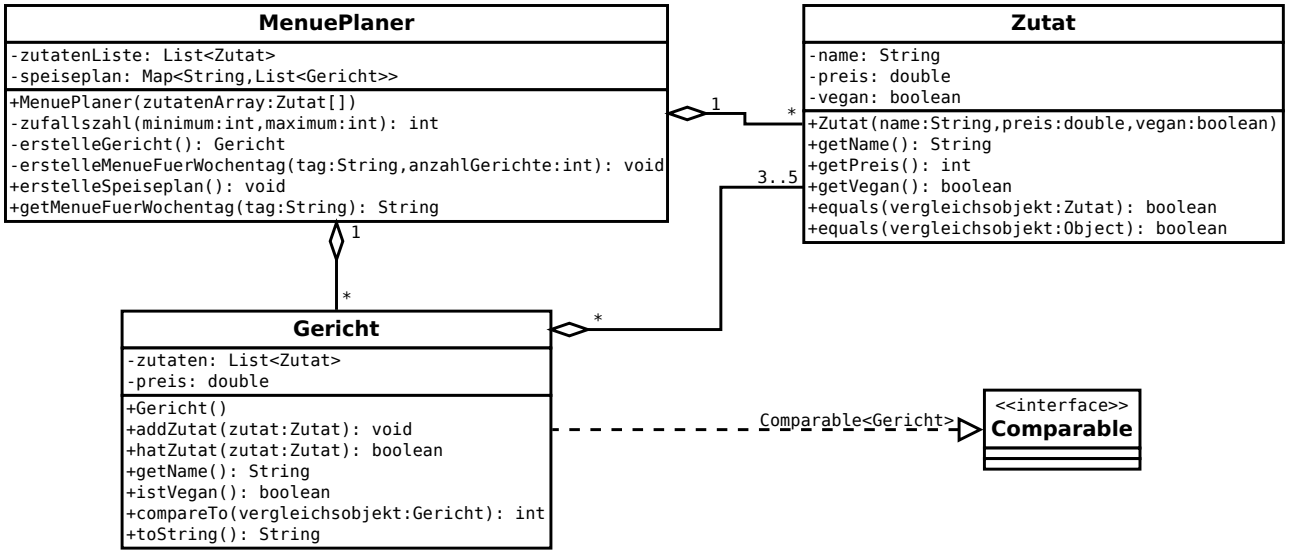
Universität Leipzig **WS 20/21**
Institut für Informatik

Modellierung und Programmierung 1 – Übungsserie 3

Abgabetermin: 16.12.2020, 22:00 Uhr
Abgabeformat: 1 ZIP-Datei
Max. Punkte: 25

Collections und Strings

Aufgrund von Sparmaßnahmen soll die Menüplanung in der Mensa automatisiert durch ein Computerprogramm erfolgen, welches Sie erstellen sollen. Als Grundlage für das Programm dient der beigefügte Code sowie das folgende UML-Diagramm:



Die Klasse `Zutat` ist bereits vollständig implementiert. `name` ist der Name und `preis` der Verkaufspreis einer Zutat. `vegan` gibt an, ob eine Zutat vegan ist. Mittels der `equals`-Methode können zwei Zutaten auf Gleichheit getestet werden.

Hinweis: Verwenden Sie bei der Erzeugung von Strings, die aus einer variablen Anzahl von Teilstrings bestehen, Instanzen der Klasse `StringBuilder`!

1. **Gericht** *(14 Punkte)*

   - *(2 Punkte)* Implementieren Sie entsprechend des obigen UML-Diagramms den Konstruktor sowie sämtliche Attribute. `zutaten` ist eine Liste aller Zutaten, aus denen das Gericht besteht. Sie soll bei der Erzeugung eines Gerichts zunächst leer sein. `preis` ist der Verkaufspreis eines Gerichts, der Initialwert soll 1.0 betragen.

   Implementieren Sie weiterhin die folgenden Methoden:

   - *(2 Punkte)* `addZutat` soll die angegebene Zutat zur Zutatenliste hinzufügen und den Verkaufspreis des Gerichts um den Verkaufspreis der Zutat erhöhen.
   - *(1 Punkt)* `hatZutat` soll zurückgeben, ob das Gericht die angegebene Zutat enthält.

- *(2 Punkte)* `getName` soll anhand der Zutatenliste einen neuen Namen für das Gericht erzeugen. Der Name besteht dabei aus der Aneinanderreihung der jeweils ersten drei Buchstaben der Namen aller enthaltenen Zutaten.

  Beispiel: Für ein Gericht, das aus "Kartoffeln", "Blumenkohl" und "Möhren" besteht, ergibt sich der Name "KarBluMöh".

