Wetthüpfen

In diesem Arbeitsblatt wird die Klassenbibliothek “hiddenClasses“ verwendet.
Für die Bearbeiteung des Arbeitsblatts wird nur die Dokumentation benötigt. Man erhält sie unter
http://www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/einfuehrungen/hiddenclasses.php
oder
http://www.moodletreff.de/course/view.php?id=38

Für eine Implementation am Rechner gibt es die hiddenClasses.* Bibliothek (rolandHelp.jar) ebenfalls unter
http://www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/einfuehrungen/hiddenclasses.php
oder
http://www.moodletreff.de/course/view.php?id=38

Es soll ein Wetthüpfen zwischen einem Hasen, einem Hund und einem Vogel programmiert werden. Dazu werden die Tiere auf den Bildschirm gezeichnet. Die Startpositionen liegen bei x = 500. Ein Mausklick ist das Startsignal. In einem „Spielzug“ machen alle Tiere (direkt nacheinander) einen Sprung in Richtung x = 0. Die Höhe und Weite jedes Hüpfers ist zufällig. Das Rennen ist beendet, wenn ein Tier, die Linie x=10 erreicht hat. Im Anhang sind Java-Klassen `Hund`, `Wetthuepfen` und `WetthuepfenStart` zu diesem Projekt angegeben. Bei der Klasse `Hund` sind einige Methoden noch unvollständig. Die Klassen `Hase` und `Vogel` werden entsprechend implementiert.

a) Vervollständige den Konstruktor, sowie die Methode `int xPosition()` der Klasse `Hund`. (Im Konstruktor sollen eine Grafik und ein Zeitmanager übernommen und ein Zufallsgenerator (d.h. ein Objekt der Klasse `Random`) erzeugt werden. Die Methode `xPosition()` gibt die Momentane x-Position des Tieres zurück.)

b) Erläutere die vorgegebene objektorientierte Modellierung mit Hilfe eines Entwurfsdiagramms.

c) In den Klassen im Anhang wurde keine Vererbung genutzt. Gesucht ist nun eine bessere Lösung, in der die Klassen `Hase`, `Hund` und `Vogel` Unterklassen einer abstrakten Oberklasse `Tier` sind. Der Programmcode der Unterklassen soll dabei so kurz wie möglich werden. Entwickle eine entsprechende Lösung. (Erkläre kurz, wo hierbei Polymorphie benötigt wird.)

d) Die Methode `void los()` der Klasse `Wetthuepfen` soll modifiziert werden. Verwende statt den Variablen `ottocar, hugo` und `berta` ein Array `t` von Objekten der Klasse `Tier und` Schleifen, um die Methode möglichst kurz zu halten. Außerdem sollen in einem „Spielzug“ nicht alle Tiere springen, sondern jeweils nur eines, das zufällig ausgewählt wird. Weiterhin soll am Ende, wenn also ein Tier die Ziellinie bei x = 10 übersprungen hat, ausgegeben werden, welches Tier das war (Hase, Vogel oder Hund). Erstelle die neue Klasse `Wetthuepfen.`

e) Ordne allen Methoden und Variablen ihre Sichtbarkeitsbereiche zu. Begründe kurz.

f) Stell die Beziehungen der Klassen `Wetthuepfen, Tier, Hund, Vogel` und `Hase` in einem Implementationsdiagramm dar.

g) Dokumentiere die Klasse Tier.

h) Stell die Kommunikation zwischen einem Objekt der Klasse `Wetthuepfen` und dem Objekt `ottocar` beim Aufruf der Methode los() grafisch dar.

