Programmiertechnik II

Sommersemester 2020 – Aufgabenblatt 6

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 6

Der zu verwendende Programmrahmen steht als Zip-Archiv `uebung6.zip` im Moodle unter der Adresse `https://moodle.hpi3d.de/course/view.php?id=137` zum Download zur Verfügung.

Aufgabe 6.1: Implementierung des ADT „Binary Knapsack“ (7 Punkte)

In der Vorlesung wurde der Abstrakte Datentyp „Binary Knapsack“ vorgestellt. Implementieren Sie in der Datei `bks.cpp` des Programmrahmens die Methoden der Klasse `BKS`, deren Header in der nicht zu verändernden Datei `bks.h` definiert wird. Dafür gelten folgende Anforderungen:

- Ein Item umfasst jeweils Gewicht, Benefit und einen Identifier.
- Der Typ des Identifiers ist anwendungsdefiniert (z. B. `int` oder `string`) und wird als Template-Parameter definiert.
- Ein `BKS`-Objekt enthält eine Liste der Items und führt die `BKS`-Berechnungen durch.
- `BKS` definiert einen Copy- und einen Default-Konstruktor sowie Assignment mittels `=`.
- Ein `BKS`-Objekt kann durch eine Itemliste, die als Vektorobjekt übergeben wird, konstruiert werden.
- Alternativ kann es durch eine Liste von Gewichts-Benefit-Paaren initialisiert werden, in diesem Fall werden die Identifier automatisch fortlaufend nummeriert vergeben. Diese Variante muss nur für `BKS`-Objekte mit `int`-Identifiern funktionieren.
- Die Methode `validateItems` überprüft, dass alle Gewichte und Benefits positiv sind und jeder Identifier nur einmal vorkommt. Wenn alle Items gültig sind, wird `true` zurückgegeben, anderenfalls `false`.
- Ein `BKS`-Objekt erlaubt die Änderung der Itemliste durch den Aufruf der `set`-Methode, welche zurückgibt, ob die neu definierte Itemliste gültig ist.
- Ein `BKS`-Objekt kann mit `select_items` zu einer als Parameter gegebenen maximalen Kapazität die Liste der ausgewählten Items (`current_selection_`) berechnen; sie wird als Indexliste formuliert, wobei sich die Indizes auf die Itemliste beziehen.
- `BKS` implementiert ein internes Caching, d. h. es merkt sich das Ergebnis für die zuletzt angefragte Kapazität. Wenn nacheinander Anfragen an `compute_for_capacity` mit gleicher Kapazität gestellt werden, wird das Ergebnis nicht neu berechnet.

In der `main`-Methode wird die Klasse für drei Testdatensätze verwendet. Überprüfen Sie, dass die Ausgabe auf der Kommandozeile den in den Kommentaren genannten Lösungen entspricht.

Aufgabe 6.2: Modellierung und Implementierung von Geldbeträgen (7 Punkte)

Modellieren und implementieren Sie eine Klasse `Amount` in `amount.cpp`, die Geldbeträge, z. B. für Waren oder Leistungen, repräsentiert. Ein Geldbetrag umfasst einen Nettobetrag, eine Mehrwertsteuer (mit normalem oder ermäßigtem Prozentsatz – nutzen Sie hierfür `enum`-Klassenelemente) und einen Bruttobetrag sowie die zugrundeliegende Währung und eine Bezeichnung (Text). Nettobetrag, Mehrwertsteuerbetrag, Mehrwertsteuersatz und Bruttobetrag sind stets konsistent. Bezeichnung, Nettobetrag und Mehrwertsteuersatz sollen per Methodenaufruf veränderbar sein. In der Klasse sollen die zwei Währungen Euro (EUR) und Dollar (USD) als `enum`-Klassenelemente eingetragen werden. Überlegen Sie sich sinnvolle Attribute und Methoden der Klasse `Amount` und implementieren Sie diese. Vervollständigen Sie zudem die Funktion `test`, die alle Methoden automatisch testen soll.

Anforderungen:

- Die Konstruktoren sollen eine gewisse Bequemlichkeit bieten und bei Bedarf auf Standardwerte zurückgreifen.
- Geldbeträge sollen automatisch in die jeweils andere Währung konvertierbar sein.
- Der Wechselkurs soll über einen festen Umwandlungsfaktor definiert sein.
- Die Default-Währung ist Euro (EUR).
- Als Mehrwertsteuersätze gelten die in Deutschland am 02.07.2020 gültigen Werte.

