Technische Universität Berlin
Institut für Softwaretechnik und Theoretische Informatik
FG Softwaretechnik
Ernst-Reuter-Platz 7
10587 Berlin

Jähnichen
Mehlhase
Rein-Jury

MPGI 3

Muster-SLK A

Wintersemester 2011/2012
27. Februar 2012

Prüfen Sie zunächst, ob dieses Exemplar vollständig ist (**15** einseitig bedruckte Blätter inklusive Titelblatt).

Tragen Sie auf diesem Titelblatt und darüber hinaus auf allen Blättern, die Sie für Ihre Niederschrift verwenden, Ihren Namen und Ihre Matrikelnummer ein. Zum Bestehen der Prüfungsleistung „Klausur" müssen Sie in den beiden schriftlichen Leistungskontrollen (SLKs) insgesamt mindestens **50 Punkte** erreichen. Darüber hinaus müssen Sie in jeder der beiden schriftlichen Leistungskontrollen mindestens je 10 Punkte erreichen. Insgesamt sind in der vorliegenden schriftlichen Leistungskontrolle 50 Punkte möglich. Die Bearbeitungszeit beträgt 75 Minuten. Bevor die Bearbeitungszeit anfängt, haben Sie 15 Minuten Einlesezeit. In der Klausur sind außer eines beschriebenen DIN-A4-Blattes **keine Hilfsmittel** zugelassen. Verwenden Sie ausschließlich das ausgeteilte Klausur-Papier. Es dürfen nur **permanent-schwarze oder -blaue Stifte** zum Lösen der Aufgaben verwendet werden.

Viel Erfolg!

Name, Vorname: _____

Matrikelnummer: _____

Studienrichtung: _____

MPGI3-Übungen habe ich im _____ besucht.
(z.B. WS 2011/12)

Aufgabe	1	2	3	4	5	6	7	8	Gesamt
Maximal:	3	3	6	5	14	9	4	6	50
Erreicht:									

Aufgabenstellung

Entwickeln Sie das unten beschriebene System nach der in der Veranstaltung vermittelten Methodik. Ergänzen Sie hierzu die geforderten Artefakte in den nachfolgenden Aufgaben.

Tennisclub

Ein Tennisclub möchte seinen Spielern mehr Service bieten und beauftragt Sie mit dem Entwurf einer geeigneten Software, die es den Spielern ermöglicht Tennisstunden zu reservieren und zu stornieren.

Der Tennisclub hat zehn Tennisplätze, die von der einen Sekretärin des Clubs verwaltet werden. Spieler können festgelegte Zeitslots für Tennisstunden reservieren, die von der Sekretärin eingetragen werden. Zeitslots sind je eine Stunde lang und beginnen immer zur vollen Stunde (*anfangszeit*). Zum Eintragen eines Zeitslots gibt die Sekretärin eine Anfangszeit, einen Platz und einen Tennislehrer an. Der neue Zeitslot wird daraufhin mit den angegebenen Daten im System angelegt.

Ein Spieler kann sich über ein Webportal registrieren, woraufhin seine Daten mit einer eindeutigen Identifikationsnummer im System gespeichert werden. Nachdem der Spieler registriert ist, kann er Zeitslots reservieren. Dafür gibt er seine Identifikationsnummer und ein Wunschdatum ein. Hat der Spieler noch fünf ausstehende Zeitslots reserviert (später als heute), so kann er keine weiteren Zeitslots mehr reservieren. Hat er weniger als fünf ausstehende Reservierungen, so überprüft das System, ob irgendein Zeitslot am Wunschdatum frei ist. Ein Zeitslot gilt als frei, solange er noch keinem Spieler zugewiesen ist. Ist dies der Fall wird eine Reservierung vorgenommen, indem der gefundene Zeitslot dem Spieler zugewiesen wird. Der Spieler wird über den Ausgang seines Reservierungsversuchs benachrichtigt. Eine Besonderheit tritt ein, wenn der Spieler gerade erfolgreich seine 100. Reservierung durchführt, da er dann zu einem Premiumspieler wird. Die alten Reservierungen eines Spielers werden nicht aus dem System entfernt!

