Aufgabe 1: Arten von Kreaturen

Implementieren Sie einen Aufzählungstyp `CreatureType.java`, der als mögliche Datenwerte genau die Werte „Heady“, „Footy“ und „Trunky“ repräsentiert.

Halten Sie bei der Implementierung dieses Aufzählungstyps die Java-Konventionen ein!

Aufgabe 2: Kreaturen

Implementieren Sie im Folgenden eine Klasse `Creature.java`, welche die folgenden Eigenschaften und Verhaltensweisen realisiert.

a) Jede Kreatur umfasst

- einen Information, um welche Art von Kreatur es sich handelt,
- einen Hinweis darauf, ob die Kreatur gerade hungrig ist oder nicht.

  Standardmäßig sei eine neue Kreatur hungrig.

Schützen Sie diese Attribute vor dem unbefugten Zugriff von außen.

b) Definieren Sie eine geeignete Methode, die ermittelt, ob die Kreatur gerade hungrig ist oder nicht. Beachten Sie dabei die Java-Konventionen zur Implementierung von Zugriffsmethoden.

c) Implementieren Sie einen Konstruktor, der als Parameter einen geeigneten Wert für das noch nicht standardmäßig vorbelegte Attribut der Klasse empfängt und diesen Wert entsprechend an das Attributezuweist.

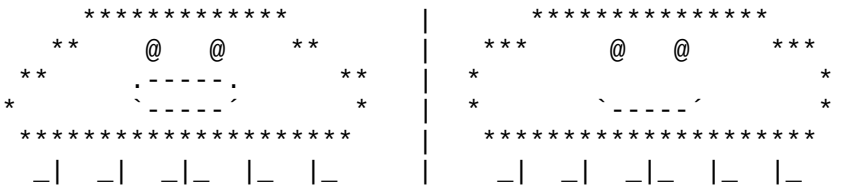
d) Erstellen Sie eine Methode `feedCreature()`, die eine Kreatur füttert. Nach dem Füttern ist die Kreatur nicht mehr hungrig.

e) Schreiben Sie eine Methode `printCreature()`, welche die aktuelle Kreatur graphisch ausgibt. Dazu muss die Methode unterscheiden, von welcher Art die aktuelle Kreatur ist. Verwenden Sie die dazu am besten geeignete Kontrollstruktur von Java.

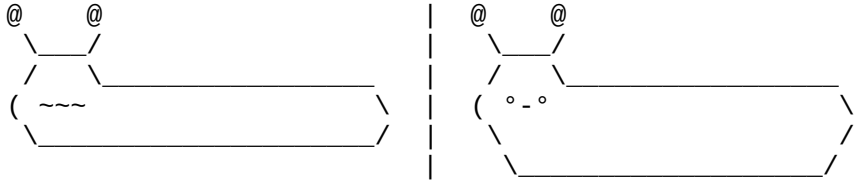
Stützen Sie diese Methode ab auf geeignete, nur intern sichtbare Hilfsmethoden. Jede dieser Hilfsmethoden gibt dabei genau eine Art von Kreatur aus. Sie unterscheidet dabei, ob die Kreatur gerade hungrig ist oder nicht. Hungrige Kreaturen gucken eher grantig und sind dünner als nicht hungrige Kreaturen.

- Die Kreatur Heady könnte beispielsweise so aussehen (links hungrig, rechts nicht hungrig):

- Die Kreatur Footy könnte beispielsweise so aussehen (links hungrig, rechts nicht hungrig):



- Die Kreatur Trunky könnte beispielsweise so aussehen (links hungrig, rechts nicht hungrig):



Aufgabe 3: Stall

Implementieren Sie eine Klasse `Stable.java`, welche die Wohnungen der Kreaturen repräsentiert.

a) Jeder Stall umfasst die folgenden Eigenschaften:

- eine Kreatur (sofern schon eine vorhanden ist; Anfangs ist der Stall unbewohnt!),
- eine Referenz auf den nächsten Stall.

