Technische Universität Dortmund
Lehrstuhl Informatik VII
Priv.-Doz. Dr. Frank Weichert

9. März 2020

Vorkurs Informatik SS 2020
Aufgaben zur Objektorientierten Programmierung

Aufgabe 1
Die vorliegende Aufgabe beschäftigt sich mit der Realisierung einfacher Verkehrsampeln. Ziel der Aufgabe ist es, die Steuerung der Verkehrsampeln mit einem Java-Programm zu übernehmen.

1. Um Verkehrsampeln mit einem Java-Programm steuern zu können, werden Objekte des Typs `Ampel` benötigt. Deklarieren Sie dazu eine Java-Klasse mit Namen `Ampel`, welche die Boolschen Attribute `roteLampe` und `grueneLampe` beinhaltet. `true` repräsentiert eine angeschaltete Lampe, `false` eine ausgeschaltete Lampe.

2. Die Klasse `Ampel` soll einen Konstruktor haben, der die Ampel auf „rot“ initialisiert. Ferner soll die Klasse um die Methode `umschalten` erweitert werden, die ein Umschalten der Ampelfarbe bewirken soll. Schließlich soll die Klasse noch eine Methode `druckeZustand` haben, die den Wert der Attribute auf dem Bildschirm ausgibt.

3. Nun soll eine Kreuzung zweier Einbahnstraßen modelliert werden. Deklarieren Sie dazu die Klasse `Kreuzung`, die aus zwei Ampeln besteht. Der Konstruktor soll derart implementiert werden, dass eine der beiden Ampeln auf „rot“ und die andere auf „grün“ initialisiert wird. Zusätzlich soll die Methode `fahrtrichtungAendern` implementiert werden, die die beiden Ampeln entsprechend umspringen lassen soll. Auch die Klasse `Kreuzung` soll eine Methode `druckeZustand` zur Ausgabe des aktuellen Kreuzungszustands besitzen.

4. Testen Sie Ihre Klassen, indem Sie im Hauptprogramm (`main`-Methode) ein Objekt vom Typ `Kreuzung` erzeugen, die Fahrtrichtungen mehrmals ändern und sich die Änderungen ausgeben lassen.

Aufgabe 2
In dieser Aufgabe soll eine Klasse `Multimenge` deklariert werden – eine Multimenge ist eine Menge, in der Elemente mehrfach enthalten sein dürfen. Die Klasse soll eine Multimenge über einem ganzzahligen Zahlenbereich $0, \ldots, n$ durch ein `int`-Array mit $n+1$ Einträgen repräsentieren. Jedem Array-Element ist dabei ein potentielles Element der Multimenge zugeordnet. Wenn ein Array-Element einen Eintrag $h > 0$ hat, liegt das entsprechende Element h-mal in der Multimenge vor. Für $h = 0$ ist das Element somit nicht in der Multimenge enthalten.

1. Implementieren Sie innerhalb einer Klasse `Multimenge` einen Konstruktor der Klasse `Multimenge`, der die Elemente aus dem Zahlenbereich $0, \ldots, n$ enthalten kann.

2. Schreiben Sie eine Methode `fuegeEin` mit den Attributen `element` und `anzahl`, jeweils vom Typ Integer und einem Booleschen Rückgabewert. Falls `element` im definierten Zahlenbereich liegt, fügt die Methode das Element `element` mit der Häufigkeit `anzahl` in die Multimenge ein und gibt `true` zurück, sonst `false`.

3. Ergänzen Sie die Klasse `Multimenge` um eine Methode `vereinigung` mit einem Attribut `M` vom Typ `Multimenge`, welche die Vereinigung zweier Multimengen durchführt. Falls die Menge, die das Objekt repräsentiert, und die Multimenge `M` nicht den gleichen Zahlenbereich haben, wird

`null` zurückgegeben. Andernfalls wird eine Multimenge über dem gesamten Zahlenbereich zurückgegeben, welche die Vereinigung der beiden Multimengen ist. Dies bedeutet, dass ein Element, das in der einen Menge g-mal und in der anderen h-mal vorkommt, in der Vereinigung $(g + h)$-mal auftritt.

Aufgabe 3
Realisieren Sie eine Klasse `Time`, die in der Lage ist, eine Uhrzeit, bestehend aus Stunde, Minute und Sekunde, darzustellen. Intern soll die Uhrzeit durch einen einzigen Wert vom Typ `int` dargestellt werden, der die Uhrzeit in Sekunden speichert.

1. Die Klasse soll sowohl einen parameterlosen Konstruktor, der ein `Time`-Objekt mit der aktuellen Uhrzeit initialisiert und einen Konstruktor mit drei Argumenten (für Stunde, Minute und Sekunde), der die Konstruktion beliebiger Uhrzeiten ermöglicht, zur Verfügung stellen. Hinweise für die Abfrage der aktuellen Uhrzeit finden Sie im Kapitel 11.4.4 der Online-Version des Buches *Java ist auch eine Insel* unter *http://openbook.galileocomputing.de/javainsel/index.html*

2. Erweitern Sie die Klasse um die Methoden `getHour`, `getMinute`, `getSecond`, `setHour`, `setMinute` und `setSecond` zum Einstellen und Abfragen der Uhrzeit.

3. Realisieren Sie eine Methode `incrementSecond`, um die Uhrzeit eine Sekunde weiter zu stellen. Steht die Uhrzeit auf `23:59:59`, soll sie auf `00:00:00` geändert werden.

Aufgabe 4
Polymorphismus ist ein zentrales Konzept der objektorientierten Programmierung. Zusammen mit abstrakten Klassen ergibt sich ein mächtiges Instrumentarium für eine Vielzahl von Anwendungen. In dieser Aufgabe soll ein Programm gemäß der nachfolgenden Spezifikation erstellt werden.

1. Schreiben Sie eine abstrakte Klasse `Angestellter`, die Daten über einen Angestellten enthält. Die Klasse hat die beiden Methoden `monatsGehalt` und `jahresGehalt`, die das Gehalt des Angestellten für einen Monat bzw. ein komplettes Jahr ermitteln. `monatsGehalt` ist abstrakt und wird erst in abgeleiteten Klassen implementiert. Dagegen ist `jahresGehalt` bereits in der Basisklasse implementiert und realisiert das Jahresgehalt durch Multiplikation des Monatsgehalts mit `12`.

2. Schreiben Sie die abgeleiteten Klassen `Manager`, `Sekretaerin` und `Verkaeufer` und implementieren sie darin die Methode `monatsGehalt` wie folgt:

   - Die Sekretärin bekommt ein festes Gehalt je Monat.
   - Der Verkäufer bekommt ein festes Gehalt und zusätzlich anteilige Verkaufsprovisionen.
   - Der Manager bekommt ein festes Gehalt und zusätzlich eine Zulage für jede der folgenden Aktivitäten, sofern er sie ausübt:
     - Mitglied der Geschäftsführung.
     - Führungskraft für mehr als 12 Mitarbeiter.
     - Mehr als 10-jährige Firmenzugehörigkeit.

   Neben `monatsGehalt` sollen in den abgeleiteten Klassen Methoden zum Zugriff und zur Abfrage der einzelnen Gehaltsbestandteile zur Verfügung stehen.

3. Schreiben Sie ein Testprogramm, das ein Array von Angestellten unterschiedlichen Typs erzeugt und berechnen Sie das monatliche und jährliche Gehalt für alle Angestellten. Verwenden Sie innerhalb des Testprogramms keine expliziten Verzweigungen, um den Typ des Angestellten herauszufinden, sondern nutzen Sie die Vorteile polymorpher Variablen.