Spieler können vorhandene Reservierungen ebenfalls über das Webportal stornieren. Dafür gibt der Spieler seine Indentifikationsnummer, das Datum und die Anfangszeit des zu stornierenden Zeitslots ein. Liegt keine Reservierung des Spielers für den entsprechenden Zeitslot vor, so wird er darüber informiert. Ansonsten wird der Zeitslot storniert. Ist der Spieler ein Premiumspieler, wird er über seine erfolgreiche Stornierung informiert. Ist er jedoch kein Premiumspieler, muss er er für die Stornierung eine Säumnisgebühr zahlen, sofern sich der Zeitslot auf das heutige Datum bezieht. Der Spieler und die Sekretärin werden über diesen Fall informiert. Bezieht sich der Zeitslot nicht auf das heutige Datum, kann der Zeitslot ohne Stornogebühren storniert werden und nur der Spieler wird über die erfolgreiche Stornierung informiert.

Hinweis: Zur Vereinfachung können Sie davon ausgehen, dass sich die von Spielern eingegeben Daten immer auf den heutigen Tag oder die Zukunft beziehen.

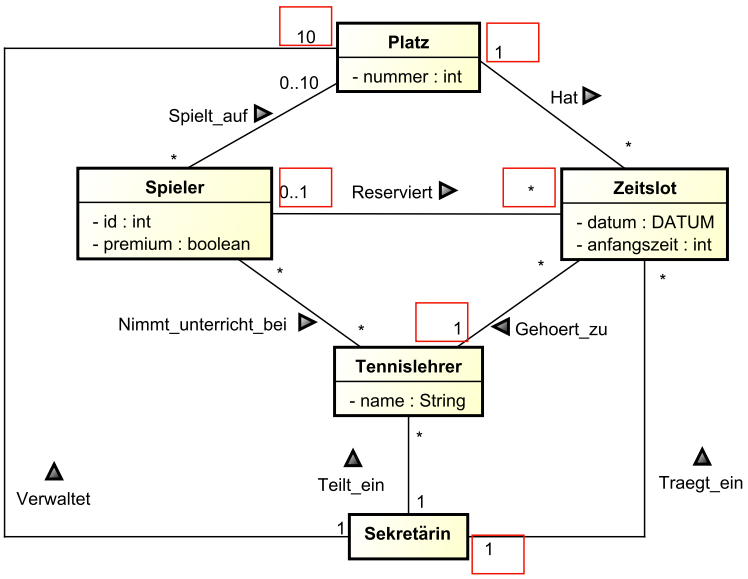
1. Klassenmodell des Gegenstandsbereichs

3 Punkte

Ergänzen Sie an den markierten Stellen die fehlenden Multiplizitäten im vorgegebenen Klassenmodell des Gegenstandsbereichs.

- Es dürfen keine weiteren Assoziationen oder Klassen ergänzt werden.
- Gegebene Multiplizitäten dürfen nicht geändert werden.

Lösung:



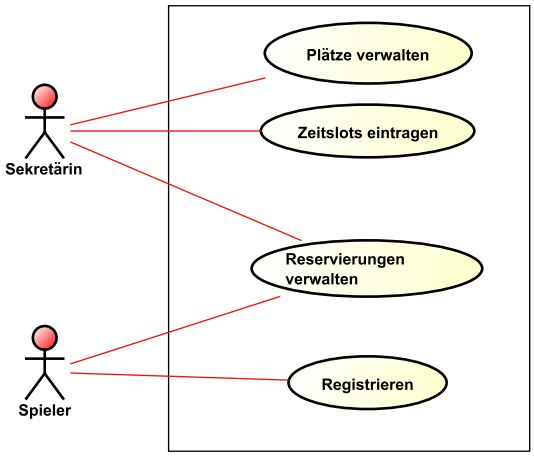
2. Use-Case-Diagramm

3 Punkte

Vervollständigen Sie das vorgegebene Use-Case-Diagramm.

- Bestimmen Sie die Akteure des Systems und ordnen Sie diese den Funktionsgruppen zu.
- Fügen Sie der Vorgabe **keine** neuen Funktionsgruppen hinzu.
- Beziehungen zwischen den Funktionsgruppen **dürfen nicht** hinzugefügt werden.

