Spezifikationen besonderer Operationen des Belegsystem 2.0

Auflistung einiger zentraler Operationen für grundlegende Muss-Funktionalitäten des Belegsystems. `Klassenoperationen` sind deutlich gemacht, `Klassen/Klassenatribute` ebenso, nur ohne Unterstrich.

Klasse Participant

Zweck: Ein Participant bzw. Teilnehmer oder Student/in kann Kurse in dem Belegsystem belegen.

Signatur	`bookCourse(`
	`  course: Course`
	`): boolean`

Effect:	Mittels der Methode `bookCourse` belegt (=bucht) ein `Participant` einen konkreten Kurs (`Course`) - soweit dies noch möglich ist.

Funktionsweise	• Es wird überprüft ob der `Course` im aktuellen Term stattfindet (`startDate/endDate` des `Term`). Bereits abgeschlossene Kurse (bzw. Kurse die schon länger laufen als der Belegzeitraum war, falls es einen gibt) können nicht mehr belegt werden, d.h. die Methode wird mit Rückgabewert `false` beendet.
	• Anschließend wird überprüft ob ein `Registration`-Objekt für diesen Kurstyp bereits vorliegt. Falls nicht (=er hat noch keinen Course dieses "Kurstyps" (ermittelbar über `courseIdentifier` bzw. über `CourseTemplate`) belegt), wird ein neues `Registration`-Objekt für diesen Kurstyp erzeugt, mit dem `CourseTemplate` verknüpft und im `Participant` gespeichert.
	• Nun wird mittels der Methode `canSubscribe` in der `Registration` überprüft ob der `Participant` sich überhaupt noch in den Kurs eintragen darf.
	  ◦ Falls ja wird der Kurs den `courses` in der Registration hinzugefügt, der `Participant` der Liste der Teilnehmer des konkreten Kurses hinzugefügt (mittels der Methode `addParticpant` im `Course`) und `true` zurückgegeben.
	  ◦ Falls nein wird der `Course` nicht hinzugefügt und `false` zurückgegeben.

In (Parameter):	`course` - der Kurs, den der `Participant` belegen will (d.h. sich in ihn

	einschreiben will), wie z.B. Softwareprojekt 1 im Sommersemester 2011
Out:	`true` falls Belegung erfolgreich (auch falls Kurs bereits belegt) `false` falls Belegung nicht erfolgreich
Exceptions	InvalidArgumentException: Falls der übergebene Parameter null ist wird diese Exception geworfen.

Signatur	`cancelCourse(`
	`course: Course`
	`override: boolean`
	`): boolean`

Effect:	Mittels der Methode `cancelCourse` kann ein `Participant` einen bereits belegten konkreten Kurs (`Course`) wieder ablegen (=canceln).
Funktionsweise	Im Rahmen der Methode wird überprüft ob erstens der `Course` überhaupt beim `Participant` eingetragen ist, er also diesen Kurs belegt hat und zweitens ob eine Austragung noch zulässig ist (abhängig von `Course.startDate` bzw. immer zulässig wenn `override==true`). Nur dann wird der `Course` aus der zugehörigen `Registration` entfernt (und der `Participant` aus der Liste der `participants` des `Course` mittels der Methode `removeParticipant` entfernt).
In (Parameter):	`course` - der Kurs, den der Participant ablegen will (d.h. aus dem er sich austragen will), wie z.B. Softwareprojekt 1 im Sommersemester 2011
In (Parameter):	`override` - hiermit wird eine mögliche Stornierungsfrist überschrieben, d.h. bei `override == true` kann der Kurs immer noch storniert werden auch wenn die Frist bereits abgelaufen ist.
Out:	`true` falls Austragung erfolgreich `false` falls Austragung nicht erfolgreich
Exceptions	InvalidArgumentException Falls der übergebene Parameter `course` null ist wird diese Exception geworfen.

Registration

In der Klasse `Registration` werden die Belegung(en) für einen abstrakten Kurs (Kurstyp) verwaltet, d.h. eine `Registration` für den Kurstyp Softwareprojekt kann die Belegung mehrerer konkreter Kurse (SP1 im SoSe11, SP1 im WiSe11/12 usw) umfassen.

