Waschmaschine

Implementieren Sie die Klassen **`KleidSt`** und **`WaschM`** zur Repräsentation von Kleidungsstücken und Waschmaschinen.

Ein Kleidungsstück hat eine Bezeichnung (String mit Länge>0), ein Gewicht (in Gramm; ganze Zahl >0 und <5000) und einen Pflegehinweis. Der Pflegehinweis ist ein Wert aus der vordefinierten Enumeration **`Programm`** (**`Programm::Koch`**, **`Programm::Normal`**, **`Programm::Bunt`** und **`Programm::Fein`**). Folgende Methoden und Operatoren sind für die Klasse **`KleidSt`** zu implementieren:

- Konstruktor(en) mit drei Parametern und zwei Parametern. Bezeichnung, Gewicht, und Pflegehinweis in dieser Reihenfolge. Der Pflegehinweis ist optional mit Defaultwert **`Programm::Normal`**. Sollte einer der übergebenen Werte nicht den Anforderungen entsprechen (z.B. Bezeichnung mit Länge 0 bzw. Gewicht nicht im erlaubten Bereich), so ist eine Exception vom Typ **`runtime_error`** zu werfen.
- **`bool vertraeglich(Programm)const`**: Liefert **`true`**, wenn das Kleidungsstück gefahrlos mit dem als Parameter übergebenen Programm gewaschen werden kann und **`false`** sonst. Dabei ist davon auszugehen, dass ein Kleidungsstück mit allen Programmen gewaschen werden kann, die mindestens so schonend sind, wie das im Pflegehinweis angeführte Programm. Die Programme sind in der Definition der Enumeration in der Reihenfolge von am wenigsten schonend bis zu am meisten schonend sortiert. So kann z.B. ein Kleidungsstück mit dem Pflegehinweis **`Programm::Koch`** mit allen Programmen gewaschen werden, eines mit dem Pflegehinweis **`Programm::Fein`** aber nur mit dem Programm **`Programm::Fein`**.
- **`operator<<`**: Die Ausgabe eines Objekts des Typs **`KleidSt`** muss in der Form [*Bezeichnung*, *Gewicht* g, *Pflegehinweis*] erfolgen. , z.B.: [Bluse, 120 g, Kochwaesche]. (Der vordefinierte Vektor **`programm_namen`** kann für die Ausgabe der Enumerationswerte verwendet werden.)

Für eine Waschmaschine sind nur das maximale Ladungsgewicht (in Gramm; ganze Zahl>=4000 und <=50000) und die Liste der geladenen Kleidungsstücke relevant. Folgende Methoden und Operatoren sind für die Klasse **`WaschM`** zu implementieren:

- Konstruktor mit einem Parameter, der das maximale Ladungsgewicht festlegt. Es gibt keinen Defaultwert. Wird ein Wert außerhalb des erlaubten Bereichs übergeben, so ist eine Exception vom Typ **`runtime_error`** zu werfen. Die Liste der geladenen Kleidungsstücke ist für neu erstellte Waschmaschinen-Objekte leer.
- **`void zuladen(const vector<KleidSt>& z)`**: Fügt alle Kleidungsstücke aus **`z`** zur Ladung der Waschmaschine hinzu, indem sie am Ende der Liste der geladenen Kleidungsstücke eingefügt werden. Die relative Reihenfolge der Kleidungsstücke muss gleich bleiben.
- **`void waschen(Programm)`**: Ist die Waschmaschine überladen, oder enthält ein Kleidungsstück, das mit dem als Parameter übergebenen Programm nicht gefahrlos gewaschen werden kann, so ist eine Exception vom Typ **`runtime_error`** zu werfen. Andernfalls wird die Ladung aus der Waschmaschine (nach einem erfolgreichen Waschen) entnommen. Das heißt, die Liste der geladenen Kleidungsstücke wird auf leer zurückgesetzt.
- **`operator<<:`** Die Ausgabe eines Objekts des Typs **`WaschM`** soll in der Form [*maximales Ladungsgewicht* g {*Liste der Kleidungsstücke*}] erfolgen, z.B.:
  [5000 g {[Bluse, 120 g, Kochwaesche], [Bluse, 100 g, Feinwaesche]}]
- Zusatz für 10 Punkte: **`int programme() const`**: Liefert die Anzahl, wie viele der vier möglichen Pflegehinweise, auf den in der Waschmaschine geladenen Kleidungsstücken vorkommen.
- Zusatz für 15 Punkte: **`vector<KleidSt> aussortieren(Programm)`**: Entfernt alle Kleidungsstücke, die mit dem als Parameter spezifiziertem Programm nicht gefahrlos gewaschen werden können, aus der Waschmaschine und retourniert eine Liste aller entfernten Kleidungsstücke (in derselben relativen Reihenfolge wie ursprünglich in der Ladung der Waschmaschine).

Implementieren Sie die Klassen **`KleidSt`** und **`WaschM`** 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.