Universität Leipzig
Institut für Informatik

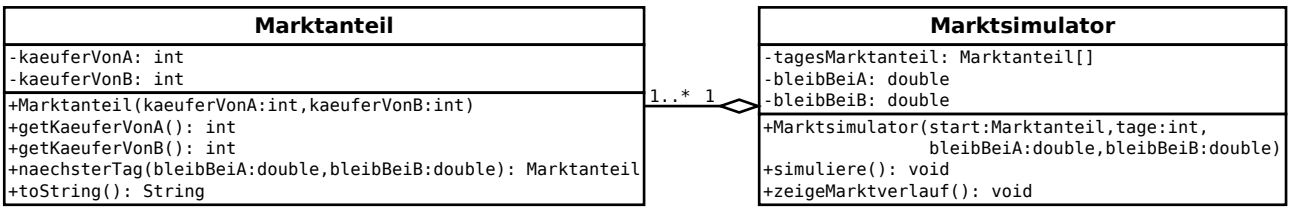
WS 20/21

Modellierung und Programmierung 1 – Übungsserie 2

Abgabetermin: 02.12.2020, 22:00 Uhr
Abgabeformat: 1 ZIP-Datei
Max. Punkte: 28

Klassen, Arrays und Vererbung

1. **Marktsimulation** *(10 Punkte)*
   Bei dieser Aufgabe soll ein Programm erstellt werden, welches einen Markt simuliert bei dem die Kunden jeden Tag zwischen zwei Produkten wählen müssen. Die Kunden wechseln dabei mit einer bestimmten Wahrscheinlichkeit zu dem jeweils anderen Produkt.



   a) *(5 Punkte)* Klasse `Marktanteil`
   Schreiben Sie die Klasse `Marktanteil` für das gegebene Klassendiagramm und implementieren Sie alle Methoden.

   - Der Konstruktor initialisiert die Attribute `kaeuferVonA` und `kaeuferVonB` mit den als Parameter übergebenen Werten. Die beiden Attribute speichern jeweils die Anzahl der Käufer für Produkt A und B.
   - Die Methode `naechsterTag` enthält die Kernfunktionalität der Simulation. Sie generiert den Marktanteil für den nächsten Tag und gibt ihn zurück. Zur Erstellung des neuen Marktanteils wird für jeden Käufer einzeln simuliert, dass er bei Produkt A mit der als Parameter übergebenen Wahrscheinlichkeit `bleibBeiA` bleibt und ansonsten zu Produkt B wechselt. Das gleiche wird für die Käufer von Produkt B simuliert.

     Der generierte Marktanteil hat also eine andere Verteilung der Käufer der Produkte, aber die Anzahl aller Käufer bleibt gleich.

     Hinweis: Für Zufallszahlen existiert in Java die statische Methode `Math.random()`. Diese gibt einen gleichverteilten Zufallswert aus dem Intervall $[0, 1)$ zurück.
   - Die Methode `toString` der Klasse `Marktanteil` soll die Marktaufteilung in folgender Form als String zurückgeben:
     `Marktanteil Käufer Produkt A: 250 (25%) Käufer Produkt B: 750 (75%)`

     Hinweis: Zum Runden der Prozente kann die Funktion `Math.round` verwendet werden.

   b) *(4 Punkte)* Klasse `Marktsimulator`
   Schreiben Sie die Klasse `Marktsimulator` für das gegebene Klassendiagramm und implementieren Sie alle Methoden.

   - Der Konstruktor initialisiert das Attribut `tagesMarktanteil` als Array der Länge `tage`+1. Im ersten Eintrag des Array wird der als Parameter gegebene Marktanteil `start` hinterlegt.

Weiterhin werden die Attribute `bleibBeiA` und `bleibBeiB` mit den entsprechenden Parametern initialisiert. Sie speichern jeweils wie wahrscheinlich es ist, dass ein Käufer bei Produkt A bzw. B bleibt.

Damit der gesamte Array sinnvoll belegt ist, wird abschließend die Methode `simuliere` aufgerufen.

- Die Methode `simuliere` simuliert die Marktveränderung mithilfe der Methode `naechsterTag` der Klasse `Marktanteil`. Für die Parameter `bleibBeiA` und `bleibBeiB` der Methode `naechsterTag` werden jeweils die entsprechenden Attribute verwendet.

  Die erzeugten Marktanteile werden nacheinander in dem Array `tagesMarktanteil` gespeichert. Am Ende enthält also jedes Feld des Arrays einen Marktanteil. Dabei ist im ersten Eintrag der Startmarktanteil der im Konstruktor übergeben wurde und im letzten Eintrag der des letzten Tages.

