Programmierung 1 (PR1)	Abschlussprüfung		Name/Mnr:	1

Bücherei

Implementieren Sie die Klassen **`Werk`** und **`Exemplar`** zur Repräsentation des Bestands in einer Bücherei.

Ein **`Werk`** hat einen Autor (String mit Länge>0), einen Namen (String mit Länge>0), ein Erscheinungsjahr (ganze Zahl >=1700 und <=2017) sowie eine Liste der Exemplare im Bestand. Ein **`Exemplar`** hat die Auflagennummer (ganze Zahl >=1 und <=1000), einen Zustand (ein Wert aus der vordefinierten Enumeration **`Zustand Zustand::Neuwertig`**, **`Zustand::Gut`**, **`Zustand::Abgegriffen`** und **`Zustand::Unbrauchbar`**) und kann verliehen sein oder nicht.

Folgende Methoden und Operatoren sind für die Klasse **`Exemplar`** zu implementieren:

- Konstruktor(en) mit zwei Parametern und einem Parameter. Auflagennummer und Zustand in dieser Reihenfolge. Der Zustand ist optional mit Defaultwert **`Zustand::Neuwertig`**. Sollte einer der übergebenen Werte nicht den Anforderungen entsprechen (z.B. Auflagennummer nicht im erlaubten Bereich), so ist eine Exception vom Typ **`runtime_error`** zu werfen. Neu erstellte Exemplare sind nicht verliehen.
- **`bool verfuegbar() const`**: Liefert **`true`**, wenn das Exemplar verfügbar, d.h nicht verliehen und nicht unbrauchbar ist. Exemplare im Zustand unbrauchbar werden nicht verliehen.
- **`bool entleihen()`**: Falls das Exemplar verfügbar ist, wird es verliehen und **`true`** wird retourniert. Andernfalls wird **`false`** retourniert.
- **`void retournieren(Zustand z)`**: Das Exemplar wird im neuen Zustand **`z`** retourniert. Der Zustand wird entsprechend upgedatet und das Exemplar wird wieder als nicht verliehen gekennzeichnet. Das ist aber nur möglich, falls das Exemplar zum Zeitpunkt des Methodenaufrufs verliehen ist und der neue Zustand **`z`** nicht besser als der ursprüngliche Zustand des Exemplars ist. Andernfalls ist das Exemplar unverändert zu lassen und eine Exception vom Typ **`runtime_error`** zu werfen.
- **`operator<<`**: Die Ausgabe eines Objekts des Typs **`Exemplar`** muss in der Form [Auflage: *Auflagennummer*, Zustand: *Zustand,* verliehen] bzw.[Auflage: *Auflagennummer*, Zustand: *Zustand*] erfolgen, je nachdem ob das Exemplar verliehen ist oder nicht. , z.B.: [Auflage: 2, Zustand: gut] (nicht verliehen) bzw. [Auflage: 1, Zustand: neuwertig, verliehen] (verliehen). Der vordefinierte Vektor **`zustand_namen`** kann für die Ausgabe der Enumerationswerte verwendet werden.

Folgende Methoden und Operatoren sind für die Klasse **`Werk`** zu implementieren:

- Konstruktor mit drei Parametern. Autor, Name und Erscheinungsjahr in dieser Reihenfolge. Es gibt keine Defaultwerte. Wird ein Wert, der nicht den Anforderungen entspricht, übergeben, so ist eine Exception vom Typ **`runtime_error`** zu werfen. Die Liste der Exemplare ist für neu erstellte Werk-Objekte leer.
- **`void erwerben(int nr, Zustand z)`**: Erstellt ein neues Exemplar mit der Auflagennummer nr und dem Zustand z und fügt dieses am Ende der Liste der Exemplare des Werks hinzu. Die relative Reihenfolge der schon vorhandenen Exemplare muss unverändert bleiben. Es darf nicht möglich sein, Exemplare im Zustand unbrauchbar zu erwerben. Gegebenenfalls ist eine Exception vom Typ **`runtime_error`** zu werfen.
- **`bool entleihen()`**: Das erste verfügbare Exemplar in der Liste der Exemplare wird verliehen. Ist kein Exemplar verfügbar, so ist **`false`** zu retournieren sonst **`true`**.
- **`void retournieren(int index, Zustand z)`**: Das Exemplar mit dem entsprechenden Index in der Liste der Exemplare soll retourniert werden. Ist dies nicht möglich (ungültiger Index, Exemplar ist nicht verliehen bzw. Zustand des Exemplars müsste verbessert werden), so ist eine Exception vom Typ **`runtime_error`** zu werfen.
- **`operator<<`**: Die Ausgabe eines Objekts des Typs **`Werk`** soll in der Form [*Autor, Name, Erscheinungsjahr* {*Liste der Exemplare*}] erfolgen, z.B.: [Adams, Hitchhiker, 1979 {[Auflage: 2, Zustand: gut], [Auflage: 1, Zustand: neuwertig, verliehen]}]
- Zusatz für 10 Punkte: **`static vector<Werk> erstausgaben(const vector<Werk>& bestand)`**: Liefert eine Liste aller Werke aus dem Bestand, für die zumindest ein Exemplar der Erstausgabe (mit Auflagennummer 1) erfasst ist. Die relative Reihenfolge der Werke im retournierten Ergebnis muss dieselbe sein wie im übergebenen Parameter **`bestand`**.
- Zusatz für 15 Punkte: **`static void aussortieren(vector<Werk>& bestand)`**: Alle unbrauchbaren Exemplare sind aus dem Bestand zu entfernen. Die relative Reihenfolge der Einträge in den Exemplarlisten bzw. im Bestand muss beibehalten werden.

Implementieren Sie die Klassen **`Exemplar`** und **`Werk`** 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.