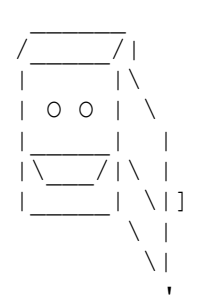
Übung 5

Hilfe, mein Kühlschrank spricht zu mir – Teil 2



Diesmal lernt der Kühlschrank endlich sprechen. Außerdem gilt es ein paar kleine Erweiterungen zu implementieren.

Die Ergebnisse sollen in einer Klasse implementiert werden, die von der FridgeEx.java Klasse der letzten Musterlösung abgeleitet ist (in template_ue05.rar enthalten).

1. Sprachausgabe (optional)

Die Sprachausgabe soll mit Hilfe von MARY erfolgen. Verwendet dazu den OpenMary 4.1.1 Download von der MARY-Webseite http://mary.dfki.de. Einfach herunterladen und starten (Doppelklick oder mit java -jar von der Kommandozeile), über die sich öffnende Installations-GUI zusätzlich die Sprachunterstützung für deutsch *und* eine deutsche Stimme (wichtig: beides!) installieren.

Das Installationstool legt (auf Wunsch) Icons auf dem Desktop an. Nach der abgeschlossenen Installation das "MARY server"-Icon auf dem Desktop anklicken und schon läuft der Mary-Server (er muss immer laufen, wenn der Kühlschrank sprechen soll). Mit dem mitgelieferten "MARY client" oder "MARY web interface" könnt ihr testen, ob die deutsche Sprachausgabe funktioniert (ebenfalls über Icons auf dem Desktop zu starten).

Der Client-Code in der Klasse MaryTtsClient.java in template_ue05.rar fungiert als "Brücke" zwischen der Fridge-Klasse und dem MARY Server; er benötigt 2 jar-Archive (maryclient.jar und log4j-1.2.15.jar), welche beide im Template unter lib/ enthalten sind. Einbinden der beiden JAR-Dateien für den MaryTTS-Client mit Project/Properties/Java Build Path/Add JARs... und (auf FridgeParser.java) Run/Run Configurations/Classpath/Add JARs. Es muss kein MARY- Quellcode in Eclipse geladen oder angepasst werden. Die Zeile

```
MaryTtsClient.say(message);
```

gibt message (einen String) über die Sprachausgabe aus. Wenn alle Ausgaben des Kühlschranks bereits über eine Methode laufen (`Fridge.message()` in der Musterlösung), genügt das Einfügen dieser einen Zeile, um den Kühlschrank-Code zum Sprechen zu bringen!

Die Integration der MARY-Sprachausgabe besteht also im Prinzip nur in der einen Zeile mit dem `say()`-Aufruf. Dieser Teil der Aufgabe ist optional, da nicht garantiert werden kann, dass die Sprachausgabe auf jedem Rechner funktioniert (insbes. Mac haben wir nicht getestet, sollte aber funktionieren). Der nun folgende Rest der Aufgabe (Korrektur der Zahlausgabe) kann unabhängig vom Funktionieren des TTS-Systems implementiert und getestet werden.

2. Sprechen von Zahlausdrücken (6 Punkte)

Zahlen und Uhrzeiten werden vom TTS-Modul weitgehend richtig gesprochen. Probleme gibt es bei der Ausgabe von Genus und Numerus, z.B. 1 Butter ("Eins Butter"). Die Zahlausgabe soll nun so

korrigiert werden, dass solche Ausdrücke richtig gesprochen werden. Aus "1 Butter" soll also "Ein**e** Butter" werden, aus "3 Joghurt" "3 Joghurt**s**", aus "0 Käse" "**Kein** Käse" usw.

Implementiere dazu eine Datenstruktur mit den nötigen Informationen, um die Worte (zumindest die, die in inventory.xml vorkommen) richtig aussprechen zu können, d.h. Korrektur bei 0, 1, Numerus, Genus. Falls nicht schon geschehen, kapsele dazu die `System.out.println()`-Anweisung wie in der Musterlösung von Übung 4 in eine Methode `Fridge.message(String message)`. In dieser soll neben der Text- nun auch die Sprachausgabe über MARY erfolgen. Ersetzung des Zahlausdrucks im String mittels regulärer Ausdrücke! Die Eingabe von Produkten soll nicht angepasst werden (weiterhin 2 Wurst statt 2 Würste).

3. Implementiere eine weitere Bestandsanfrage und die entsprechenden Antworten (4 Punkte)

Wovon (ist | sind) [noch] [(weniger | mehr) als] <Anzahl> da?

4. Implementiere die Bearbeitung der Anfrage "Wie (ist|wird) das Wetter [heute]?" (5 Punkte)

Das aktuelle Wetter soll aufgrund von Webinhalten als Antwort geliefert werden (z.B. "Heute ist es 14 Grad Celsius" oder "Heute ist es heiter bei 13 Grad"). Folgende Methode (aus dem mitgelieferten HtmlContent.java) gibt den Inhalt einer Webseite im HTML-/XML-Quellcode als String zurück:

```
String HtmlContent.getFromUrl(String httpUrl, String encoding)
```

Der Google Wetterdienst liefert das aktuelle Wetter für eine Stadt als XML String:

```
http://www.google.com/ig/api?weather=saarbruecken&hl=de
```

Extrahiere mittels geeigneten XPath Ausdrücken die aktuelle Temperatur und Wetterlage aus dem XML String.

Inhalt von template_ue05.rar:

```
data/uebung05/inventory.xml
lib/maryclient.jar
lib/log4j-1.2.15.jar
src/javakurs/uebung05/Fridge.java
src/javakurs/uebung05/FridgeEx.java
src/javakurs/uebung05/FridgeParser.jj
src/javakurs/uebung05/HtmlContent.java
src/javakurs/uebung05/MaryTtsClient.java
```