- Die Methode `zeigeMarktverlauf` gibt den simulierten Marktverlauf in der unten gezeigten Form aus.

  Beispielausgabe für 10 Tage und der Startaufteilung 250 Käufer von Produkt A und 750 Käufer von Produkt B:

  ```
  Marktentwicklung
  Produkt A: 250 415 495 536 585 581 561 573 590 588 606
  Produkt B: 750 585 505 464 415 419 439 427 410 412 394
  Finale Marktaufteilung:
  Marktanteil Käufer Produkt A: 606 (61%) Käufer Produkt B: 394 (39%)
  ```

  Es werden also jeweils in einer Zeile die Anzahl der Tageskäufe von Produkt A und B angezeigt. Die letzte Zeile zeigt die Marktaufteilung am letzten Tag im Detail.

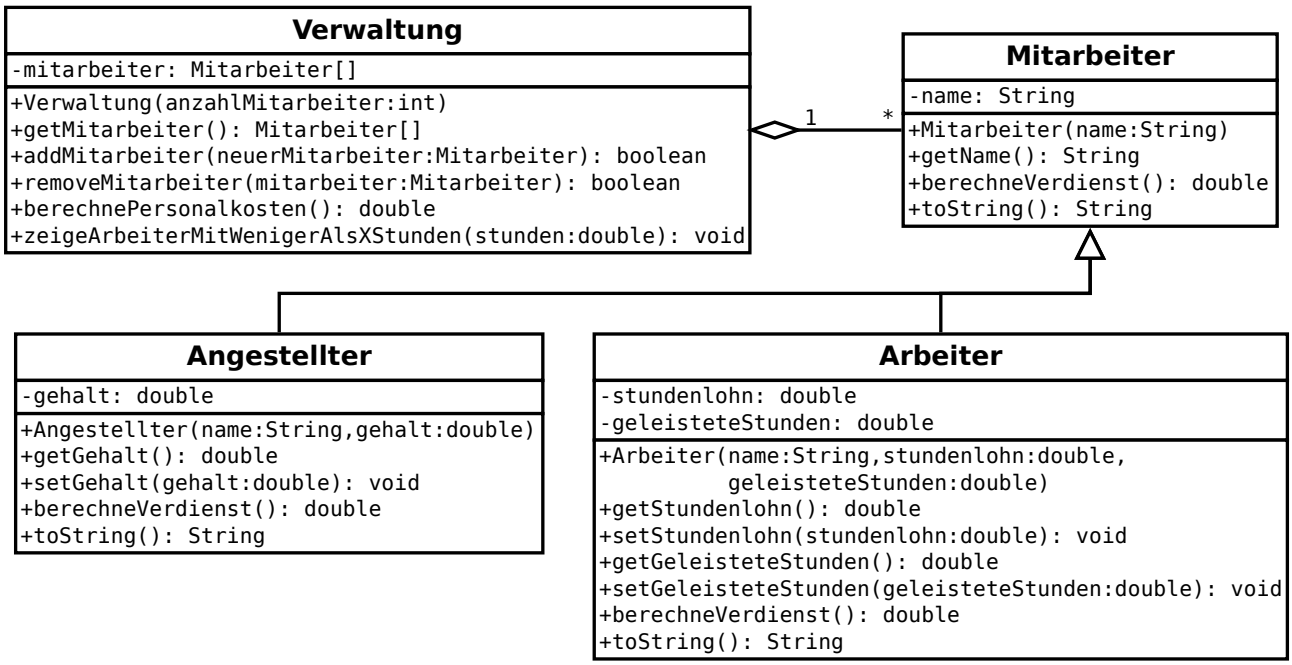
c) *(1 Punkt)* Klasse `Main`
Erstellen Sie in der `main`-Methode einer Klasse `Main` eine Instanz der Klasse Marktsimulator und setzen Sie die Konstruktorparameter wie folgt:

- `start` - Startmarktanteil: eine Instanz der Klasse Marktanteil mit der Aufteilung 5000 zu 5000
- `tage` - Anzahl der simulierten Tage: 20
- `bleibBeiA` - die Wahrscheinlichkeit, dass ein Käufer bei Produkt A bleibt: 0.8
- `bleibBeiB` - die Wahrscheinlichkeit, dass ein Käufer bei Produkt B bleibt: 0.7

Abschließend rufen Sie für die erzeugte Marktsimulation-Instanz die Methode `zeigeMarktverlauf` auf.

Hinweis: Bei den gegebenen Wahrscheinlichkeiten sollte sich die Marktaufteilung (bei mindestens 20 simulierten Tagen) ungefähr zum Verhältnis 60/40 entwickeln.

