Einführung in die objektorientierte Programmierung (OOP) mit Objekt-PAL in Paradox -- Vorbemerkung

Robert Krell

Die folgenden Seiten skizzieren Teile eines etwa 5-monatigen Grundkurses aus dem Jahr 1999 (2 Stdn/Woche) zur Einführung in die Algorithmik mit visuellen Objekten und wurden (mangels Buch) als unterrichtsbegleitendes Skript ausgeteilt. Inzwischen (2003) benutze ich Java (lieber als Delphi). Trotzdem waren in diesem Kurs viel auch heute noch gültige schöne Ansätze gut zu realisieren!
Die Schülerinnen und Schüler -- hier angehende Biologisch-Technische-Assistentinnen und -Assistenten -- hatten zuvor bereits Standardsoftware kennen- und speziell für ihre Belange einsetzen und anwenden gelernt. Als Tabellenkalkulation war Excel, als Datenbanksoftware Paradox benutzt worden. Laut Lehrplan stand für das zweite Ausbildungsjahr noch eine Einführung in das eigene Programmieren (über gelegentliche Makros in den Standardprogrammen hinaus) sowie ein Einblick in die Computerhardware an.
Informatik (bzw. „Technische Kommunikation“, wie es offiziell heißt), ist Fach der Berufsabschlussprüfung. Das Genehmigungsverfahren für die Prüfungsvorschläge durch die Schulaufsicht sowie der Prüfungsablauf sind mit der schriftlichen Prüfung im Informatik-Grundkurs des Gymnasiums vergleichbar.

Längere Überlegungen, entweder
- Algorithmik mit Niki einzuführen und auf dann auf einfache Pascal-Programme überzugehen,
- oder eine Sprache wie Profan für Windows-Applikationen zu verwenden,
- oder (sofern in unserem Schulnetz mit 486er-Rechner möglich) Delphi zu verwenden,
- oder die mit Paradox ausgelieferte Sprache Objekt-PAL zu nutzen,

führten schließlich zur Entscheidung für die letztgenannte Möglichkeit. Gründe dafür waren u.a.
- die Möglichkeit, leicht Programme mit moderner, grafischer Oberfläche zu erstellen
- Objektorientiertheit und die Tatsache, dass es bei Objekt-PAL keinen zusammenhängenden Quelltext (wie bei Delphi) mit dem automatisch erzeugten Code gibt, sondern nur der selbst eingetippte Programmtext jeweils in überschaubaren Portionen lokal am Objekt hängt
- die geringen Hardware-Anforderungen und der niedrige Preis (bei Pearl oder SMM gibt's CDs mit dem völlig ausreichenden Paradox 4.5 für rund 4,- DM; die Paradox-Versionen 5 [letzte 16-Bit-Version] und die Version 7 [für Win95/98] werden zusammen auf einer CD für ca. 8,- DM an Schüler verkauft!)

Nachteilig ist, dass keine allein stehenden exe-Dateien erzeugt werden können, sondern die erzeugten Programme immer nur unter Paradox laufen.

Der Kurs selbst hat die Schülerinnen und Schüler und mich restlos von der Richtigkeit der getroffenen Entscheidung überzeugt: Obwohl auf Seiten der Schülerinnen und Schüler auf keinerlei Vorkenntnisse in der Algorithmik oder Erfahrungen mit anderen Programmiersprachen zurückgegriffen werden konnte, machte das gewählte Vorgehen Spaß und lieferte einen gelungen Einstieg ins eigene Programmieren.

R. Krell, Lotte-Wicke-Weg 12, 40627 Düsseldorf

http://www.r-krell.de
e-Mail: mail@r-krell.de oder krell@web.de

Über Rückmeldungen und Kommentare (schriftl. oder per e-mail) freue ich mich!
August 1999 (neu gepackt: April 2003)