Beispiellösung:



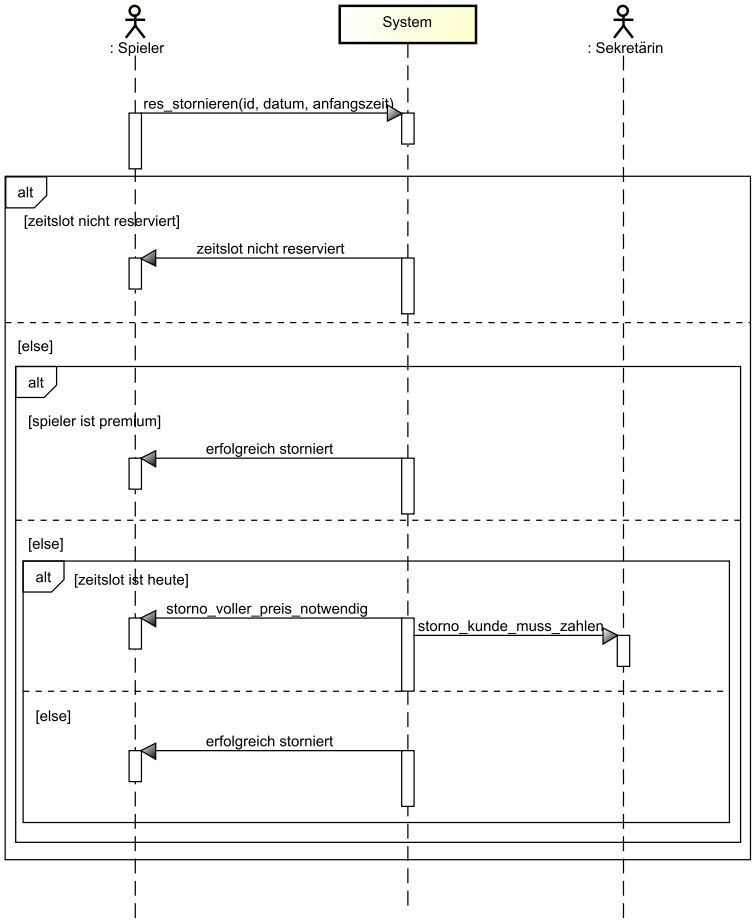
3. Sequenzdiagramm

6 Punkte

Erstellen Sie ein Sequenzdiagramm für das Szenario "Reservierung stornieren" aus dem Use-Case *Reservierungen verwalten*.

- Betrachten Sie nur die Fallunterscheidungen aus der Beschreibung.

Beispiellösung:



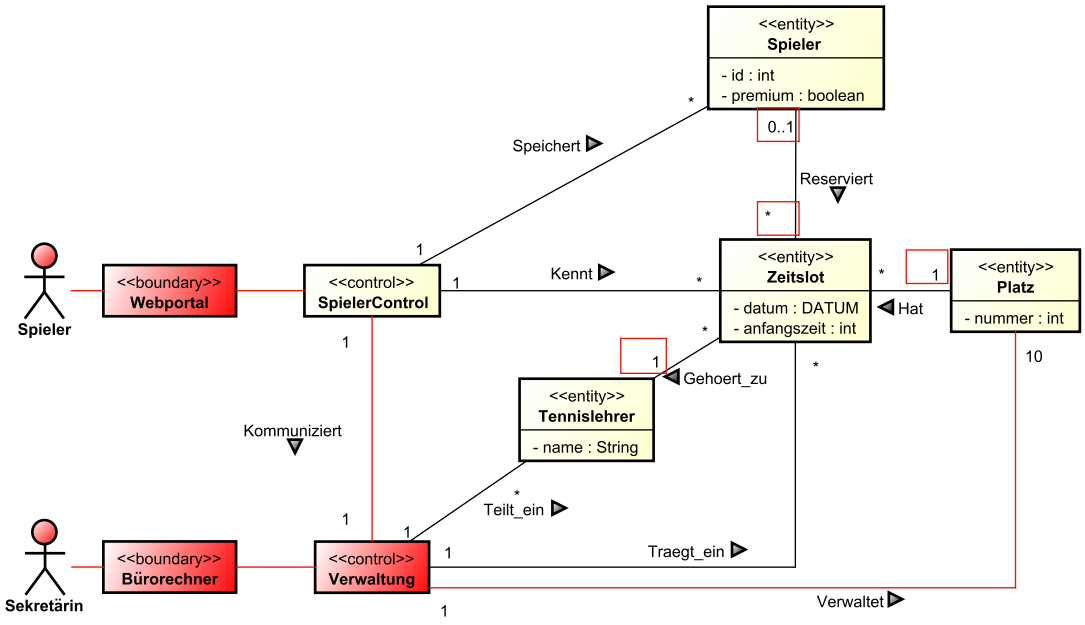
4. Systemklassenmodell / Entwurfsklassenmodell

