hochschule mannheim

Fakultät für Informatik
Prof. Dr. Peter Knauber

Bachelor-Studiengang Informatik
Übungen zur Vorlesung “Grundlagen der Informatik” (GDI), WS 2008/2009

Übungsblatt 8: Interfaces, Klassen und Objekte

Ausgabe am: 25.11.2008
Abgabe am: 9.12.2008

Auf diesem und den nächsten Übungsblättern sollen Sie eine Adressverwaltung (englisch: *Contact Manager*) erstellen, die inkrementell ausgebaut wird, indem schrittweise immer mehr Programmfunktionalität ergänzt wird. Mehrere Interfaces definieren eine einheitliche Schnittstelle zu Ihren Programmen.

Aufgabe 1 — 15 Punkte

Auf der Vorlesungshomepage finden Sie ein Interface *Address*, das konkrete Adressen mit Straße, Hausnummer, Postleitzahl und Stadt definiert. Dazu gehören auch Operationen zum Speichern (nur) der *Address*-Daten in eine Datei und zum Laden dieser Daten aus einer Datei.

Implementieren Sie dieses Interface.

Aufgabe 2 — 25 Punkte

Auf der Vorlesungshomepage finden Sie ein Interface *Contact*, das die Eigenschaften konkreter Kontakte definiert. Zu einem Kontakt gehören:

- Vor- und Nachname, Titel (z.B. Dr., Prof. Dr.)
- Name der Firma, bei der die Person arbeitet
- Privat- und Firmenadresse (beide vom Typ *Address*, siehe Aufgabe 1)
- Telefonnummern: privat, dienstlich und mobil
- Email-Adresse und URL

Außerdem gehören zu jedem Kontakt Operationen zum Speichern (nur) der *Contact*-Daten in eine Datei und zum Laden dieser Daten aus einer Datei (dabei werden für die *Address*-Objekte deren Methoden aufgerufen).

Implementieren Sie dieses Interface.

Aufgabe 3 — 20 Punkte

Auf der Vorlesungshomepage finden Sie ein Interface *ContactManagement*, das die Möglichkeiten der Adressverwaltung definiert. Folgende Funktionalität soll die Adressverwaltung in dieser ersten Version bieten:

- einen Kontakt hinzufügen bzw. einen Kontakt löschen
- alle aktuell im Speicher vorliegenden Kontakte in eine Datei speichern bzw. die in einer Datei vorliegenden Kontakte in die Adressverwaltung im Speicher laden; dabei werden zunächst alle eventuell bereits vorliegenden Kontakte in der Datei bzw. im Speicher gelöscht
- alle aktuell vorliegenden Kontakte in einem (dafür erstellten) passend dimensionierten(!) Array liefern, so dass sie zum Beispiel auf dem Bildschirm ausgegeben werden können.

Implementieren Sie dieses Interface.

Aufgabe 4 **25 Punkte**

Erstellen Sie eine (einfache) Umgebung, mit der Sie ihre Adressverwaltung aus den Aufgaben 1 bis 3 bedienen können, also z.B. neue Kontakte eingeben und anlegen, vorhandene Kontakte anzeigen, Kontaktdaten in eine Datei speichern oder von Datei laden. Ihre Umgebung darf nur Methoden aus den drei angegebenen Schnittstellen benutzen. (In Ihren Klassen aus Aufgabe 1 bis 3 dürfen Sie selbstverständlich auch andere Methoden definieren und benutzen!)
Beispiel für einen typischen Arbeitsablauf mit dem Programm:

- Start der Umgebung
- Eingabe des Namens der Datei, von der Kontakte geladen werden sollen, laden der Kontakte
- Ausgabe aller geladenen Kontaktdaten
- Löschen eines der Kontakte
- Eingabe eines neuen Kontakts
- Speichern aller Kontaktdaten
- Ausgabe aller geladenen Kontaktdaten

Hinweise

- Sie dürfen keines der Interfaces verändern!
- Ihr Programm sollte aus vier Klassen bestehen. Achten Sie auf eine gute Aufteilung der Aufgaben (wer ist für was zuständig?), damit Ihr Programm gut erweiterbar ist.
- Vergessen Sie nicht, Ihr Programm mit geeigneten Kommentaren zu versehen.

Aufgabe 5 **10 + 10 + 10 = 30 Punkte**

Kopieren und ändern Sie Ihre beiden Klassen mit den Primzahl-Programmen aus Aufgabe 1 und Aufgabe 2 von Übungsblatt 4 so, dass sie beide das Interface aus Aufgabe 2 von Übungsblatt 7 implementieren. Korrigieren Sie dabei zunächst eventuelle Fehler aus der Bewertung des 4. Übungsblattes. Achtung:
Die überarbeiteten Methoden sollen die Primzahlen nur berechnen, nicht ausgeben.

Erstellen Sie eine neue Klasse *PrimeCalculatorAndPrinter* mit einer *main*-Methode, in der Sie die maximale Zahl n eingeben (wie bisher), jeweils ein Objekt der Klassen *PrimzahlenMit-Schleifen* und *SiebDesEratosthenes* erzeugen und dann mit n nacheinander die beiden neuen *calculatePrime*-Methoden dieser Objekte aufrufen, so dass zwei mal (auf unterschiedliche Art und Weise) die Primzahlen bis n berechnet werden. Benutzen Sie die Iterator-Schleife, um die Inhalte der Arrays (in der *main*-Methode) in einer Zeile so auszugeben, dass man die Ergebnisse der beiden Berechnungen vergleichen kann.

Aufgabe 6 **15 Punkte**

Erstellen Sie eine neue Klasse *PrimeTimer*, die in einer Schleife zehnmal jede der beiden Primzahl-Implementierungen aus Aufgabe 5 mit einer relativ hohen Obergrenze n (z.B. 1000) aufruft. Stoppen Sie dabei mit Hilfe der beiden Methoden *initTimer* und *getDurationInMillis* aus *MakeItSimple* die jeweils benötigte Zeit und geben Sie den Durchschnittswert für jede Methode aus.
Achtung: Geben Sie die berechneten Zahlen nicht aus, weil die Ausgabe wesentlich länger dauern würde als die Berechungen.