PR1 VU Programmierung 1	Abschlussklausur	29.01.2019

Kleiderschrank

Implementieren Sie die Klassen `Garment` (Kleidungsstück) und `Closet`:

Ein `Garment`-Objekt hat einen Preis (`int` in Cent, größer gleich Null), eine Farbe und einen Typ. Die Farbe und der Typ sind Werte aus den vordefinierten Enumerationen `Color(Color::Red, Color::Blue, Color::Gray, Color::Yellow)` bzw. `Type(Type::Pants, Type::Blouse, Type::Shirt, Type::Skirt)`. Für die Klasse `Garment` sind folgende Methoden und Operatoren zu implementieren:

- Konstruktor(en) mit 1, 2 bzw. 3 Parametern. Preis, Farbe und Typ in dieser Reihenfolge. Die Farbe ist per Default `Color::Gray`, der Typ per Default `Type::Pants`. Bei negativem Preis ist eine Exception vom Typ `runtime_error` zu werfen.
- `int get_price() const`: Retourniert den Preis eines `Garment`-Objekts.
- `Type get_type() const`: Retourniert den Typ eines `Garment`-Objekts.
- `bool has_color(Color f) const`: Retourniert `true`, falls die Farbe des `Garment`-Objekts `f` ist, sonst `false`.
- `void deteriorate(int w)`: Verringert den Preis des Kleidungsstücks um den als Parameter `w` übergebenen Wert. Sollte der Preis dabei negativ werden, ist er auf Null zu setzen. Sollte der übergebene Parameter negativ sein, ist eine Exception vom Typ `runtime_error` zu werfen.
- `operator<<`: Ein `Garment`-Objekt muss in der Form [*Preis* Cent, *Farbe Typ*] ausgegeben werden. Die vordefinierten Vektoren `color_names` und `type_names` können für die Ausgabe der Enumerationswerte verwendet werden,
  z.B.: [40000 Cent, yellow blouse].

Ein Schrank (`Closet`-Objekt) hat eine maximale Kapazität an Kleidungsstücken (`size_t`, mindestens 5 und höchstens 300) und eine Liste von Kleidungsstücken (`vector<Garment>`). Für die Klasse `Closet` sind folgende Methoden und Operatoren zu implementieren:

- Konstruktor mit 2 Parametern. Maximale Kapazität an Kleidungsstücken und Vektor von `Garment`-Objekten in dieser Reihenfolge. Die als Parameter erhaltene Liste von Kleidungsstücken ist unter Beibehaltung der Reihenfolge in die Liste der Kleidungsstücke des Schranks zu übernehmen. Liegt die maximale Kapazität nicht im erlaubten Bereich, oder übersteigt die Anzahl der Kleidungsstücke die Kapazität des Schranks, so ist eine Exception vom Typ `runtime_error` zu werfen. Es ist außerdem nicht erlaubt, dass der Schrank leer ist oder dass alle Kleidungsstücke in einem Schrank denselben Typ haben. Sollte dies der Fall sein, so ist ebenfalls eine Exception vom Typ `runtime_error` zu werfen.
- `bool add(vector<Garment>)`: Fügt die als Parameter übergebene Liste am **Anfang** der aktuellen Kleidungsliste des Schrankes ein, sofern dadurch die maximale Kapazität des Schrankes nicht überschritten wird. Sollte die maximale Kapazität des Schrankes durch das Hinzufügen überschritten werden, ist die Kleidungsliste des Schrankes im urpsrünglichen Zustand zu belassen und `false` zu retournieren. Sollte das Hinzufügen möglich sein, werden die Kleidungsstücke hinzugefügt und es ist `true` zu retournieren. Die relative Reihenfolge der Kleidungsstücke in den Teillisten (eingefügter Teil und ursprünglicher Teil) muss beibehalten werden.
- `double mean_price() const`: Retourniert den durchschnittlichen Preis über alle Kleidungsstücke im Schrank in Cent.
- `operator<<`: `Closet`-Objekte werden in der Form [*maximale Kapazität*, {*Kleidungsliste*}, *Durchschnittspreis*] ausgegeben, z.B.: [200, {[40000 Cent, yellow blouse], [30000 Cent, gray pants]}, 35000 Cent].
- Zusatz für 10 Punkte: Erweitern Sie die Klasse `Closet` um folgende Methode:
  `vector<int> content() const`: Ermittelt, wie viele Kleidungsstücke der einzelnen Typen jeweils im Schrank vorhanden sind. Die Einträge im retournierten Vektor sind dabei in der Reihenfolge zu interpretieren, die durch die vordefinierte Enumeration `Type` definiert ist. Der erste Vektoreintrag entspricht also der Anzahl der Hosen, der zweite der Anzahl der Blusen etc.
- Zusatz für 15 Punkte: Erweitern Sie die Klasse `Closet` um folgende Methode:
  `vector<Garment> remove(Color f)`: Entfernt alle `Garment`-Objekte aus der Kleidungsliste des Schranks, welche die Farbe `f` haben. Retourniert werden soll eine Liste an Kleidungsstücken, die entfernt wurden in der relativen Reihenfolge, in der sie ursprünglich in der Kleidungsliste des Schranks auftraten. Die relative Reihenfolge der Kleidungsstücke im Schrank muss beibehalten werden. Würde das Entfernen der Kleidungsstücke zu einem Schrank führen, in dem nur mehr ein Typ von Kleidungsstücken enthalten ist, so ist der Inhalt des Schranks unverändert zu lassen und eine leere Liste zu retournieren.

Implementieren Sie die Klassen `Garment` und `Closet` 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.