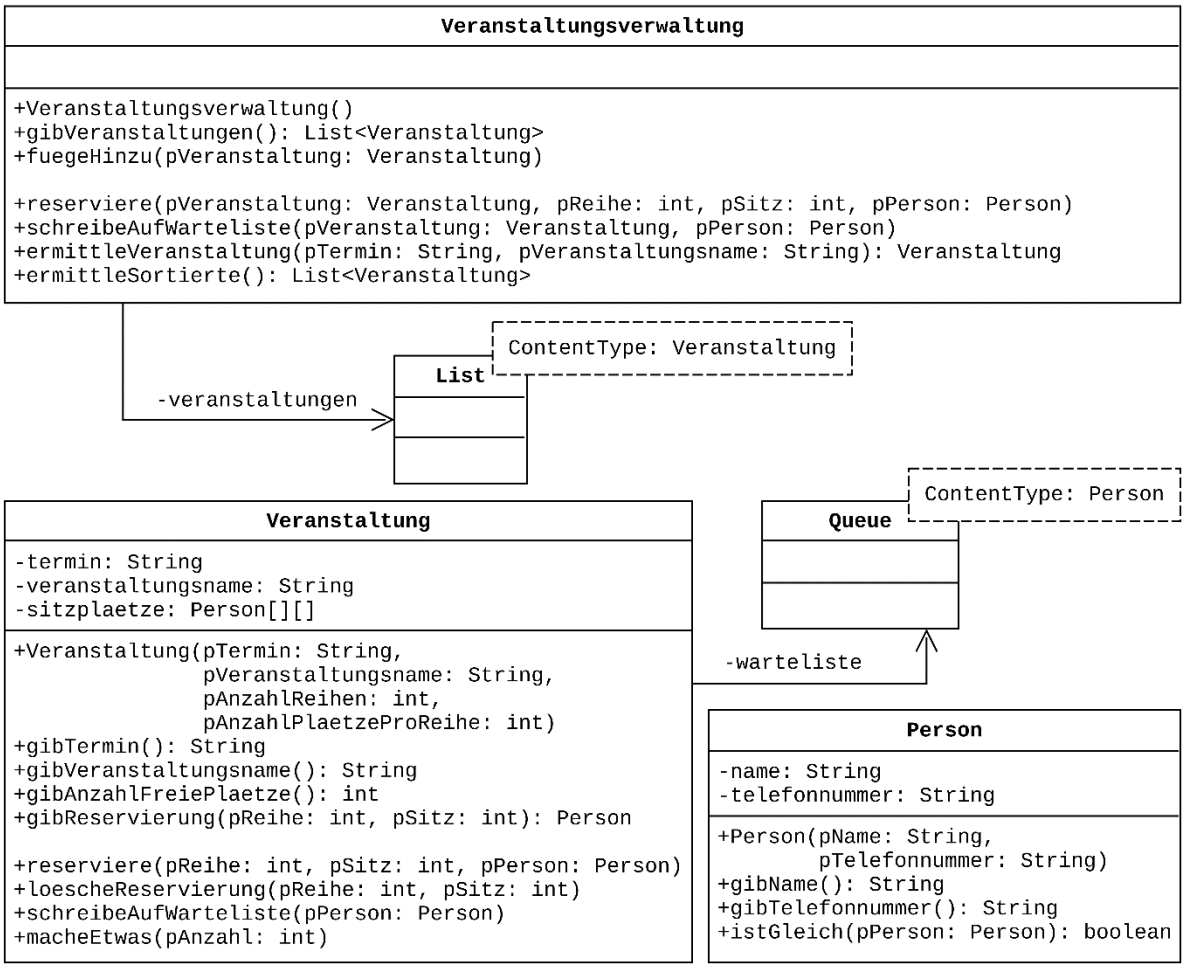

Abbildung 2: Teilmodellierung als Implementationsdiagramm

a) *Beschreiben Sie die Beziehungen zwischen den Klassen der in Abbildung 2 gegebenen Teilmodellierung.*

Begründen Sie im Sachkontext, warum es sinnvoll ist, die Datensammlungen für die Warteliste in Form einer Schlange (`Queue`*) zu realisieren.*

Begründen Sie im Sachkontext, warum es sinnvoll ist, die Sitzplätze mit den zugehörigen Reservierungen der Personen als zweidimensionales Feld (Array) und nicht als dynamische Datenstruktur zu realisieren.

