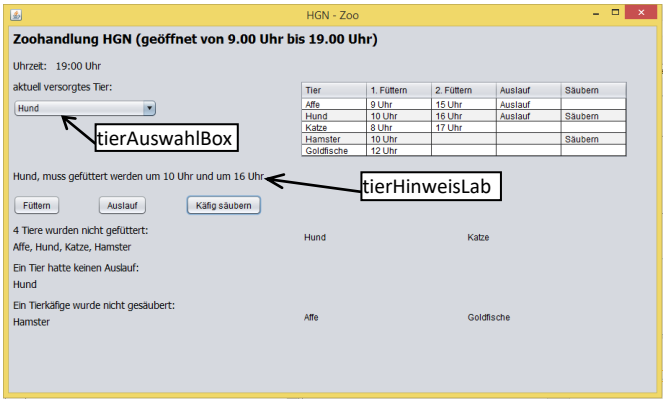
Projekt 6: Die Zoohandlung – Tiere mit verschiedenen Anforderungen

In einer Zoohandlung werden verschiedene Tiere zum Verkauf angeboten. Bei der Haltung bestehen verschiedene Anforderungen: Alle Tiere müssen gefüttert werden, einige Tiere müssen täglich Auslauf haben, bei einigen Tieren müssen regelmäßig die Käfige gesäubert werden. Die folgende Tabelle gibt einen Überblick über die Haltung derTiere:

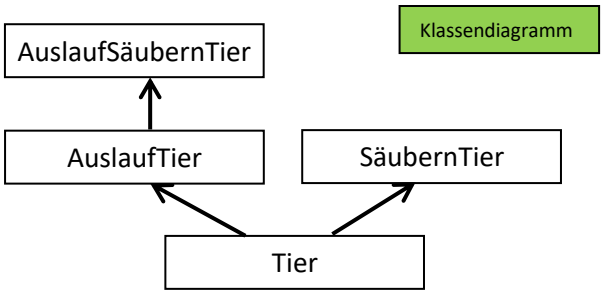
Tier	1. Füttern	2. Füttern	Auslauf	Käfig säubern
Affe	9 Uhr	15 Uhr	täglich	
Hund	10 Uhr	16 Uhr	täglich	täglich
Katze	8 Uhr	17 Uhr		
Hamster	10 Uhr			täglich
Goldfische	12 Uhr			

Mit einem Java-Programm soll der Tagesablauf in der Zoohandlung simuliert werden: Beim Programmstart beginnt die Uhr im Zeitraffer von 9 Uhr (die Zoohandlung öffnet) bis 18 Uhr (die Zoohandlung schließt) zu laufen. Aufgabe ist es, die Tiere zur richtigen Zeit zu füttern und bis zum Abend den Auslauf und das Käfigsäubern jeweils per Mausklick auf einen entsprechenden Button durchzuführen. Am Ende des Tages soll ein Bericht ausgegeben werden, auf dem genau aufgeführt wird, welche Aufgaben erfolgreich ausgeführt wurden.