5 Punkte

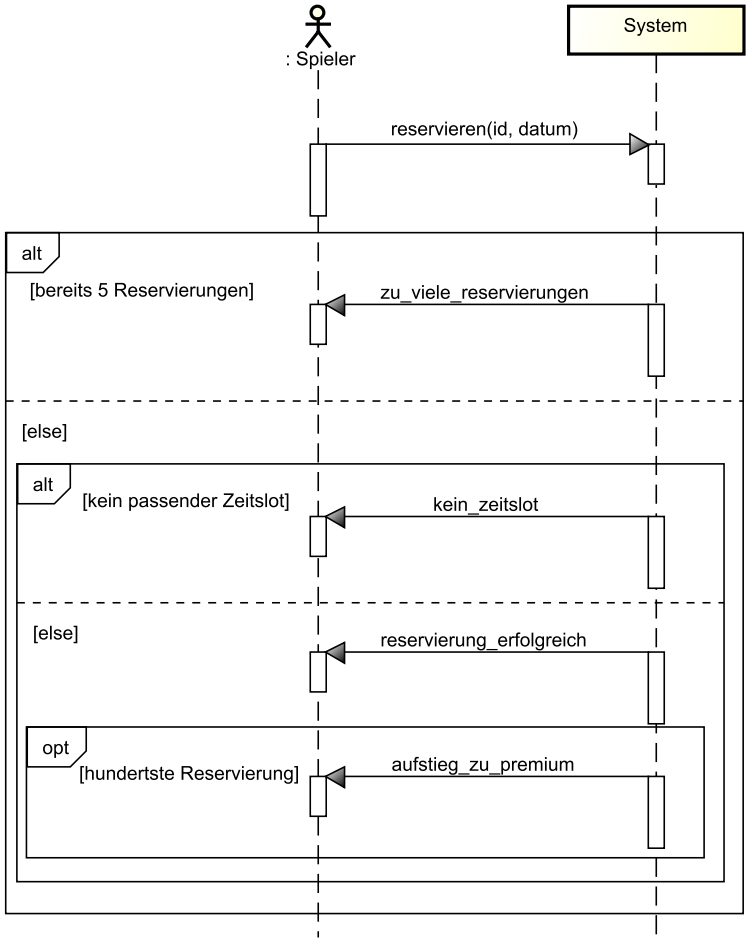
Vervollständigen Sie das vorgegebene Systemklassenmodell.

- Ergänzen Sie benötigte Klassen und Assoziationen mit deren Multiplizitäten.
- Ergänzen Sie die fehlenden Multiplizitäten in den Rechtecken.
- Ergänzen Sie die fehlenden Stereotypen in den Ovalen.
- Gegebene Multiplizitäten dürfen **nicht** geändert werden.

Beispiellösung:



Vorgabe: Sequenzdiagramm *reservieren* aus dem Use-Case *Reservierungen verwalten*



Die Eingabeparameter der Systemoperation *reservieren* haben folgende Bedeutungen:

id	Identifikationsnummer des Spielers
datum	Datum an dem der Zeitslot gebucht werden soll

5. Operationsschema *reservieren*

14 Punkte

Vervollständigen Sie auf Basis Ihres Systemklassenmodells das vorgegebene Operationsschema für die Systemoperation *reservieren* auf der nächsten Seite. Es sollen nur die im Sequenzdiagramm dargestellten Fälle abgedeckt werden.

