Hochschule Darmstadt
Prof. Dr. Benjamin Meyer

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

9. Februar 2021

Programmieren, Algorithmen und Datenstrukturen I, WS 2020/21
Zusatzleistung

1 Einleitung

Diese Zusatzleistung dient als Nachweis, dass Sie ausreichend mit den Inhalten der PAD1-Vorlesung vertraut sind und dieses Wissen auch praktisch anwenden können. Mit Bestehen dieser Zusatzleistung wird Ihnen ein "Mit Erfolg teilgenommen" im Zeugnis eingetragen, Nichtbestehen zählt als normaler Fehlversuch. Die Bearbeitung und die Abnahme Ihrer Lösung ähnelt den Übungsblättern des PAD-Praktikums zum Erlangen der PVL, jedoch gibt auch es einige Unterschiede. Hier daher nochmal eine kurze Zusammenfassung der Regeln:

1. Lösen Sie alle Aufgaben vollständig bis zum 15.02.2021.
2. Arbeiten Sie eigenständig (keine Gruppenarbeit!), nutzen Sie bei Fragen zuerst das Vorlesungsskript, bevor Sie googeln.
3. Begeben Sie sich zur richtigen Zeit in den richtigen Raum im BBB: vFBI/021, `https://rooms.fbi.h-da.de/r/vFBI/021`
4. Jeder bekommt zu Beginn eine genaue Zeit für einen der Break-Out-Räume und einen Slot von ca. 15min. In dieser Zeit stellen Sie (einzeln) Ihre Lösungen vor und müssen Fragen dazu beantworten.
5. Es werden keine Noten vergeben: nur bestanden / nicht bestanden. Wer jedoch eine Lösung präsentiert, die augenscheinlich nicht die eigene ist (z.B. weil nichts davon erklären werden kann), muss mit dem Vorwurf des Plagiats rechnen.
6. Die Aufgabenbeschreibung ist bewusst etwas allgemeiner gehalten als in den Praktika und soll Ihnen die Möglichkeit geben, selber die genaue Programmstruktur zu wählen. Lesen Sie die Aufgaben sorgfältig!
7. **Legen Sie Ihren Ausweis bereit und schalten Sie im Break-Out-Raum Ihre Kamera an!** Ohne Kamerabild kann Ihre Identität nicht geprüft werden und damit keine Abnahme erfolgen.

Viel Erfolg!

1.1 Lagerverwaltung

Das Lagerhaus eines großen Spediteurs soll modelliert und digital verwaltet werden. Einzulagernde Objekte erhalten eine fortlaufende `ID`, die den Objekttyp speichert (zwei identische Objekte erhalten später auch die gleiche ID), die `Größe` des Objektes (klein, mittel, groß) sowie eine `Objektbeschreibung` als String.
Der eigentliche Lagerraum soll von Ihnen als Array realisiert werden (also kein Vector oder moderne Speicherstruktur). Gehen Sie anfänglich von einer Maximalgröße des Lagers von 2 Elementen aus.
Es wird von Ihnen eine Objekt-Orientierte Lösung erwartet. Arbeiten Sie mit dynamischer Speicherverwaltung, erzeugen Sie also auch einen Destruktor.

Nach Programmstart soll folgendes Menu angezeigt und von Ihnen implementiert werden:

```
** Willkommen in der Lagerverwaltung. **
Bitte taegigen Sie ihre Eingabe:
(b): Den Lagerbestand ausgeben
(a): Ein Objekt aus dem Lager auslagern
(e): Ein neues Objekt einlagern
(x): Die Eingabe beenden
```

Implementieren Sie zusätzliche Aus- und Eingaben, damit der Nutzer weiß, was genau zu tun ist. Stellen Sie zudem sicher, dass inhaltlich falsche Eingaben nicht zu fehlerhaftem Programmverhalten führen (falsche Datentypen können Sie hier ignorieren).
Anmerkungen zu den Methoden:

- `ausgeben` soll eine Liste aller bereits eingelagerten Objekte ausgeben. Die genaue Art der Ausgabe ist Ihnen überlassen.
- `auslagern` bekommt als Parameter eine Objekt-ID. Ist diese ID im Lager vorhanden, wird das Objekt zurückgegeben und der Lagerraum freigegeben (also der Speicherplatz im Array als frei markiert – überlegen Sie sich, wie Sie freie Lagerplätze erkennen können). Ist die ID mehrfach vorhanden, reicht es, das erste gefundene Objekt auszulagern. Geben Sie die Kenndaten des gefundenen Objektes aus, danach wird es nicht weiter verwendet.

  Wurde das Objekt nicht gefunden, geben Sie ein leeres Objekt zurück.
- `einlagern` bekommt als Parameter die beiden Objekt-Attribute `Größe` und `Objektbeschreibung` und erzeugt anschließend selber ein Objekt und lagert es ein. Sollte sich schon ein identisches Objekt im Lager befinden, wird für das neue Objekt die gleiche ID verwendet, ansonsten wird die zuletzt generierte ID einfach um eins erhöht.

  Sollte das Lager bereits voll sein, geben Sie eine entsprechende Fehlermeldung zurück und lagern das Objekt nicht ein.
- Es können solange Befehle gegeben werden, bis mit (x) die Eingabe beendet werden soll.

Erzeugen Sie nun ein Lager und testen Sie Ihre Methoden.

1.2 Dynamische Lagerverwaltung

Erweitern Sie die Methode `einlagern`, sodass der zur Verfügung stehende Speicherplatz dynamisch verdoppelt wird, sollte der Platz nicht mehr ausreichen. Entsprechend kann der Speicherplatz auch wieder halbiert werden, sollten genügend Objekte aus dem Lager entfernt worden sein (also die Hälfte aller Lagerplätze frei sein).

1.3 Sortieren des Lagers

Erweitern Sie Ihr Eingangsmenu um die Möglichkeit, das Lager sortieren zu lassen. Entsprechend des Nutzerwunsches soll entweder nach der ID oder nach der Größe der Objekte sortiert werden können. Implementieren Sie hierfür selber eine Sortiermethode (in Anlehnung an die Vorlesung – Standardbibliotheken zu nutzen zählt nicht). Wie wollen Sie mit leeren Lagerplätzen umgehen?
Am Beispiel der Sortierung nach Objektgrößen, was genau ist mit einer stabilen Sortierung gemeint? Ist Ihre Implementierung stabil? Warum?