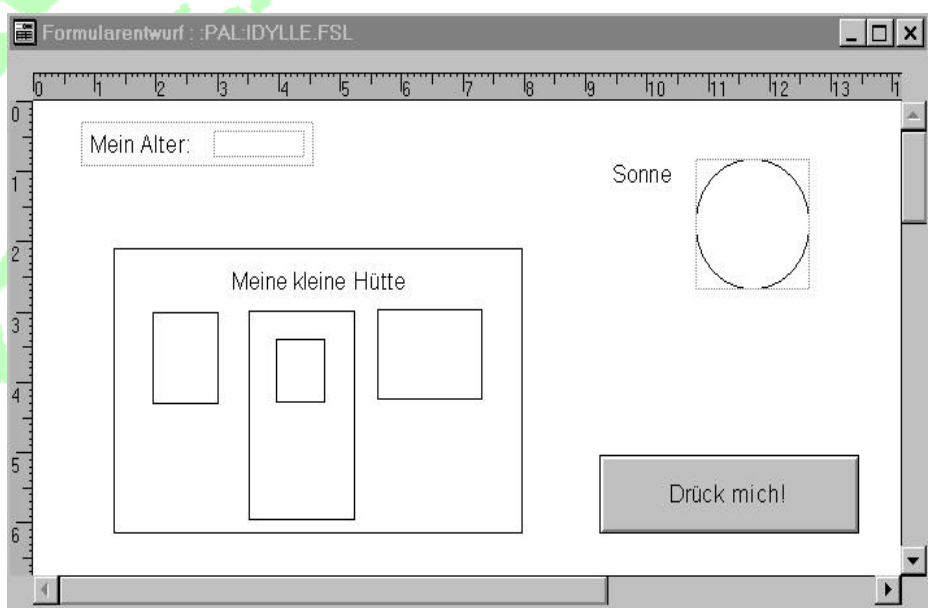
Einführung in die objektorientierte Programmierung (OOP) mit Objekt-PAL in Paradox

In den vergangenen Jahren wurden Programmiersprachen neu- und weiterentwickelt. Es galt, die Nachteile der frühen Hochsprachen (wie Cobol, Fortran und Basic) zu überwinden, die sich bei den immer größeren Programmen immer deutlicher zeigten. Nach den blockstrukturierten Sprachen (wie z.B. Pascal, Modula, C und späteren, weiterentwickelten Basic-Versionen) wurden in letzter Zeit objektorientierte Sprachen entwickelt (z.B. Smalltalk, Eiffel, Oberon) bzw. bekannte Sprachen auf Objekte erweitert (Objekt-Pascal, Delphi, C++). Bei objektorientierten Sprachen schafft der Programmierer Objekte, verleiht ihnen Eigenschaften, und haucht ihnen mit Methoden Leben ein. Die Objekte reagieren nach dem Programmstart dann praktisch selbständig auf verschiedene Ereignisse -- so, wie es im Programmtext der Methoden festgelegt wurde. Der Programmtext ist dabei meist nicht mehr zusammenhängend, sondern jedes Objekt und jede Methode erhält und enthält eigene Anweisungen. Die Programme werden dadurch übersichtlicher, klarer strukturiert und verständlicher. Das Programm muß nicht in einem Guß geschrieben werden, sondern Objekte können einzeln hinzugefügt und verbessert bzw. aus früheren Programmierarbeiten übernommen werden.

Mit Objekt-PAL (PAL=Paradox Application Language) enthält Paradox eine sehr mächtige und zukunftsweisende Sprache, die vieles von dem bereits verwirklicht hat, was selbst neuesten Versionen anderer Sprachen noch fehlt (aber für kommende Erweiterungen angekündigt ist). Viele Konzepte z.B. von Delphi wurden bereits vorweggenommen. Darüber hinaus kann Objekt-PAL natürlich mit Paradox-Datenbanken arbeiten, was ein weiterer Vorteil gegenüber "normalen" Programmiersprachen ist.

Grundlage (und Grundobjekt!) von Objekt-PAL ist ein Formular.

Weitere Objekte sind z.B. alle Elemente, die auf dem Bildschirm zu sehen sind bzw. in das Formular eingefügt werden. Die Werkzeugleiste bietet hier u.a. Rechtecke, Ellipsen, Textfelder, Schalter u.ä. an.