(3 + 2 + 2 Punkte)

Name: _____

b) Eine Schülerin entdeckt in der Klasse `Veranstaltung` die undokumentierte Methode `macheEtwas`.

```
 1 public void macheEtwas(int pAnzahl) {
 2   if (pAnzahl > 0) {
 3     int reihen = sitzplaetze.length;
 4     int sitze = sitzplaetze[0].length;
 5     int rNeu = reihen + pAnzahl;
 6     Person[][] sitzplaetzeNeu = new Person[rNeu][sitze];
 7     for (int r = 0; r < reihen; r++) {
 8       for (int s = 0; s < sitze; s++) {
 9         sitzplaetzeNeu[r][s] = sitzplaetze[r][s];
10       }
11     }
12     for (int r = reihen; r < rNeu; r++) {
13       for (int s = 0; s < sitze; s++) {
14         if (sitzplaetzeNeu[r][s] == null) {
15           sitzplaetzeNeu[r][s] = warteliste.front();
16           warteliste.dequeue();
17         }
18       }
19     }
20     sitzplaetze = sitzplaetzeNeu;
21   }
22 }
```

Angenommen, für die in Abbildung 1 dargestellte Veranstaltung wird die Methode `macheEtwas` mit dem Parameter `pAnzahl = 2` aufgerufen.

Gehen Sie davon aus, dass das Objekt `warteliste` wie folgt belegt ist:

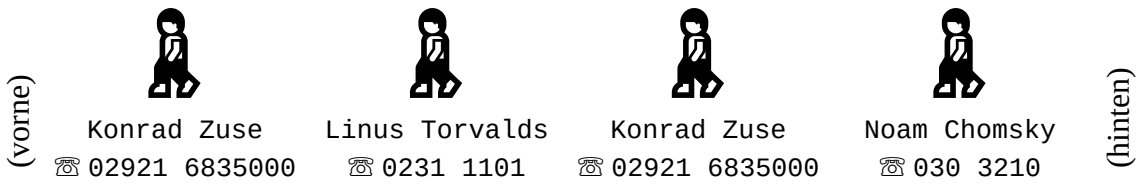

Abbildung 3: Personenobjekte, die von der Warteliste (Objekt `warteliste`) verwaltet werden

Analysieren Sie die Methode für diese Belegungen bei Anwendung auf die Beispieldaten aus Abbildung 1 und erläutern Sie die Funktionsweise der Methode sowie die Veränderungen der Belegung der Sitzplätze und der Warteliste.

Erläutern Sie die Aufgabe der Methode im Sachzusammenhang.

Beurteilen Sie im Sachkontext die Bedeutung der Verzweigung in Zeile 2.

Beurteilen Sie die Notwendigkeit der Verzweigung in Zeile 14.

(6 + 3 + 2 + 2 Punkte)

Name: _____

c) Die Klasse `Veranstaltungsverwaltung` soll die Methode mit dem folgenden Methodenkopf erhalten:

```
public List<Veranstaltung> ermittleSortierte()
```

Die Methode soll eine neue Liste mit allen Veranstaltungsobjekten zurückliefern. Diese neue Liste soll nach der Anzahl der noch freien Plätze absteigend sortiert sein. Bei gleicher Anzahl der noch freien Plätze spielt die Reihenfolge dieser Veranstaltungen untereinander keine Rolle.

Entwickeln und erläutern Sie ein algorithmisches Verfahren für die Methode `ermittleSortierte`.

Implementieren Sie die Methode `ermittleSortierte`.

(5 + 7 Punkte)

d) Die Schülerinnen und Schüler möchten in Rücksprache mit der Schulleitung ihr Projekt erweitern. Folgende Anforderungen sollen zusätzlich umgesetzt werden:

i. Die Schule möchte in Zukunft auch für jede Veranstaltung einen einheitlichen Eintrittspreis pro Sitzplatz erheben können. Dieser Eintrittspreis soll gespeichert werden. Der Eintrittspreis sowie die Summe der Eintrittsgelder auf Basis der Reservierungen sollen auch abgerufen werden können.
ii. Für jede Veranstaltung soll die Frage beantwortet werden können, ob für eine bestimmte Sitzplatzreihe eine bestimmte Anzahl an Sitzplätzen nebeneinander noch frei ist. Falls ja, dann soll die Nummer des ersten freien Sitzes der nebeneinander liegenden Sitzplätze zurückgeliefert werden. Falls für diese Veranstaltung die nebeneinander liegenden Plätze in der entsprechenden Anzahl in der gewünschten Reihe nicht mehr frei sind, muss auch diese Information in einer geeigneten Form zurückgegeben werden.
iii. Jede Person soll jeweils ihre gebuchten Veranstaltungen zurückliefern können.

Modellieren Sie die oben genannten Anforderungen als Erweiterung des Implementationsdiagramms aus Abbildung 2.