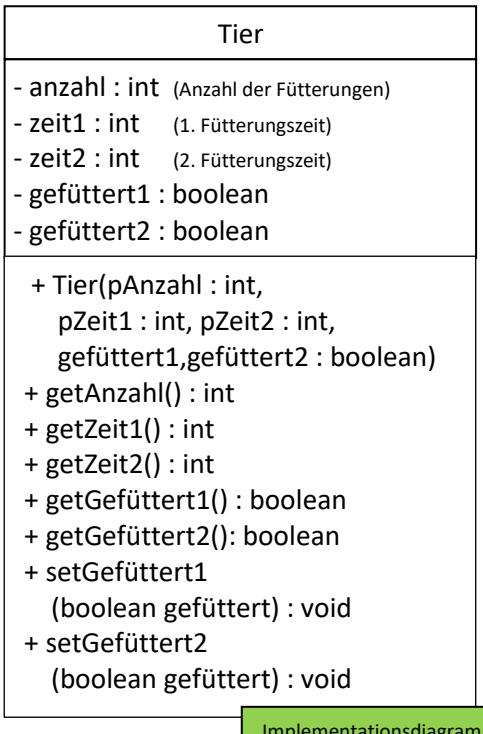


Klassendiagramm

Die Tiere sollen als Objekte der rechts angegebenen Klassen implementiert werden: **AuslaufTier** und **SäubernTier** sollen Tochterklasse der Klasse **Tier** sein. **AuslaufSäubernTier** ist nochmals Tochterklasse von AuslaufTier. Wichtige Eigenschaften für jedes Tier (Objekte Affe, Hund, Katze, Hamster, Goldfische) sollen in der jeweiligen Klasse als Attribute abgespeichert werden: **anzahl** gibt an, ob einmal oder zweimal am Tag gefüttert werden muss, **zeit1** und **zeit2** sind die Fütterzeiten (angegeben wird nur die volle Stunde).
gefüttert1 und **gefüttert2** speichern, ob die Tiere zu den beiden möglichen Zeiten bereits gefüttert wurden oder nicht. Das untere Implemetations-diagramm zeigt alle notwendigen Methoden, mit der die Variablenfelder verwaltet werden können.



Implementationsdiagramm



Implementationsdiagramm für die Klasse Tier

Aufgabe 1: Kopiere das Projekt **Zoohandlung** aus dem Schülerordner in dein persönliches Verzeichnis und öffne das Projekt in Netbeans. Die Klasse Gui ist weitgehend fertig und beinhaltet insbesondere folgende Objekte:

uhrzeitLab	Zeigt die aktuelle Uhrzeit an
tierAuswahlBox	Ermöglicht die Auswahl einer der 5 verschie-denen Tierarten
fütternBut, **auslaufBut**, **säubernBut**	Buttons, mit denen der Benutzer eine Aktion auslösen kann
tierHinweisLab	Hinweise zum aktuell ausgewählten Tier

Aufgabe 2: Die Java-Klasse Tier ist bereits angelegt, jedoch noch fast leer. Implementiere die Attribute, den Konstruktor und die benötigten Methoden, wie im Implementationsdiagramm auf der letzten Seite angegeben. Erzeuge zunächst die fünf Tiere als Objekte der Klasse Tiere, später werden wir uns um die Unterklassen von Tiere können.

```
Tier tier,affe,hund,katze,hamster,goldfische;
```

Die Namen der Tierobjekte wurden bereits deklariert, bevor du sie im nächsten Schritt mit dem new-Befehl erzeugen musst: dies hat den Vorteil, dass man in der Klasse **Gui** überall auf die Tierobjekte zugreifen kann.

Aufgabe 3: Wenn man auf ein Tier in der Comboauswahlbox klickt, soll(en) im **tierHinweis-Lab** sofort die Fütterzeit(en) angezeigt werden. Die Tiere Affe und Hund sind bereits implementiert, ergänze die weiteren Tiere.

```
private void TierAuswahlBoxItemStateChanged(java.awt.event.
    aktuelleTierNummer = TierAuswahlBox.getSelectedIndex();
    switch (aktuelleTierNummer){
        case 0:
            tier = affe;
            tierText = "Affe";
```

...

Timerereignisse

Aufgabe 4: Wenn man auf den Button **fütternBut** klickt, muss Java in der Event-Methode **Fuettern-ButMouseClicked** überprüfen, ob das Tier zur richtigen Zeit gefüttert wird. Dazu müssen wir zunächst einen Timer starten, der die Zeit (im Zeitraffer) darstellt:

1.) Der Timer und drei Variablen zum Darstellen der aktuellen Uhrzeit werden zunächst als Variablen-felder von **Gui** deklariert. (Schon fertig !)

2.) Im Konstruktor von Gui **publuc Gui()** wird der Timer erzeugt und gestartet. (Schon fertig !)

3.) Danach wird die Methode **actionPerformed** eingefügt: Die Variable **zähler** wird alle 10 Millisekunden um eins hochgezählt. Erreicht der Zähler den Wert 10, wird die Minutenzahl um eins hochgezählt, analog wird die Stundenzahl erhöht. Mit den beiden Variablen **minuten** und **stunden** kann die Uhrzeit im **uhrzeitLab** angezeigt werden. Um 18 Uhr wird der Timer angehalten. An dieser Stelle wird später die Auswertung angezeigt: Du musst du entsprechenden Zeilen aus dem Kopier-fenster Rechts nur in dein Programm kopieren.

