PR1 VU Programmierung 1	Klausurvorbereitung	

Programmierkurse

Implementieren Sie die Klassen `Programmer` und `Course`: Ein `Programmer`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), `credits` (`int`, nicht negativ; = Guthaben) und `languages` (Liste der Sprachen, die die Person beherrscht). Die Liste der Sprachen enthält Einträge aus der Enumeration `Language` (`Language::JAVA`, `Language:CPP`, `Language::PYTHON`, `Language::LISP` und `Language::PERL`). In der Liste der Sprachen darf jede Sprache höchstens einmal auftreten. Der Einfachheit halber wird davon ausgegangen, dass die Namen der Personen eindeutig sind.

Für die Klasse `Programmer` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor(en) mit 1, 2 oder 3 Parametern: Name, Guthaben und Sprachenliste in dieser Reihenfolge. Guthaben und Sprachenliste sind optional mit den Defaultwerten 100 bzw. leere Liste. Sollte einer der Parameter nicht die Voraussetzungen erfüllen (z. B. Name ist leer, Guthaben ist negativ oder die Sprachenliste enthält eine Sprache doppelt), ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool knows_one_of(const vector<Language>& langs) const`: Retourniert `true`, falls die Person (`this`-Objekt) zumindest eine der Sprachen in der Liste `langs` beherrscht, `false` sonst.
- `bool eligible(int cost, Language lang) const`: Retourniert `true`, wenn die Person (`this`-Objekt) ein Guthaben von mindestens `cost` hat und die Sprache `lang` **nicht** beherrscht, `false` sonst.
- `void finish(int cost, Language lang)`: Wenn die Person (`this`-Objekt) ein Guthaben hat, das kleiner als `cost` ist, oder die Sprache `lang` bereits beherrscht, ist eine Exception vom Typ `runtime_error` zu werfen. Andernfalls ist das Guthaben um `cost` zu vermindern und `lang` ist am Ende der Liste der beherrschten Sprachen hinzuzufügen.
- `operator<<`: `Programmer`-Objekte müssen in der Form [*name*: *credits*, {*Liste der beherrschten Sprachen*}] ausgegeben werden, z. B. [**Susan: 261, {C++, Java}**]. Der vordefinierte Vektor `language_names` kann für die Ausgabe der Enumerationswerte verwendet werden.

Ein `Course`-Objekt hat die Instanzvariablen `language` (Wert aus der Enumeration `Language`, die unterrichtete Sprache), `cost` (`int`, nicht negativ) und `participants` (nicht leere Liste der Personen, die den Kurs belegen). Für die Klasse `Course` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor mit drei Parametern Sprache, Kosten und Teilnehmerliste. Sollte ein Parameter die Voraussetzungen nicht erfüllen (z. B. Kosten kleiner Null oder Teilnehmerliste leer), ist eine Exception vom Typ `runtime_error` zu werfen. (Sie dürfen davon ausgehen, dass in der Teilnehmerliste keine Personen mehrfach auftreten, das muss nicht von Ihrem Programm überprüft werden.) Eine Exception ist außerdem zu werfen, wenn die Teilnehmerliste eine Person enthält, die nicht berechtigt ist, den Kurs zu belegen. Das ist der Fall, wenn die Person nicht genügend Guthaben besitzt, um die Kosten des Kurses abzudecken oder die Sprache, die unterrichtet wird, bereits beherrscht (Vgl. `Programmer::eligible()`).
- `void finish()`: Bei allen Teilnehmern sind die Kosten des Kurses vom Guthaben abzuziehen und die unterrichtete Sprache am Ende der Liste der beherrschten Sprachen einzutragen (Vgl. `Programmer::finish()`).
- `operator<<`: Die Ausgabe eines Objekts vom Typ `Course` muss in der Form [*language*: *cost*, {*Teilnehmerliste*}] erfolgen, z. B.: [**Python: 30, {[Susan: 261, {C++, Java}], [Richard: 50, {}]}**].
- Zusatz für 10 Punkte: Erweitern Sie die Klasse `Programmer` um eine Methode
  `bool revoke(const vector<Language>& langs)`: Diese entfernt alle Sprachen der Liste `langs` aus der Liste der Sprachen, die die Person (`this`-Objekt) beherrscht. Die relative Reihenfolge der Einträge in der Liste der beherrschten Sprache ist beizubehalten. Zu retournieren ist `true`, wenn mindestens eine Sprache entfernt wurde, `false` sonst
- Zusatz für 15 Punkte: Erweitern Sie die Klasse `Course` um die Methode
  `vector<int> statistic() const`: Diese retourniert für jede Programmiersprache, von wie vielen Teilnehmern des Kurses sie beherrscht wird. Die Einträge im retournierten Vektor sind den Programmiersprachen in der Reihenfolge zuzuordnen, wie sie für die Enumerationswerte in `Language` definiert ist. Der erste Eintrag ist also die Anzahl der Teilnehmer, die Java beherrschen, der zweite Eintrag die Anzahl der Teilnehmer, die C++ beherrschen, etc.

Implementieren Sie die Klassen `Programmer` und `Course` mit den notwendigen Konstruktoren, Methoden und Operatoren, sodass jedenfalls das Rahmenprogramm kompiliert und ausgeführt werden kann und die gewünschten Ergebnisse liefert. Achten Sie in Ihren Konstruktoren darauf, dass nur gültige Objekte erstellt werden können. Werfen Sie gegebenenfalls eine Exception vom Typ `runtime_error`.

Für Ihr Programm dürfen Sie **nur** die im vorgegebenen Rahmenprogramm angeführten include-Dateien verwenden!

Instanzvariablen sind `private` zu definieren und die Verwendung globaler Variablen ist (abgesehen von im Rahmenprogramm eventuell bereits definierten) nicht erlaubt! Die Datenkapselung darf nicht durchbrochen werden. Es ist daher unter anderem nicht erlaubt, Referenzen oder Pointer auf private Instanzvariablen einer Klasse nach außen zu vermitteln, `friend`-Deklarationen (mit Ausnahme bei Operatorfunktionen) zu verwenden, oder setter-Methoden zu implementieren, die die Integrität der Daten nicht gewährleisten. Interpretationsspielraum in der Angabe können Sie zu Ihren Gunsten nutzen.

Die Teilaufgaben, bei denen keine Punkteanzahl angegeben ist, gelten als Basisfunktionalität. Für eine positive Beurteilung ist zumindest die Basisfunktionalität zu implementieren. Diese wird mit 30 Punkten bewertet. Die übrigen Teilaufgaben müssen nicht unbedingt implementiert werden, führen aber im Falle einer korrekten Implementierung zu einer entsprechenden Erhöhung der Punkteanzahl.