  Hinweis: Der Fall, dass der Name einer Zutat aus weniger als drei Buchstaben besteht, muss nicht berücksichtigt werden.
- *(2 Punkte)* `istVegan` soll zurückgeben, ob das Gericht ausschließlich aus veganen Zutaten besteht.
- *(3 Punkte)* `compareTo` soll es ermöglichen, das aktuelle Gericht (für welches die Methode aufgerufen wurde) mit einem zweiten Gericht `vergleichsobjekt` anhand ihres Preises zu vergleichen:
  - Ist `vergleichsobjekt` teurer als das aktuelle Gericht, dann soll -1 zurückgegeben werden.
  - Sind beide Gerichte gleich teuer, dann soll 0 zurückgegeben werden.
  - Ist `vergleichsobjekt` günstiger als das aktuelle Gericht, dann soll 1 zurückgegeben werden.

  Achten Sie auf die im UML-Diagramm angegebene Abhängigkeit von der Schnittstelle `Comparable`.
- *(2 Punkte)* `toString` soll einen String erzeugen, der aus den folgenden Komponenten besteht:
  - der Name des Gerichts
  - der String "(vegan)", falls das Gericht vegan ist
  - die Namen sämtlicher Zutaten
  - der auf zwei Nachkommastellen gerundete Verkaufspreis

  Die einzelnen Komponenten sollen durch Leerzeichen getrennt werden.

  Beispiel:
  `BroKarBlu (vegan) Broccoli Kartoffeln Blumenkohl 2.59 Euro`

  Hinweis: für die Rundung und Ausgabe der Nachkommastellen bietet sich die statische Methode `String.format` an.

2. **MenuePlaner** *(10 Punkte)*
   Der Konstruktor, die Methode `zufallszahl` sowie sämtliche Attribute sind im beigefügten Code bereits vorgegeben. `zutatenListe` ist eine Liste aller verfügbaren Zutaten. `speiseplan` soll für jeden Wochentag, an dem die Mensa geöffnet hat, ein Menü (eine Liste von Gerichten) speichern.

   Die Methode `zufallszahl` können Sie nutzen, um eine Zufallszahl zwischen einem Minimum und einem Maximum (einschließlich **beider** Werte) zu erzeugen.

   Implementieren Sie die folgenden Methoden:

   - *(4 Punkte)* `erstelleGericht` soll ein Gericht nach dem folgenden Schema erstellen:

```
gegeben: zutatenListe (Liste aller zur Verfügung stehenden Zutaten)
initialisiere anzahlZutaten mit einem zufälligen Wert zwischen 3 und 5
initialisiere hinzugefuegt mit 0
initialisiere gericht als Gericht
SOLANGE hinzugefuegt < anzahlZutaten:
    wähle zutat zufällig aus zutatenListe
    WENN zutat nicht in gericht enthalten:
        füge zutat zu gericht hinzu
        erhöhe hinzugefuegt um 1
gebe gericht zurück
```

- *(2 Punkte)* `erstelleMenueFuerWochentag` soll für einen gegebenen Wochentag ein Menü erstellen, welches aus der angegebenen Anzahl an Gerichten besteht. Das erstellte Menü soll im Speiseplan gespeichert werden (ein für den gegebenen Wochentag bereits vorhandenes Menu kann dabei überschrieben werden).
- *(1 Punkt)* `erstelleSpeiseplan` soll für "Montag" und "Dienstag" ein Menü mit 4 Gerichten und für "Mittwoch" ein Menü mit 5 Gerichten erstellen.
- *(3 Punkte)* `getMenueFuerWochentag` soll das Menü für den angegebenen Wochentag zurückgeben. Gehen Sie hierfür wie folgt vor:
  - Überprüfen Sie, ob der Wochentag im Speiseplan vorkommt. Ist dies der Fall, dann soll der Wochentag zur Rückgabe hinzugefügt werden. Andernfalls soll "Die Mensa hat geschlossen!" zurückgegeben werden.
  - Sortieren Sie die Liste von Gerichten des entsprechenden Wochentags nach aufsteigendem Verkaufspreis.
    Hinweis: Eine Liste kann mit der statischen Methode `Collections.sort` sortiert werden.
  - Fügen Sie die nach Verkaufspreis sortierten Gerichte zur Rückgabe hinzu.

  Beispiel:

```
Montag:
BohKürCur Bohnen Kürbis Currysauce 2.82 Euro
ReiFalSei (vegan) Reiskornnudeln Falafel Seitan 2.89 Euro
SeiBulBraGar Seitan Bulgur Braune Sauce Garnelen 3.91 Euro
ReiForSeiRum Reiskornnudeln Forelle Seitan Rumpsteak 4.55 Euro
```

3. **Main** *(1 Punkt)*
   Gehen Sie in der Klasse `Main` wie folgt vor:
   - Erstellen Sie die `main`-Methode.
   - Erzeugen Sie einen neuen `MenuPlaner`, die hierfür notwendige Zutatenliste erhalten Sie von der Methode `getZutatenListe`.
   - Erstellen Sie einen Speiseplan.
   - Geben Sie das Menü für "Donnerstag" und "Mittwoch" aus.