- Benutzen Sie *no_effect* nur an den notwendigen Stellen.
- Der Spieler kann keine Anfangszeit für seine Reservierung angeben, es wird lediglich das Datum überprüft.
- Sie können Daten beim Vergleichen genauso wie natürliche Zahlen behandeln – d.h. für zwei Daten $d_1 : DATUM$ und $d_2 : DATUM$ bedeutet die Aussage $d_1 < d_2$, dass das Datum d_1 vor dem Datum d_2 liegt. Sie dürfen in dem Operationsschema den Platzhalter $TODAY : DATUM$ verwenden, der den heutigen Tag beschreibt.
- Benutzen Sie die folgenden Attribut- und Typdeklarationen:

Spieler
id: int premium: boolean

Zeitslot
datum: DATUM anfangszeit: int

Op. = reservieren

Desc. = Ein Spieler möchte über das Webportal einen Zeitslot für ein bestimmtes Datum reservieren.
Vorbedingungen: Der Spieler mit der Identifikationsnummer *id* existiert im System und das übergebene Datum ist später als TODAY.

Input = $id : int, datum : DATUM$
Reads = sc: SpielerControl,
Speichert, Zeitslot, Spieler, Kennt
Changes =
$s : Spieler$ **with** $s.id = id \land (sc, s) \in Speichert$,
Reserviert
Sends = Spieler: $\{zu\_viele\_reservierungen, reservierung\_erfolgreich, aufstieg\_zu\_premium, kein\_zeitslot\}$
Pre = $implicit \land datum > TODAY$
Post = **let**

$$
\begin{aligned}
&alle\_slots == \{sl : Zeitslot \mid (s, sl) \in Reserviert\} \\
&offene\_res == \{sl : alle\_slots \mid sl.datum > TODAY\} \\
&freie\_slots == \{sl : Zeitslot \mid (sc, s) \in Kennt \land sl.datum = datum \land \\
&\qquad\qquad \nexists s_2 : Spieler \bullet (s_2, sl) \in Reserviert\} \\
&\bullet \\
&\quad (\#offene\_res \geq 5 \Rightarrow \\
&\qquad is\_sent\{zu\_viele\_reservierungen\} \land no\_effect) \land \\
&\quad (\#offene\_res < 5 \Rightarrow \\
&\qquad (freie\_slots = \varnothing \Rightarrow \\
&\qquad\quad is\_sent\{kein\_zeitslot\} \land no\_effect) \land \\
&\qquad (freie\_slots \neq \varnothing \Rightarrow \\
&\qquad\quad is\_sent\{reservierung\_erfolgreich\} \land \\
&\qquad\quad \land \exists sl : freie\_slots \bullet \\
&\qquad\qquad\qquad Reserviert' = Reserviert \cup \{(s, sl)\}) \land \\
&\qquad\quad (\#alle\_slots = 99 \Rightarrow s.premium' \land s.id' = id \land is\_sent\{aufstieg\_zu\_premium\}) \land \\
&\qquad\quad (\#alle\_slots < 99 \lor \#alle\_slots > 99 \Rightarrow no\_effect) \\
&\qquad)
\end{aligned}
$$

