PR1 VU Programmierung 1	Abschlussklausur	19.01.2021

Sportverein

Implementieren Sie die Klassen `Member` und `Association`:

Ein `Member`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), und eine nicht leere Liste von ausgeübten Sportarten (`vector<Sport>`). Eine Sportart ist ein Wert aus der vordefinierten Enumeration `Sport(Sport::Volleyball, Sport::Dancing, Sport::Tennis, Sport::Chess, Sport::Soccer, Sport::Basketball, Sport::Swimming)`. Für die Klasse `Member` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor mit 2 Parametern: Name und Liste von Sportarten in dieser Reihenfolge. Sollte einer der Parameter leer sein oder die Liste von Sportarten Einträge enthalten, die mehrfach auftreten, so ist eine Exception vom Typ `runtime_error` zu werfen.
- `bool practices_sport(Sport s) const`: Retourniert `true`, falls das `this`-Objekt die Sportart `s` ausübt (die Sportart also in der Liste der ausgeübten Sportarten enthalten ist), sonst `false`.
- `operator<<`: Ein `Member`-Objekt muss in der Form [*name*, {*Liste der Sportarten*}] ausgegeben werden, z.B.: [Anna, {Volleyball, Dancing}]. Der vordefinierte Vektor `sport_names` kann für die Ausgabe der Sportarten verwendet werden.

Ein `Association`-Objekt hat die Instanzvariablen `name` (`string`, nicht leer), `offered_sports` (`vector<Sport>`, nicht leer) und `members` (`vector<Member>`, nicht leer). Für die Klasse `Association` sind folgende Methoden und Funktionen zu implementieren:

- Konstruktor mit 3 Parametern: Name, Liste von angebotenen Sportarten und Liste von Mitgliedern (`Member`-Objekten) in dieser Reihenfolge. Sollte einer der Parameter leer sein, ist eine Exception vom Typ `runtime_error` zu werfen. Befindet sich in der Liste von Mitgliedern eines, das keine der vom Verein angebotenen Sportarten ausübt, so ist ebenfalls eine Exception vom Typ `runtime_error` zu werfen.
- `int add_sports(const vector<Sport>& slist)`: Fügt die Sportarten aus `slist`, die noch nicht angeboten werden, unter Beibehaltung der Reihenfolge am Ende der Liste der angebotenen Sportarten hinzu. Sportarten in `slist`, die schon im Angebot sind, werden ignoriert. Sie dürfen davon ausgehen, dass `slist` keine Einträge doppelt enthält. Zu retournieren ist die Anzahl der erfolgreich zum Angebot hinzugefügten Sportarten.
- `int calculate_revenue() const`: Berechnet die veranschlagten Einnahmen des Vereins. Jedes Mitglied bezahlt einen Grundbetrag von 100 sowie 5 für jede ausgeübte Sportart, die vom Verein auch angeboten wird.
- `operator<<`: Die Ausgabe eines Objekts vom Typ `Association` muss in der Form [*name*, {*Liste angebotener Sportarten*}, *Mitgliederanzahl* Mitglied(er)] erfolgen, z.B.: [Sporteln Wien, {Dancing, Volleyball}, 2 Mitglied(er)].
- Zusatz für 10 Punkte: Erweitern Sie die Klasse `Association` um die Methode `vector<Member> add_members(const vector<Member>& mlist)`: Diese fügt die Objekte aus `mlist`, die zumindest eine der angebotenen Sportarten ausüben, unter Beibehaltung der Reihenfolge am Ende der Mitgliederliste ein. Retourniert wird die Liste der Objekte aus `mlist`, die nicht eingefügt wurden, in der ursprünglichen relativen Reihenfolge.
- Zusatz für 15 Punkte: Erweitern Sie die Klasse `Association` um die Methode `vector<Member> stop(Sport s)`: Diese soll die Sportart `s` aus dem Angebot entfernen. Wird `s` gar nicht angeboten oder ist es die einzige angebotene Sportart, so ist eine Exception vom Typ `runtime_error` zu werfen. Sonst wird die Sportart aus der Liste der angebotenen Sportarten entfernt und alle Mitglieder, die jetzt nicht mehr zumindest eine der im Angebot verbliebenen Sportarten ausüben, sind aus der Mitgliederliste zu entfernen. Die relative Reihenfolge in der Mitgliederliste ist dabei beizubehalten. Zu retournieren ist die Liste der entfernten Mitglieder in der ursprünglichen relativen Reihenfolge.

Implementieren Sie die Klassen `Member` und `Association` 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.