Aufgabe 6.3: Verrechnungskonto mit Protokoll (5 Punkte)

Die gegebene Klasse `Account` in `account.cpp` repräsentiert allgemeine Verrechnungskonten (wie in der Vorlesung behandelt). Die im Programmrahmen angegebene Implementierung dieser Klasse ist bislang nicht darauf ausgelegt, durch Vererbung spezialisiert zu werden.

Überarbeiten Sie diese Klasse und entwickeln Sie eine abgeleitete Klasse `LoggedAccount`. Die Klasse `LoggedAccount` repräsentiert ein Verrechnungskonto, das zusätzlich jede Buchung (`debit`, `credit`) und die Änderung des Überziehungsrahmens (`setLimit`) protokolliert, d. h. jedes Konto führt eine interne Liste, die die Kontovorgänge verzeichnet. Als Protokolleintrag reicht ein Schlüsselwort („credit“, „debit“, „setLimit“) und der zugehörige Betrag. Auch sollen jederzeit (also auch direkt nach Konstruktion eines `LoggedAccount`-Objekts) in dieser Transaktionsliste der Ausgangssaldo („**initial balance**“) als erster Eintrag und der aktuelle Saldo („**current balance**“) genau einmal als letzter Eintrag stehen; dazwischen sind die oben genannten Transaktionen (`credit`, `debit`, `setLimit`) aufzuführen. Ein `LoggedAccount`-Objekt unterhält eigenständig sein Protokoll, das auch abgefragt werden kann.

Zusatzaufgabe: Klausuraufgaben (2 Bonuspunkte)

Formulieren Sie je eine mögliche Klausuraufgabe zu den Inhalten der Kapitel 6 (Objektorientierte Programmierung) und 7 (Datenstrukturen) der Vorlesung und skizzieren Sie kurz zugehörige Lösungen. Die Aufgaben sollen nicht einfach konkretes Wissen, bzw. Folieninhalte abfragen, sondern das Gesamtverständnis eines Sachverhalts und die Befähigung zur Übertragung auf neue Probleme zum Gegenstand haben.

Allgemeine Hinweise zur Bearbeitung und Abgabe

- Die Aufgaben können allein oder zu zweit bearbeitet werden. Je Gruppe ist nur eine Abgabe notwendig.
- Bonuspunkte können innerhalb des Übungsblattes verlorene Punkte ausgleichen. Sie sind nicht auf andere Übungsblätter übertragbar und werden nicht über die reguläre Punktzahl hinaus angerechnet.
- Bitte reichen Sie Ihre Lösungen bis spätestens **Donnerstag, den 16. Juli um 12:00** ein.
- Die Implementierung kann auf einer üblichen Plattform (Windows, Linux, OS X) erfolgen, darf aber keine plattformspezifischen Elemente enthalten, d. h. die Implementierung soll plattformunabhängig entwickelt werden.
- Die Lösungen von theoretischen Aufgaben können als Textdatei oder PDF abgegeben werden. Benennen Sie die Dateien im Format `<Aufgabenblatt>_<Aufgabe>_<Teilaufgabe>`.
- Bestehen weitere Fragen und Probleme, kontaktieren Sie den Übungsleiter oder nutzen Sie das Forum im Moodle.
- Archivieren Sie zur Abgabe Ihren bearbeiteten Programmrahmen als Zip-Archiv und ergänzen Sie Ihre Namen im Bezeichner des Zip-Archivs im folgenden Format: **`uebung6_vorname1_nachname1_vorname2_nachname2.zip`**. Beachten Sie, dass dabei nur die vollständigen Lösungen sowie eventuelle Zusatzdaten gepackt werden (alle Dateien, die im gegebenen Programmrahmen vorhanden waren). Laden Sie keine Kompilate und temporären Dateien (*.obj, *.pdb, *.ilk, *.ncb, *.exe, etc.) hoch. Testen Sie vor dem Hochladen, ob die Abgabe fehlerfrei kompiliert und ausgeführt werden kann.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
`https://moodle.hpi3d.de/course/view.php?id=137`.