2. Übung

Abgabe: 02.07.2024, 06 Uhr im Moodle als zip-Datei
Abgabeverzeichnis: ueb02

Lernziele

- Entwickeln einer Klasse in C++
- Arbeiten mit der vector-Klasse
- Arbeiten mit Pointern auf Objekte
- Dialog mit C++
- Dynamische Speicherplatzreservierung und Freigabe
- Kopierkonstruktor und Zuweisungsoperator implementieren

1. Aufgabe - Klasse Article

Die LKWs in SimTransport müssen Waren transportieren. Entwerfen Sie eine Klasse – erst einmal unabhängig von SimTransport – die einen einfachen Artikel repräsentiert.

Attribute

- Artikelnummer
- Beschreibung
- Aktueller Artikelbestand in Stück
- Preis

Methoden

Entwickeln Sie mindestens die folgenden Methoden:

- increaseStock(int): Buchen eines Zugangs, d.h. der Bestand wird um eine bestimmte Menge erhöht
- reduceStock(int): Buchen eines Abgangs, d.h. der Bestand um eine bestimmte Menge vermindert
- toString(): bereitet ein Artikel-Objekt als eine Zeichenkette auf.
- changePrice(double): Andert den Preis um eine bestimmte Prozentzahl. Dabei kann es sich um eine Preiserhöhung oder eine Preisverringerung handeln.
- get-Methoden zu allen Attributen
- set-Methoden nur zu den Attributen, für die eine set-Methode sinnvoll ist.

Konstruktoren

Entwickeln Sie die folgenden Konstruktoren:

- Article(int articleNumber, std::string& description, double price, int stock)
- Article(int articleNumber, std::string& description, double price)

Prüfen Sie außerdem in der Klasse *Article* alle Übergabeparameter auf ihre logische Gültigkeit, d.h. wenn sich die Daten eines Artikels durch fehlerhafte Übergabeparameter ungültig ändern würden, erzeugen Sie und werfen Sie eine Ausnahme. Sie müssen natürlich diese Überprüfungen in allen Methoden vornehmen, die die Daten des Artikels verändern.

2. Aufgabe - Klasse Storehouse

Erstellen Sie eine Klasse Storehouse zum Verwalten von mehreren Artikeln. Mindestens die folgenden Attribute werden benötigt:

- Ein Container mit Artikel (Vektor oder Array mit Pointern auf Artikel-Objekte)
- Eine maximale Artikelanzahl, die nicht überschritten werden darf.

Das Lager benötigt mindestens die folgenden Konstruktoren:

- Storehouse(int size): Lager mit der angegebenen Maximalgröße wird angelegt.
- Storehouse(): Lager mit einer vorgegebenen Größe (z.B. 10) wird angelegt.

Sie benötigen außerdem mindestens diese Methoden (bitte beachten Sie die Namen und die Parameter der Methoden):

- void addArticle(Article* article): Ein neuer Artikel wird in das Lager gelegt. Eine Sortierung ist nicht erforderlich.
- removeArticle(int articleNumber): Entfernen eines Artikels aus dem Lager. Schließen Sie die Lücken durch Nachrücken, d.h. der zu löschende Artikel wird gelöscht, indem die nachfolgenden Artikel jeweils um eine Stelle nach vorne gerückt werden.
- void increaseStock(int articleNumber, int amount): Zugang buchen für einen Artikel.
- void reduceStock(int articleNumber, int amount): Abgang buchen für einen Artikel.
- void changePrice(int articleNumber, int percent): Den Preis eines Artikels um einen bestimmten Prozentsatz verändern (positiv oder negativ möglich).
- void changePrice(int percent): Preis für alle Artikel um einen bestimmten Prozentsatz verändern (positiv oder negativ möglich).
- double calculateTotalValue(): Gesamtwert der Artikel im Lager berechnen. Beachten Sie dabei auch den Bestand der Artikel.
- Article* findArticle(int articleNumber): Gibt einen Pointer auf einen Artikel mit einer bestimmten Artikelnummer zurück. Wenn der Artikel nicht im Lager enthalten ist, wird NULL zurückgegeben.
- bool istFull(): Gibt true zurück, wenn das Lager voll ist, sonst false.
- bool isEmpty(): Gibt true zurück, wenn das Lager leer ist, sonst false.
- int printArticleCount(): Gibt die Anzahl der Artikel im Lager zurück. Beachten Sie dabei, dass die Menge einer Artikelart mithilfe des Bestand ausgedrückt werden.
- String toString(): Eine übersichtliche Liste aller Artikel im Lager.
- Implementieren Sie den Destruktor.

Implementieren Sie auch die notwendigen Überprüfungen.

3. Aufgabe - Klasse City

Unsere LKWs werden später Waren in Städte transportieren. Entwickeln Sie eine einfache Klasse City mit den folgenden Attributen:

- Name der Stadt
- Entfernung zur Heimatstadt
- Lager der Stadt

Implementieren Sie get-Methoden für alle Attribute. set-Methoden werden keine benötigt, da die Attribute nur einmal im Konstruktor gesetzt werden und sich im Programmablauf nicht mehr verändern sollen. Der Name einer Stadt und ihre Entfernung zur Heimatstadt verändern sich nun einmal nie. Implementieren Sie außerdem noch eine toString()-Methode.

4. Aufgabe - Kopierkonstruktor und Zuweisungsoperator

Wir brauchen eine Klasse Game, die das Spiel kontrolliert. Diese Klasse enthält eine Liste mit Städten und ein Lager. Das Lager ist so etwas wie eine Kopiervorlage, d.h. wenn eine neue Stadt angelegt wird, soll das Lager aus Game in die Stadt kopiert werden. Implementieren Sie dazu für die Storehouse-Klasse den Kopierkonstruktor und den Zuweisungsoperator so, dass tatsächlich eine tiefe Kopie angelegt wird.
Game enthält ebenfalls ein Objekt der Klasse VehicleFleet (aus der Vorlesung Kurseinheit 6). Auch von VehicleFleet soll es eine Kopiervorlage geben. Implementieren Sie auch hier Kopierkonstruktor und Zuweisungsoperator