Erläutern Sie, wie Sie die zweite und dritte Anforderung in Ihrem Implementationsdiagramm realisieren.

Hinweis: Unveränderte Klassen, Klassenbeziehungen, Methoden und Attribute aus dem Implementationsdiagramm in Abbildung 2 müssen nicht angegeben bzw. dargestellt werden.

(9 + 4 Punkte)

Name: _____

e) Bei kurzfristigen Veranstaltungsabsagen konnte man leider nicht immer alle Personen telefonisch rechtzeitig erreichen. Daher soll nur zum Zweck der schnellen Erreichbarkeit bei Veranstaltungsabsagen zusätzlich zwingend die E-Mail-Adresse einer Person verwaltet werden.
Außerdem legt der Schulleiter der Projektgruppe eine E-Mail mit der Bitte um Prüfung vor. In der E-Mail wird ausschließlich auf zukünftige Schulveranstaltungen hingewiesen. Der Schulleiter argumentiert, dass er in Zukunft gerne monatlich so eine E-Mail als Serviceleistung an alle im Buchungssystem gespeicherten Personen versenden möchte, da sich diese ja ohnehin für die Schulveranstaltungen interessierten.

Beurteilen Sie den Vorschlag des Schulleiters aus der Perspektive des Datenschutzes.
(5 Punkte)

Zugelassene Hilfsmittel:

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse Veranstaltungsverwaltung

Objekte der Klasse **`Veranstaltungsverwaltung`** verwalten Veranstaltungen.

Auszug aus der Dokumentation der Klasse `Veranstaltungsverwaltung`

`Veranstaltungsverwaltung()`

Ein Objekt der Klasse `Veranstaltungsverwaltung` wird initialisiert.

`List<Veranstaltung> gibVeranstaltungen()`

Eine Liste mit allen Veranstaltungen wird zurückgeliefert. Wenn keine Veranstaltung verwaltet wird, dann wird eine leere Liste zurückgegeben.

`void fuegeHinzu(Veranstaltung pVeranstaltung)`

Die im Parameter übergebene Veranstaltung wird hinzugefügt.

`void reserviere(Veranstaltung pVeranstaltung, int pReihe, int pSitz, Person pPerson)`

Für die Veranstaltung `pVeranstaltung` wird für den Platz (`pReihe`, `pSitz`) für die Person eine Reservierung vorgenommen. Falls für die entsprechende Veranstaltung an dem Platz bereits eine Reservierung vorliegt, dann wird diese überschrieben. Falls der Platz nicht existiert, `pVeranstaltung` `null` ist oder `pPerson` `null` ist, dann geschieht nichts.

`void schreibeAufWarteliste(Veranstaltung pVeranstaltung, Person pPerson)`

Das Personenobjekt `pPerson` wird an die Warteliste für die Veranstaltung `pVeranstaltung` angehängt. Falls `pVeranstaltung` `null` ist oder `pPerson` `null` ist, dann geschieht nichts.

`Veranstaltung ermittleVeranstaltung(String pTermin, String pVeranstaltungsname)`

Für den im Parameter übergebenen Termin und Veranstaltungsnamen wird das entsprechende Veranstaltungsobjekt zurückgeliefert. Falls es zu dem Termin keine Veranstaltung mit dem Namen gibt, wird `null` zurückgeliefert.

`List<Veranstaltung> ermittleSortierte()`

Die Methode soll eine neue Liste mit allen Veranstaltungsobjekten zurückliefern. Diese neue Liste soll nach der Anzahl der noch freien Plätze absteigend sortiert sein. Bei gleicher Anzahl der noch freien Plätze spielt die Reihenfolge dieser Veranstaltungen untereinander keine Rolle.

Name: _____

Die Klasse Veranstaltung

Objekte der Klasse **`Veranstaltung`** verwalten Termine, Veranstaltungsnamen und die Sitzplätze mit den entsprechenden Reservierungen der Personen.