2. **Personalverwaltung** *(18 Punkte)*
Ziel dieser Aufgabe ist die Programmierung einer Personalverwaltung. Wichtiges Element dabei ist, dass bei dem Personal zwischen Arbeitern und Angestellten unterschieden wird. Arbeiter erhalten einen monatlichen Verdienst in Abhängigkeit von den geleisteten Stunden, während Angestellte ein festes Gehalt als Verdienst pro Monat bekommen.



a) *(9 Punkte)* Klassen `Mitarbeiter`, `Angestellter` und `Arbeiter`
Schreiben Sie die Java-Klassen `Mitarbeiter`, `Angestellter` und `Arbeiter` für das gegebene Klassendiagramm und implementieren Sie alle Methoden.

- Die Konstruktoren initialisieren die Attribute.
- Die Methode `berechneVerdienst` soll für die jeweilige Klasse jeweils unterschiedlich implementiert werden:
  - `Mitarbeiter`: Gibt 0 zurück.
  - `Angestellter`: Gibt das Gehalt zurück.
  - `Arbeiter`: Gibt das Produkt aus geleisteten Stunden und Stundenlohn zurück.
- Die Methode `toString` soll für die jeweilige Klasse wie folgt unterschiedlich implementiert werden:
  - `Mitarbeiter`: Gibt den Namen des Mitarbeiters zurück.
  - `Angestellter`: Gibt einen String in der Form "`Karl Knallgas Gehalt: 5000.00 Euro`" zurück.
  - `Arbeiter`: Gibt einen String in der Form "`Tommy Tadel Stundenlohn: 35.00 Euro geleistete Stunden: 7.0 Stunden`" zurück.

Hinweise:

- Die Geldbeträge sollen auf zwei Nachkommastellen gerundet werden. Dazu gibt es in Java unter anderem die statische Funktion `String.format( ... )`.Informieren Sie sich in der Java Dokumentation (oder einer anderen Quelle) über diese oder eine andere Funktion zur Formatierung und nutzen Sie diese.
- Die Stunden können einfach als Dezimalzahl ausgegeben werden.

b) *(7 Punkte)* Klasse `Verwaltung`
Schreiben Sie eine Java-Klasse `Verwaltung` für das gegebene Klassendiagramm und implementieren Sie alle Methoden.

- Der Konstruktor initialisiert das Attribut `mitarbeiter` mit einem Array der Länge `anzahlMitarbeiter`
- Die Methode `addMitarbeiter` fügt den Mitarbeiter an einen freien Platz des `mitarbeiter`-Attribut ein und gibt `true` zurück. Sollte kein freier Platz vorhanden sein, gibt sie entsprechend `false` zurück.
- Die Methode `removeMitarbeiter` entfernt den Mitarbeiter aus dem Array `mitarbeiter` und gibt `true` zurück. Falls der Mitarbeiter nicht im Array enthalten war, wird `false` zurückgegeben.
- Die Methode `berechnePersonalkosten` gibt die Personalkosten aller Mitarbeiter zurück. Die Personalkosten entsprechen der Summe der Verdienste aller Mitarbeiter.
- Die Methode `zeigeArbeiterMitWenigerAlsXStunden` gibt Arbeiter mit weniger als `stunden` geleisteten Stunden aus.

  Beispielausgabe:

  ```
  Arbeiter mit weniger als 10.0 Stunden:
  Stefan Stulle Stundenlohn: 40.00 Euro geleistete Stunden: 5.0 Stunden
  Tommy Tadel Stundenlohn: 35.00 Euro geleistete Stunden: 7.0 Stunden
  ```

Hinweis: Beachten Sie, dass die Methoden auch funktionieren sollen, falls der `Mitarbeiter`-Array der Klasse `Verwaltung` `null`-Einträge zwischen belegten Einträgen enthält.

c) *(2 Punkte)* Klasse `Main`
Erstellen Sie in der `main`-Methode einer Klasse `Main` eine Instanz der Klasse `Verwaltung` mit Platz für 20 Mitarbeiter.

Führen Sie dann nacheinander folgende Dinge aus:

- Fügen Sie der Instanz der Klasse `Verwaltung` jeweils 4 Angestellte und 4 Arbeiter hinzu.
- Geben Sie die Personalkosten aus.
- Entfernen Sie anschließend wieder einen Mitarbeiter.
- Geben Sie noch einmal die Personalkosten aus.
- Lassen Sie alle Arbeiter ausgeben, welche weniger als eine bestimmte Stundenanzahl haben. Achten Sie darauf, dass dies mindestens auf einen der von Ihnen erstellten Arbeiter zutrifft.

Hinweis: Sie müssen beim Hinzufügen und Entfernen die Rückgabe der Methoden `addMitarbeiter` und `removeMitarbeiter` nicht abprüfen.