6. Kommunikationsdiagramm *reservieren*

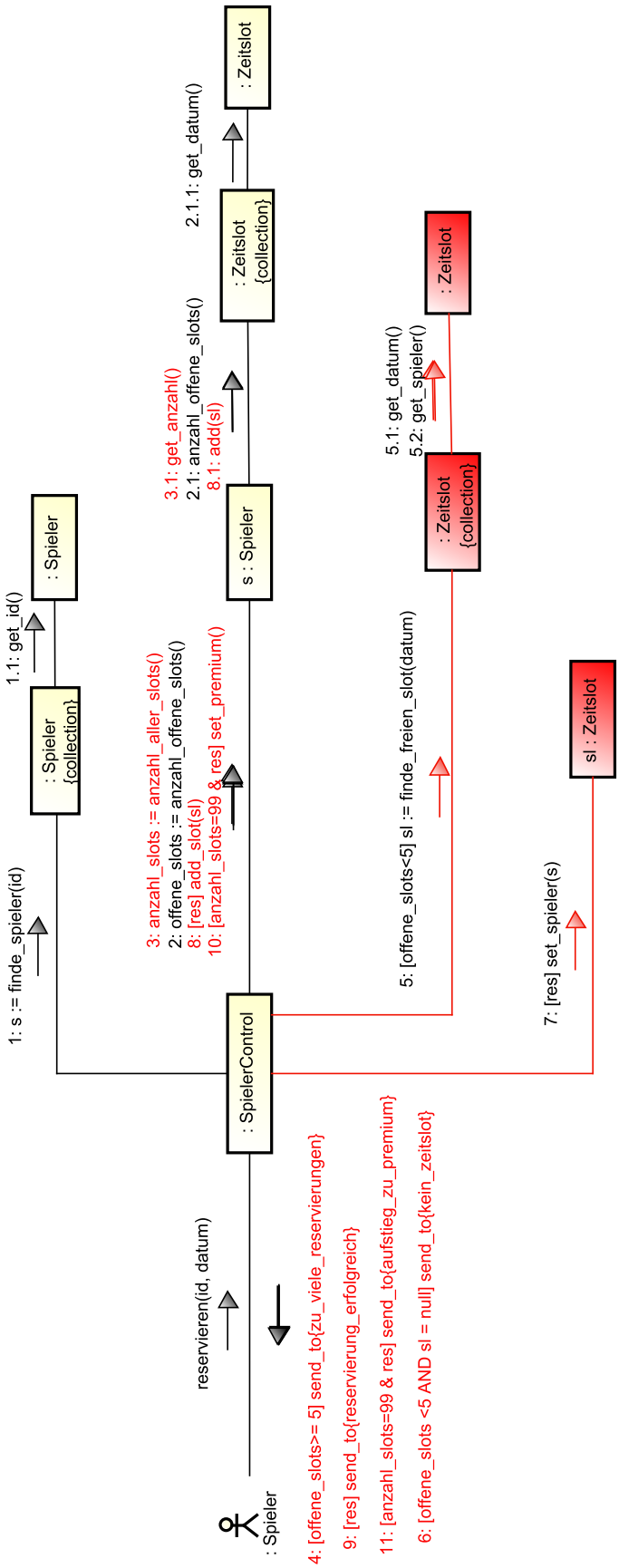
9 Punkte

Vervollständigen Sie auf Basis der bisherigen Modelle das Kommunikationsdiagramm auf der nächsten Seite für die Systemoperation *reservieren*.

Beachten Sie dabei folgende Punkte:

- Ergänzen Sie an den markierten Stellen in der Vorlage die Nummerierungen und die notwendigen Bedingungen.
- Fügen Sie fehlende Objekte, Links, Systemereignisse und Nachrichten hinzu, um die Systemoperation zu vervollständigen.
- Bereits vorgegebene lokale Variablen:
  - ***s***: Der Spieler mit der Identifikationsnummer *"id"*
  - ***offene_slots***: Anzahl der reservierten Zeitslots des Spielers *"s"*, die später als TODAY sind.
- Wenn Sie Abkürzungen für Bedingungen einführen, müssen diese in der folgenden Tabelle eindeutig definiert werden. Textuelle Beschreibungen für Bedingungen sind nicht zulässig!

Abkürzung	Definition
res	$offene\_slots < 5\ AND\ sl\ != null$


: Spieler
: SpielerControl
: Spieler {collection}
: Spieler
s : Spieler
: Zeitslot {collection}
: Zeitslot
: Zeitslot {collection}
: Zeitslot
sl : Zeitslot
reservieren(id, datum)
1: s := finde_spieler(id)
1.1: get_id()
2: offene_slots := anzahl_offene_slots()
2.1: anzahl_offene_slots()
2.1.1: get_datum()
3: anzahl_slots := anzahl_aller_slots()
3.1: get_anzahl()
4: [offene_slots>= 5] send_to{zu_viele_reservierungen}
5: [offene_slots<5] sl := finde_freien_slot(datum)
5.1: get_datum()
5.2: get_spieler()
6: [offene_slots <5 AND sl = null] send_to{kein_zeitslot}
7: [res] set_spieler(s)
8: [res] add_slot(sl)
8.1: add(sl)
9: [res] send_to{reservierung_erfolgreich}
10: [anzahl_slots=99 & res] set_premium()
11: [anzahl_slots=99 & res] send_to{aufstieg_zu_premium}