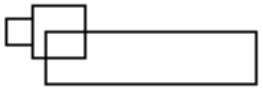
Erweiterungen:

i) Schreibe für die Klasse `Wetthuepfen` eine Methode `void tausche (int a, int b),` mit der im Array t der Tiere das Tier mit dem Index a mit dem mit dem Index b vertauscht werden kann. (Diese Methode könnte z.B. genutzt werden, um die Reihenfolge, in der die Tiere zu sehen sind (oben, unten, mittig), zufällig zu bestimmen, indem zufällig ausgewählte Tiere vertauscht werden. Dies ist jedoch nicht Teil der Aufgabe.)

j) Schreibe eine Klasse `BellenderHund`: Bellende Hunde unterscheiden sich von anderen Hunden nur darin, dass beim Zeichnen zusätzlich an der Position (xPos-50;yPos-50) die Zeichenkette „Wuff“ auf dem Bildschirm erscheint. Die Klasse soll möglichst kurz sein.

k) Nun soll es auch noch Hunde geben, die bellen oder sich auf den Boden legen können. Dies sind dann Objekte der Klasse `DuckenderBellenderHund` mit der Methode

```
void zeichne(){
    g.drawRect(xPos,yPos,80,20);
    g.drawRect(xPos-5,yPos-10,20,20);
    g.drawRect(xPos-15,yPos-5,10,10);
  }
```



Wenn diese Methode ausgeführt wird, sieht man im Fenster einen liegenden Hund, wie in der obigen Abbildung dargestellt.
Die Hunde dieser Klasse sollen ansonsten alles können, was auch Hunde der Klasse `BellenderHund` können und zusätzlich noch die Methoden `void sitz()` und `void auf()` erhalten. Bei Aufruf der Methode `sitz()` soll der Hund auf dem Bildschirm an der aktuellen Position liegend zu sehen sein. (Falls der Hund an der Stelle vorher stehend zu sehen war, muss er vorher noch gelöscht werden.) Ein Aufruf der Methode `auf()` soll entsprechend bewirken, dass auf dem Bildschirm an der Position des Hundes ein stehender Hund und der Schriftzug „Wuff“ zu sehen ist, wie dies bei bellenden Hunden üblich ist. (Stehende Hunde sehen also aus wie bellende Hunde. Beachte dass der Hund möglicherweise vorher an dieser Stelle liegend zu sehen sein könnte.) In der Methode `zeichne()` wird jeweils keine Farbe gesetzt, sondern in der momentanen Farbe (üblicherweise in schwarz) gezeichnet. Die Methoden `void sitz()` und `void auf()`sollen möglichst kurz sein.
Schreibe die Klasse `DuckenderBellenderHund.`

`l)` Für eine andere Anwendung „Karneval“ bekommt die Klasse `Tier` eine neue Variable `vorgaenger` vom Typ `Tier,` sowie die Klassen `Tier`, `Hase`, `Hund` und `Vogel` jeweils eine Methode `yPosition()` und eine Methode `int breite(),` mit der die y-Position und die Breite des Tiers erfragt werden können. (Ein Hase ist 20 Pixel breit, ein Hund 80 Pixel und ein Vogel 35, ein Tier (maximal) 100 Pixel.) Statt einzelnen Tieren soll nun eine Schlange von Tieren über das Fenster hüpfen. Dazu merkt sich jedes Tier seinen Vorgänger. Falls es keinen hat, ist der Vorgänger `null`.

(1) Schreibe für die Klasse `Tier` eine Methode `void stelleDichHinter(Tier vorheriges)` mit dem sich ein Tier von seiner bisherigen Position verschwinden und sich an eine Schlange anstellen kann. . (Dabei soll das Tier sich seinen Vorgänger merken.)

`(2)` Ändere die Methode `void huepfe()` so, dass jeweils zunächst (falls vorhanden) der Vorgänger des Tieres einen Sprung um 10 Pixel nach links ausführt, bevor das Tier selbst um 10 Pixel nach links hüpft. (Die Sprünge können direkt entlang des Bodens ausgeführt werden; man muss das Tier also nicht zwischendurch weiter oben auf dem Bildschirm sehen.)

Materialien:
Klassen `Hund`, `Wetthuepfen` und `WetthuepfenStart,` Dokumentation der Klassen `Grafik, Fenster, Zeitmanager, Maus`