4.) **ActionListener** und **ActionEvent** müssen als Methoden für die Abfrage des Timers importiert werden, es kann sein, dass dies Netbeans automatisch erledigt, ansonsten müssen diese beiden Zeilen ganz zu Beginn des Programms geschrieben werden. Probiere aus, ob die Uhrzeit jetzt richtig angezeigt wird. (Schon fertig !)

```
public class Gui extends javax.swing.JFrame

    int zaehler = 0;  int minuten = 0;
    int stunden = 8;
    Timer time;
```

neue Variablenfelder

```
public Gui() {
    initComponents();
    time = new Timer(10,this);
    time.start();
```

Im Konstruktor den Timer starten

```
public void actionPerformed(ActionEvent e){
    DecimalFormat f1 = new DecimalFormat("#00");
    zaehler++;
    if (zaehler == 10){
        minuten++;
        zaehler = 0;
        if (minuten == 60){
            stunden++;
            minuten = 0;
        }
    }
    uhrzeitLab.setText(String.valueOf(stunden) + ":" +
        f1.format(minuten) + " Uhr");
    if (stunden == 18){
        time.stop();
        //auswertung();
    }
}
```

Methode actionPerformed reagiert auf das Timerereignis

Kopiere diesen Abschnitt in dein Programm!

```
public void actionPerformed(ActionEvent e){
        DecimalFormat f1 = new
DecimalFormat("#00");
    zaehler++;
    if (zaehler == 10){
      minuten++;
      zaehler = 0;
      if (minuten == 60){
        stunden++;
        minuten = 0;
      }
    }
    uhrzeitLab.setText(String.valueOf(stunden) + ":" +
        f1.format(minuten) + " Uhr");

    if (stunden == 18){
      time.stop();
      // auswertung();
    }
  }
```

Aufgabe 5: Nach Klick auf den **fütternBut** soll in einem **ergebnisLab** unterhalb des Buttons eine Nachricht angezeigt werden: Zunächst muss die aktuelle Uhrzeit mit der Fütterzeit für das aktuelle Tier (**tier**, vgl. **tierAuswahlBoxItemChanged**) verglichen werden. Ist die Fütterzeit korrekt wird die Variable **gefüttert1** (bzw. **gefüttert2**, wenn die zweite Fütterzeit aktiv ist) in der Klasse Tier gesetzt. Anschließend wird der passende Text gesetzt (vgl. Tabelle rechts). Die Variable **tierText** (vgl. **tierAuswahlBoxItemChanged** auf dieser Seite oben) vereinfacht die Ausgabe, da dieser unabhängig vom gewählten Tier gesetzt werden kann. Du brauchst nur noch den Fall zu ergänzen, dass zur falschen Zeit gefüttert wurde.

```
package my.Zoohandlung;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

notwendige Importe

zu Aufgabe 5:

Richtige Fütterzeit	Affe ist satt.
Fichtige Fütterzeit	Der Affe wurde zur falschen Zeit gefüttert.

Im **ergebnisLab** wird der passende Text gesetzt, nachdem der Benutzer ein Tier füttern möchte.

Aufgabe 6: Nach Ablauf des Tages, d. h. sobald die Uhr 18 Uhr anzeigt, soll eine Auswertung über alle Tätigkeiten in der Zoohandlung erfolgen, die der Benutzer im Laufe des Tages ausführen sollte. Zunächst geht es um das Füttern.
a) Tier für Tier wird jetzt abgefragt und nachgesehen ob die Fütterung (ggf. beide Fütterungen) durchgeführt wurden. Am einfachsten ist es, in einer String-Variable, die zu Beginn leer ist, alle Tiere nacheinander aufzulisten, bei der eine Fütterung fehlt. Mit der Methode **equals** kann man bei String-Variablen abfragen, ob zwei Strings gleich sind. Überlege, warum hier der bisherige Text mit einem leeren String (““) verglichen wird. In einer weiteren Variablen **nichtGefüttert** werden die nicht gefütterten Tiere gezählt. Für Affe und Hund ist die Auswertung bereits implementiert, ergänze ähnliches für die restliichen Tiere.