Signatur	`canSubscribe(): boolean`
Effect:	Mittels der Methode `canSubscribe` wird überprüft, ob ein `Participant` noch einen konkreten Kurs (`Course`) des abstrakten Kurses (Kurstyp bzw. `CourseTemplate`) belegen dürfte oder ob er ihn vielleicht schon abgeschlossen hat oder die maximale Anzahl an Versuchen verbraucht hat.
Funktionsweise	Im Rahmen der Methode wird überprüft ob dieser abstrakte Kurs nicht bereits schon absolviert wurde. Falls `Registration.finished == true` wird false zurückgegeben. Falls nicht `finished` wird überprüft ob die Zahl der Einträge in der courses - Liste der Registration geringer ist als die maximal Anzahl von Versuchen (`Registration.maxTries`). `Registration.maxTries` kann bei der `Registration` individuell gesetzt sein, muss es aber nicht, falls nicht wird der Wert des abstrakten Kurses herangezogen (also `CourseTemplate.maxTries`). Falls weniger Kurse als möglich eingetragen sind kann noch belegt werden, also Rückgabe von `true`, sonst Rückgabe von `false`
In (Parameter):	---
Out:	`true` falls Belegung noch möglich `false` falls Belegung nicht mehr möglich
Exceptions	---

Course

Ein Course ist ein konkreter Kurs der auf einem CourseTemplate basiert. Ein Beispiel wäre der Kurs "Softwareprojekt 1" der im Sommersemester 2011 mit Hr. Steppat als Dozent stattfindet.

Signatur	`addParticipant(`
	`        participant : Participant`
	`): boolean`

Effect:	Mittels der Methode `addParticipant` wird ein Participant der Teilnehmerliste (`Course.participants`) hinzugefügt - soweit dies für diesen `Participant` möglich ist (d.h. dieser `Participant` den Kurs noch nicht abgeschlossen hat und ihn noch einmal belegen darf).
Funktionsweise	Die Funktionsweise ist analog zur Methode `bookCourse` beim `Participant` und wird deshalb hier nicht noch einmal beschrieben.
In (Parameter):	`participant` - der Teilnehmer/Student/`Participant` der in den `Course` eingetragen werden soll, d.h. `Course.participants` hinzugefügt werden soll.
Out:	`true` falls Hinzufügen erfolgreich (auch falls bereits in Liste enthalten) `false` falls Hinzufügen nicht erfolgreich
Exceptions	InvalidArgumentException: Falls einer der übergebenen Parameter null ist wird diese Exception geworfen.

Signatur	`removeParticipant(`
	`        participant : Participant`
	`        override: boolean`
	`) : boolean`

Effect:	Mittels dieser Methode wird ein `Participant` aus einem Kurs entfernt - solange dies noch zulässig ist.
Funktionsweise	Die Funktionsweise ist analog zur Methode `cancelCourse` des `Participants` und wird daher hier nicht nochmal beschrieben.
In (Parameter):	`participant` - der Teilnehmer/Student/`Participant` der entfernt werden soll.
In (Parameter):	`override` - hiermit wird eine mögliche Stornierungsfrist überschrieben, d.h. bei `override == true` kann der Kurs immer noch storniert werden auch wenn die Frist bereits abgelaufen ist.
Out:	`true` falls Austragung erfolgreich `false` falls Austragung nicht erfolgreich
Exceptions	InvalidArgumentException: Falls einer der übergebenen Parameter null ist wird diese Exception geworfen.

CourseTemplate

Das `CourseTemplate` ist die abstrakte Vorlage für einen konkreten Kurs (`Course`), d.h. so etwas wie Softwareprojekt 1 das dem 4. Semester (Level 4) zugeordnet ist.
Synonyme sind: abstrakter Kurs, Kurstyp, Kursvorlage

Signatur	`createCourse(`
	`instructors : List<Instructor>,`
	`term : Term,`
	`timeslots: List<Timeslot>,`
	`studyGroup : studyGroup`
	`) : Course`

Effect:	Mittels dieser Methode wird ein konkreter Kurs (`Course`) auf Grundlage des `CourseTemplate` erstellt.
Funktionsweise	Es wird ein neues `Course`-Objekt erstellt, und mit den übergebenen Parametern befüllt.
In (Parameter):	`instructors` - die Liste der Dozenten (`instructor`) die diesen konkreten `Course` unterrichten werden. Meistens nur einer.
In (Parameter):	`term` - der konkrete Zeitraum zu dem der `Course` stattfinden soll. Im Falle einer Hochschule ein konkretes Semester, z.B. Sommersemester 2011.
	`timeslots` - die Liste der Zeit"slots" an denen der Kurs stattfinden soll (normalerweise ein Termin in der Woche). Ein Zeitslot hat einen Anfangszeitpunkt (z.B. 10:00 Uhr am 01.04.2011), einen absoluten Endzeitpunkt (15.07.2011), eine Länge pro Termin (in Minuten, z.B. 90) und ein Wiederholungsperiode (in Tagen, z.B. sieben für jede Woche). Meistens hat ein `Course` nur einen timeslot.
	`studyGroup` - die `studyGroup` (= Zug an der BHT). So kann ein `Course` einem Zug (exklusiv) zugeordnet werden.
Out:	das erstellte und befüllte neue `Course`-Objekt
Exceptions	Falls einer der übergebenen Parameter außer `studyGroup` null (oder einer der Listen-parameter leer) sind wird diese Exception geworfen.