Di, 5.16. Std, LOK Informatik: Schüler sollen Wirtschaftssimulation.zip
R 348 runterladen & Aufgaben 1-3 bearbeiten.
LG Daniel Gnida

Wirtschaftssimulationen

Eine Wirtschaftssimulation soll in Java mit grafischer Benutzeroberfläche programmiert werden. Ausgangspunkt bildet hier als Beispiel eine Milchfarm. Später kann dieses Beispiel in seiner Komplexität um ein Vielfaches erweitert werden oder Sie können alternative Wirtschaftssimulationen mit ähnlichen Funktionsprinzipien entwerfen.

Zunächst geht es im Milchfarm-Beispiel darum ein Spiel zu entwerfen, in dem der Spieler eine Milchfarm verwaltet. Er hat dabei die Möglichkeit Kühe zu kaufen und wieder zu verkaufen sowie die Milch zu verkaufen, die seine Kühe produzieren. Eine Kuh produziert am Tag 50 Liter Milch. Mit jedem Klick auf den Button mit der Beschriftung *Nächster Tag* soll entsprechend die vorhandene Menge Milch ansteigen.

Folgende Komponenten stehen Ihnen zur Verfügung:

Textfelder	**Buttons**
`TAnzahlKuh`	`BKuhKaufen`
`TAnzahlMilch`	`BKuhVerkaufen`
`TVermoegen`	`BMilchVerkaufen`
`TTag`	`BNaechsterTag`
`TKuhPreis`	
`TMilchPreis`	

1 Der Quellcode für den Button `BKuhKaufen` ist wie folgt implementiert:

```
1    double Vermoegen = Double.parseDouble(TVermoegen.getText());
2    int AnzahlKuh = Integer.parseInt(TAnzahlKuh.getText());
3    int KuhPreis = Integer.parseInt(TKuhPreis.getText());
4    Vermoegen = Vermoegen - KuhPreis;
5    AnzahlKuh = AnzahlKuh + 1;
6    TVermoegen.setText(Vermoegen);
7    TAnzahlKuh.setText(AnzahlKuh);
```

Alle Programmteile funktionieren in der Regel nach demselben Prinzip:

1. Auslesen der erforderlichen Werte aus den Textfeldern (Zeile 1 – 3)
2. Verändern der Werte (Zeile 4, 5)
3. Zurückschreiben der veränderten Wert in die Textfelder (Zeile 6, 7)

Beim Starten des Programms werden Sie merken, dass man bei der derzeitigen Implementierung des Buttons `BKuhKaufen` die Möglichkeit hat, unendlich viele Kühe zu kaufen. Außerdem geht das Vermögen dadurch in den negativen Wertebereich. Verhindern Sie dies, indem Sie den Quellcode so umschreiben, dass der Spieler nur noch so viele Kühe an einem Tag kaufen kann, wie sein Vermögen dies zulässt.

2 Implementieren Sie den Quellcode für die Buttons

- `BKuhVerkaufen`
- `BMilchVerkaufen`
- `BNaechsterTag`

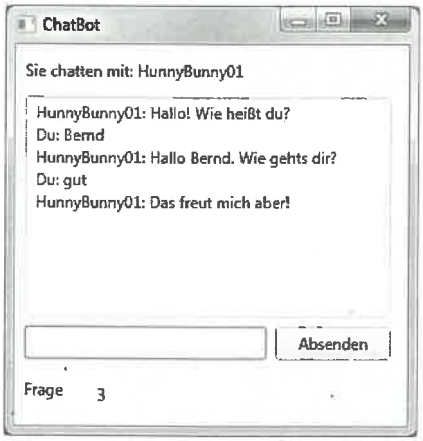
3 Spielen Sie das Spiel nach folgendem Szenario:

> Ein Onkel aus dem entfernten Verwandtenkreis vererbt Ihnen seine Milchfarm. Aber eigentlich haben Sie gar nicht vor Milchbauer zu werden, sondern hatten eine Weltreise geplant, für die Sie 10.000 € benötigen.

Erarbeiten Sie eine Strategie, mit der man möglichst schnell mit der Milchfarm ein Vermögen von 10.000 € erwirtschaftet.

Übungen

4.1 Gegeben ist nebenstehender ChatBot, der sich in *ChatBot.zip* befindet. ChatBots sollen einem Benutzer einen Chat-Parner simulieren, damit dieser den Eindruck hat, er unterhalte sich mit einer realen Person in einem Internet-Chat.
Wird der Button `bAbsenden` betätigt, reagiert der ChatBot entsprechend auf die Eingabe des Nutzers.



a) Analysieren Sie die Methode `bAbsenden_ActionPerformed()` und erläutern Sie die Funktionsweise anhand des Java-Codes.

b) Erweitern Sie Ihren ChatBot so, dass dieser noch mindestens drei zusätzliche Fragen stellt und möglichst flexibel auf mögliche Eingaben des Benutzers reagieren kann. Hilfreich können auch die Erweiterungsmöglichkeiten in Aufgabe 4.2 sein. Im Optimalfall sollte für einen Außenstehenden tatsächlich der Eindruck entstehen, er unterhalte sich mit einem echten Menschen.

4.2 Unter
https://de.wikipedia.org/wiki/Chatbot
sind diverse Funktionsweisen benannt, die einen Chatbot „menschlicher" machen konnen.

a) Fragt der Chatbot nach, wie es einem geht, muss dieser damit rechnen, dass der Nutzer `"gut"` oder `"Gut"` antwortet. Einfacher ist es, wenn nicht zwischen Groß- und Kleinschreibung unterschieden werden müsste. Hierfür ist die Methode `toUpperCase()` hilfreich, die bei
https://java-tutorial.org/zeichenketten.html
beschrieben ist.

b) Natürlich kann der Benutzer statt `"gut"` auch `"Mir geht es gut, danke!"` antworten. Eine einfache Methode, das Wort `"gut"` aus einer längeren Antwort herauszufiltern, bietet die Methode `Pattern.matches()`. Importiert man die Pattern-Bibliothek `java.util.regex.Pattern`, dann lässt sich das Vorhandensein des Wortes `"gut"` wie folgt abfragen:

```
String Muster = ".*gut.*";
String Antwort = "Mir geht es gut, danke!";
if (Pattern.matches(Muster, Antwort)) {
    System.out.println("gut ist enthalten");
}
else {
    System.out.println("gut ist nicht enthalten");
}
```

Das Muster `".*"` vor `"gut"` bedeutet, dass alle Zeichen, die vor `"gut"` stehen, ignoriert werden sollen. Das Muster `".*"` nach `"gut"` gilt dementsprechend für alle nachfolgenden Zeichen.

4.3 Testen Sie wie in Aufgabe 2.1 folgende Code-Zeilen als Konsolenprogramme

```
int a = 20;
int b = 20;
String Summe = "";
Summe = Summe + a + b;
```

```
int a = 20;
int b = 20;
String Summe = "";
Summe = Summe + (a + b);
```

```
int a = 20;
String b = "20";
String Summe = "";
Summe = Summe + (a + b);
```

und erläutern Sie das Ergebnis von `System.out.println(Ergebnis)` anhand des Java-Codes.

Di, 3. 14. Stunde, GK Informatik: Schüler sollen Aufgabe 4.1 fertigstellen, danach 4.3 bearbeiten, danach 4.2 bearbeiten, wenn noch Zeit ist.
LG Daniel Gnida