```
private void auswertung(){
    int nichtGefuettert = 0;
    String antwortText = "";
    if ((affe.gefüttert1 == false) || (affe.gefüttert2 == false)){
      antwortText = antwortText + "Affe";
      nichtGefuettert++;
    }
    if ((hund.gefüttert1 == false) || (hund.gefüttert2 == false)){
      if (!antwortText.equals ("")){antwortText = antwortText + ", ";}
```

Die Methode **auswertung()**: der String **antwortText** soll zum Schluss alle Tiere enthalten, die nicht gefüttert wurden. Die Tiere werden mit einem Komma jeweils voneinander getrennt.

Noch ein Tipp: Damit man die Methode **auswertung** gut testen kann, ist ein Button **auswertungAnzeigen-But** in der Gui-Oberfläche integriert. Beim Klick auf den Button wird der Timer angehalten (**time.stopp();**) und die Auswertung angezeigt.

b) Die bisherige Auswertung bei den einzelnen Tieren kann bei der abschließenden Ausgabe gut benutzt werden: jenachdem ob kein, ein oder mehrere Tiere nicht gefüttert wurden, sollte eine grammatikalisch richtige Aussage gemacht werden, es werden zwei Labels dazu benötigt: in einem wird angegeben, wie viele Tiere nicht gefüttert wurden, ds andere enthält ggf. die nichtgefütterten Tiere (z. B. „Affe und Hund wurden nicht gefüttert.“). Dieser Teil ist bereits implementiert.

Aufgabe 7: Jetzt werden die Unterklassen **AuslaufTier**, **SäubernTier** und **AuslaufSäubernTier** eingefügt. Jeweils eine neue boolsche Variable wird als Attribut eingesetzt: **auslauf** beim **AuslaufTier** bzw. **gesäubert** beim **SäubernTier** speichert, ob der Auslauf bzw. die Käfigsäuberung ausgeführt wurde. Überlege jeweils, welche Methoden ergänzt werden müssen. Das **AuslaufSäubernTier** erbt vom **AuslaufTier** und muss sowohl den Auslauf als auch das Käfigsäubern speichern können. Zwei neue Buttons **auslaufBut** und **säubernBut** werden benutzt, um dem Benutzer die entsprechenden Aktivitäten zu ermöglichen. Jeweils zwei neue **ergebnisLabs** sind für die Ausgabe eines Textes vorhanden.
Die entsprechenden Event-Methoden sind bereits in Kommentarzeichen (/* bis */) vorhanden, du musst nur die Kommentierungszeichen wegnehmen. In der Methode **auswertung** muss im letzten Teil noch einiges ergänzt werden, damit die Mitteilungen vollständig sind.
Wann der Benutzer den Auslauf durchführt oder den Käfig säubert, ist ihm freigestellt. Teste das Programm anschließend gründlich aus, versuche die Zoohandlung einen Tag fehlerfrei zu führen.

Tabellen verwenden

Zusatzaufgabe 1: a) In der Grafik zu Beginn dieses Kapitels, siehst du, wie man mit einer Tabelle (**jTable**) mehr Übersicht in die einzlnen Tätigkeiten bringen kann. Auf der rechten Seite siehst du den Code, den du über die Funktion **Customize Code** einfügen musst (vgl. Kapitel 2). Mache die Tabelle sichtbar.
b) Wenn ein Tier gefüttert, ausgeführt oder der Käfig gesäubert wurde, soll der entsprechende Text in der Tabellenzelle verschwinden. Mir der folgenden Methode kann man den Inhalt einer Zelle der Tabelle zur Laufzeit verändern:
Tabelle.setValueAt (text, ypos,xpos);
xpos und ypos sind dabei x- und y-Koordinaten der Zelle (also Spalten- und Zeilennummer), wobei die Zählung jeweils mit 0 beginnt.
Am Ende des Konmstruktors der Klasse Gui musst du noch zwei zeilen einfügen, die die Tabelle sichtbarmachen und die größe der Schriftart in der Überschriftszeile steuern (s. Rechts).

```
default code        Tabelle = new javax.swing.JTable();

custom property     Tabelle.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {"Affe", "9 Uhr", "15 Uhr", "Auslauf", null},
            {"Hund", "10 Uhr", "16 Uhr", "Auslauf", "Säubern"},
            {"Katze", "8 Uhr", "17 Uhr", null, null},
            {"Hamster", "10 Uhr", null, null, "Säubern"},
            {"Goldfische", "12 Uhr", null, null, null}
        },
        new String [] {
            "Tier", "1. Füttern", "2. Füttern", "Auslauf","Säubern"
        }
```

{"Affe", "9 Uhr", "15 Uhr", "Auslauf", null},
{"Hund", "10 Uhr", "16 Uhr", "Auslauf", "Säubern"},
{"Katze", "8 Uhr", "17 Uhr", null, null},
{"Hamster", "10 Uhr", null, null, "Säubern"},
{"Goldfische", "12 Uhr", null, null, null}

Inhalte können kopiert werden!

```
Tabelle.setShowGrid (true);
Tabelle.getTableHeader().setFont(new
Font("Tahoma", 0, 18));
```