Auszug aus der Dokumentation der Klasse `Veranstaltung`

```
Veranstaltung(String pTermin, String pVeranstaltungsname,
              int pAnzahlReihen,
              int pAnzahlPlaetzeProReihe)
```

Ein Objekt der Klasse `Veranstaltung` wird initialisiert mit dem Termin, dem Veranstaltungsnamen, der Anzahl der Sitzplatzreihen und der Anzahl der Plätze pro Reihe.

```
String gibTermin()
```

Der Termin der Veranstaltung wird zurückgeliefert.

```
String gibVeranstaltungsname()
```

Der Veranstaltungsname der Veranstaltung wird zurückgeliefert.

```
int gibAnzahlFreiePlaetze()
```

Die Anzahl der noch freien Plätze der Veranstaltung wird zurückgeliefert.

```
Person gibReservierung(int pReihe, int pSitz)
```

Das Personenobjekt mit einer Reservierung für den Platz (`pReihe`, `pSitz`) wird zurückgeliefert. Falls der Platz nicht reserviert ist, wird `null` zurückgeliefert.

```
void reserviere(int pReihe, int pSitz, Person pPerson)
```

Für die Veranstaltung wird für den Platz (`pReihe`, `pSitz`) für die Person eine Reservierung vorgenommen. Falls an dem Platz bereits eine Reservierung vorliegt, dann wird diese überschrieben. Falls der Platz nicht existiert oder `pPerson` `null` ist, dann geschieht nichts.

```
void loescheReservierung(int pReihe, int pSitz)
```

Für die Veranstaltung wird die Reservierung für den Platz (`pReihe`, `pSitz`) gelöscht. Falls der Platz nicht existiert, dann geschieht nichts.

```
void schreibeAufWarteliste(Person pPerson)
```

Das Personenobjekt `pPerson` wird an die Warteliste für die Veranstaltung angehängt. Falls `pPerson` `null` ist, dann geschieht nichts.

```
void macheEtwas(int pAnzahl)
```

Diese Methode ist Bestandteil der Teilaufgabe b).

Name: _____

Die Klasse `Person`

Objekte der Klasse **`Person`** verwalten einen Namen und eine Telefonnummer einer Person.

Auszug aus der Dokumentation der Klasse `Person`

`Person(String pName, String pTelefonnummer)`

Ein Objekt der Klasse `Person` wird initialisiert mit dem Namen und der Telefonnummer.

`String gibName()`

Der Name der Person wird zurückgeliefert.

`String gibTelefonnummer()`

Die Telefonnummer der Person wird zurückgeliefert.

`boolean istGleich(Person pPerson)`

Wenn das Personenobjekt inhaltlich mit dem im Parameter übergebenen Personenobjekt `pPerson` gleich ist, dann wird `true` zurückgeliefert und sonst `false`.
Für die Inhaltsgleichheit müssen die jeweiligen Namen und Telefonnummern übereinstimmen.

Name: _____

Die generische Klasse `Queue`

Objekte der generischen Klasse `Queue` (Warteschlange) verwalten beliebige Objekte vom Typ `ContentType` nach dem First-In-First-Out-Prinzip, d. h., das zuerst abgelegte Objekt wird als erstes wieder entnommen. Alle Methoden haben eine konstante Laufzeit, unabhängig von der Anzahl der verwalteten Objekte.

Dokumentation der Klasse `Queue<ContentType>`

`Queue()`

Eine leere Schlange wird erzeugt. Objekte, die in dieser Schlange verwaltet werden, müssen vom Typ `ContentType` sein.

`boolean isEmpty()`

Die Anfrage liefert den Wert `true`, wenn die Schlange keine Objekte enthält, sonst liefert sie den Wert `false`.

`void enqueue(ContentType pContent)`

Das Objekt `pContent` wird an die Schlange angehängt. Falls `pContent` gleich `null` ist, bleibt die Schlange unverändert.

`void dequeue()`

Das erste Objekt wird aus der Schlange entfernt. Falls die Schlange leer ist, wird sie nicht verändert.

`ContentType front()`

Die Anfrage liefert das erste Objekt der Schlange. Die Schlange bleibt unverändert. Falls die Schlange leer ist, wird `null` zurückgegeben.

Name: _____

Die generische Klasse `List`

Objekte der generischen Klasse `List` verwalten beliebig viele, linear angeordnete Objekte vom Typ `ContentType`. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste kann durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.
Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

Dokumentation der Klasse `List<ContentType>`

`List()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

`boolean hasAccess()`

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

`void next()`

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

`void toFirst()`

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

`void toLast()`

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

`ContentType getContent()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben. Andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

Name: _____

`void setContent(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pContent` ungleich `null` ist, wird das aktuelle Objekt durch `pContent` ersetzt. Sonst bleibt die Liste unverändert.

`void append(ContentType pContent)`

Ein neues Objekt `pContent` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`). Falls `pContent` gleich `null` ist, bleibt die Liste unverändert.

`void insert(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt `pContent` vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pContent == null` ist, bleibt die Liste unverändert.

`void concat(List<ContentType> pList)`

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls es sich bei der Liste und `pList` um dasselbe Objekt handelt, `pList == null` oder eine leere Liste ist, bleibt die Liste unverändert.

`void remove()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.