Schützen Sie diese Attribute vor dem unbefugten Zugriff von außen.

Initialisieren Sie diese Attribute explizit mit geeigneten Standardwerten.

b) Implementieren Sie einen Konstruktor, der als Parameter mitgeteilt bekommt, wie viele Einzelwohnungen der Stall am Ende enthalten soll und entsprechend viele Einzelwohnungen anlegt und aneinanderkettet.

c) Schreiben Sie eine Methode `printStable()`, welche den Stall und die darin enthaltenen Kreaturen ausgibt. Eine leere Stallzelle habe dabei die folgende Form:

```
+----------------------------+
|                            |
|                            |
|                            |
|                            |
|                            |
+----------------------------+
```

Wird eine Stallzelle von einer Kreatur bewohnt, so wird diese Kreatur mit ausgegeben. Eine Stallzelle, die von einem hungrigen Footy bewohnt wird, habe beispielsweise die folgende Form:

```
+----------------------------+
|        *************       |
|      **     @   @     **     |
|    **       .-----.       **  |
|   *         `-----´         *  |
|    ********************       |
|     _|  _|  _|_  |_  |_       |
+----------------------------+
```

Hinweis:

Stützen Sie entsprechend die Implementierung dieser Methode ab auf die Methode `printCreature()` der Klasse `Creature.java`. Überarbeiten Sie bei Bedarf die Methode `printCreature()` der Klasse `Creature.java` so, dass der linke und rechte Seitenrand der Stallzelle gemeinsam mit der Kreatur ausgegeben wird.

Lagern Sie evtl. benötigte statische Hilfsmethoden in eine Klasse `PrintHelper.java` aus, die Sie bei Bedarf neu anlegen.

d) Definieren Sie eine Methode `addCreature()`, die als Parameter eine Kreatur empfängt und diese in die erste leere Stallzelle einfügt.

Ist der Stall voll, kann die Kreatur nicht in den Stall einziehen. Geben Sie dann eine geeignete Fehlermeldung aus:

`Stable is full, no room for new creature!`

e) Implementieren Sie abschließend eine Methode `feedCreatures()`, die als Parameter die Anzahl der verfügbaren Futterkübel empfängt. Solange noch Futter da ist, durchläuft die Methode der Reihe nach alle Stallzellen.

Wohnt in einer Stallzelle eine hungrige Kreatur, so wird diese gefüttert. Dabei wird ein Futterkübel verbraucht. Diese Kreatur ist anschließend nicht mehr hungrig. Leere Stallzellen werden nicht gefüttert.

Je nachdem, ob das Futter nicht für alle Kreaturen ausreicht, doch ausreicht oder sogar zuviel ist, wird eine der folgenden Meldungen ausgegeben:

```
Not enough food, some creatures go hungry!
All creatures are fed, no food left!
All creatures are fed, some food is left!
```

Aufgabe 4: Hauptprogramm

Implementieren Sie eine Klasse `Application` zur Steuerung Ihrer Applikation.

f) Erzeugen Sie zunächst einen Stall mit vier Zellen.

g) Fügen Sie der Reihe nach einen Heady und einen Footy in den Stall ein. Geben Sie den Stall anschließend aus.

h) Füttern Sie anschließend die Kreaturen im Stall. Sie haben dazu genau einen Futtereimer zur Verfügung. Geben Sie den Stall anschließend aus.

i) Fügen Sie als weitere Kreaturen einen weiteren Heady und einen Trunky in den Stall ein. Geben Sie den Stall anschließend aus.

j) Füttern Sie anschließend die Kreaturen im Stall. Sie haben dazu zwei Futtereimer zur Verfügung. Geben Sie den Stall anschließend aus.

k) Füttern Sie nun erneut die Kreaturen im Stall. Sie haben dazu drei Futtereimer zur Verfügung. Geben Sie den Stall anschließend aus.