7. Java-Implementierung

4 Punkte

Implementieren Sie die Methode `anzahlOffeneSlots` der Klasse `Spieler` auf der nächsten Seite, indem Sie die Vorgabe ergänzen. Diese Methode zählt, wie im vorangegangenen Kommunikationsdiagramm dargestellt, die Zeitslots der Collection `zeitslots`, die später als heute sind, und gibt die resultierende Anzahl zurück. Für den Vergleich von `Calendar`-Werten können Sie die Methoden `before` und `after` der Klasse `Calendar` benutzen. Das heutige Datum, mit dem verglichen werden soll, ist bereits in der lokalen Variable `today` gespeichert.

Gehen Sie davon aus, dass die Objekte korrekt initialisiert werden. Sie müssen **keine** Konstruktoren implementieren. Sie müssen **keine** `import`-Anweisungen angeben.

Auszug aus der Java-API

Methoden der Klasse **java.util.Collection**<**E**>

TYP	NAME UND PARAMETER
boolean	add(E e)
boolean	addAll(Collection <? extends E> c)
void	clear()
boolean	contains(Object o)
boolean	containsAll(Collection<?> c)
boolean	equals(Object o)
int	hashCode()
boolean	isEmpty()
Iterator<E>	iterator()
boolean	remove(Object o)
boolean	removeAll(Collection<?> c)
boolean	retainAll(Collection<?> c)
int	size()
Object[]	toArray()
<T>T[]	toArray(T[] a)

```
public class Zeitslot {
        private Calendar datum;
        public Calendar getDatum() {
                return datum;
        }
}

public class Spieler {
        private LinkedList<Zeitslot> zeitslots;
        public int anzahlOffeneSlots(){
                Calendar today = Calendar.getInstance();
                int result = 0;
                for (Zeitslot zs: zeitslots){
                        if (zs.getDatum().after(today))
                                result++;
                }
                return result;
        }
}
```

8. Fragen

6 Punkte

Beantworten Sie die folgenden 6 Multiple Choice Fragen.

- Es ist immer nur eine Antwort korrekt.
- Eine falsche Antwort oder keine Antwort bedeutet einen Punkt Abzug.

Frage 1:

Was ist der Unterschied zwischen einem Objekt und einer Klasse?

x Ein Objekt ist eine Instanz einer Klasse

o Eine Klasse ist eine Instanz eines Objekts

o Eine Klasse und ein Objekt sind das Gleiche

Frage 2:

Was repräsentieren die Klassen im Klassenmodell des Gegenstandsbereichs (KMG)?

o Eine Klasse im KMG beschreibt eine Implementierungsklasse

o Eine Klasse im KMG beschreibt eine abstrakte Datenstruktur

x Eine Klasse im KMG beschreibt reale Gegenstände

Frage 3:

Aus wie vielen Szenarien besteht ein Use-Case?

o Ein Use-Case repräsentiert genau ein Szenario

x Ein Use-Case repräsentiert meist mehrere Szenarien

o Ein Use-Case repräsentiert immer mehrere Szenarien

o Ein Use-Case enthält keine Szenarien

Frage 4:

Wie viele Szenarien werden in einem Aktivitätsdiagramm zusammengefasst?

- o Alle Szenarien des Systems
- o Alle Szenarien eines Use-Cases
- o Alle Szenarien eines Akteurs
- x Alle Szenarien einer Use-Case – Akteur Beziehung

Frage 5:

Welcher Ausdruck ist in der Z-Notation syntaktisch korrekt?

- o $\exists a : \mathbb{N} \mid a > 5 \land a < 7$
- x $\exists a : \mathbb{N} \bullet a > 5 \land a < 7$
- o $\exists a : \mathbb{N} \bullet a > 5 \mid a < 7$
- o $\exists a \in \mathbb{N} \mid a > 5 \bullet a < 7$

Frage 6: Was bedeutet es einen Akteur im Systemklassenmodell zu spiegeln?

- o Der Akteur darf das System benutzen
- o Für den Akteur wird eine Control-Klasse erstellt
- o Für den Akteur wird eine Boundary-Klasse erstellt
- x Für den Akteur wird eine Entity-Klasse erstellt