Übung 1a: *Öffnen Sie ein neues Formular (in Paradox mit Datei | Neu | Formular und 2 x OK) und zeichnen Sie die abgebildete Landschaft.*

Wählen Sie dazu aus der Werkzeugleiste (bei Paradox auch "Schalterleiste" genannt) passende Objekte und fügen Sie diese neben- und z.T. auch übereinander in das Formular (Mausklick auf den Knopf in der Werkzeugleiste, dann Maus aufs Formular -- linke Maustaste drücken, Objekt aufziehen und linke Maustaste loslassen). Das mit "Mein Alter" beschriftete Objekt oben links ist übrigens ein Feld und wurde fertig aus der Werkezugleiste übernommen. Nur auf die die vorgegebene Bezeichnung "LABEL" muß so oft geklickt werden, bis sie grau ist. Dann kann der Text verändert werden!

Betrachten Sie die Objekthierarchie (Menü Formular | Objekthierarchie oder Knopf rechts auf der Werkzeugleiste) und benutzen Sie die Pfeiltasten zum Bewegen innerhalb der Hierarchie!

Wählen Sie außerdem durch Anklicken im Fenster Objekthierarchie oder direkt auf dem Formular ein Objekt aus und drücken Sie die rechte Maustaste zum Inspizieren bzw. Ändern der Eigenschaften dieses Objekts. Ersetzen Sie zunächst die von Paradox automatisch vergebenen Namen wie #Rechteck5 oder #Ellipse3 durch aussagekräftige Namen wie Hütte (Tür, Fenster,..), Sonne (Kugel, Ballon...). Verändern Sie außerdem die Farben der Objekte. Färben Sie die Sonne gelb und beachten Sie beim Färben des Hauses die Auswirkungen auf nach- (Tür, Fenster, Türfenster) bzw. übergeordnete Objekte (Formular), um das wichtige Konzept der "Vererbung" in den objektorientierten Sprachen kennen zu lernen.

Die Eigenschaften der Objekte können Sie -- wie eben erfolgreich ausprobiert -- interaktiv (mit deutschen Bezeichnungen) festlegen. Bisher erscheint die Landschaft jedoch nur statisch: Nach Anklicken des Blitzsymbols oder Druck auf F8 läßt sich zwar die Taste (der Schalter) unten rechts mit der Maus drücken -- sonst passiert aber nichts. Das soll sich jetzt ändern: Eine Methode soll zur Laufzeit Objekteigenschaften ändern. So soll bei Druck auf den Schalter “Drück mich” die Sonne rot werden und der Benutzer über das Hereinbrechen des Abends informiert werden.

Übung 1b: *Öffnen Sie die in Übung 1 erstellte Landschaft im Entwurfsmodus, drücken Sie mit der rechten Maustaste auf den Schalter “Drück mich” und wählen aus dem PopUp-Menü “Methoden...”. Doppelklicken Sie auf* pushbutton *und fügen Sie zwischen den beiden vorgegebenen Zeilen zwei Zeilen Programmtext ein! Starten Sie dann das Programm und prüfen, wie es auf das Ereignis (engl. event) 'Tastendruck' (pushbutton) reagiert!*

```
method pushButton(var eventInfo Event)
  Sonne.Color = Red
  msgInfo ("Hinweis", "Es wird Abend!")
endMethod
```

Um Programme auch international les- und austauschbar zu halten, müssen im Programmtext die Eigenschaften der Sonne und aller anderen Objekte leider englisch benannt werden: Statt Farbe muß -- in der OOP-typischen Punktschreibweise *Objektname.Eigenschaft* -- jetzt `Sonne.Color = Red` gesetzt werden. Die Tabelle nennt einige Eigenschaften und ihre Werte. Je nach Objekttyp sind nicht alle Eigenschaften verfügbar!

im Menü	Eigenschaft	Mögliche Werte
Farbe	.Color	Black, Blue, Brown, DarkBlue, DarkCyan (=türkis), DarkGray, DarkGreen, DarkMagenta (=lila), DarkRed, Gray, Green, LightBlue, Magenta, Red, Translucent, Transparent, White, Yellow
Rahmen | Art	.Frame.Style	NoFrame, DashDotFrame, Inside3DFrame, Outside3DFrame, SolidFrame, ShadowFrame,...
Muster | Art	.Pattern.Style	No, BricksPattern, CrosshatchPattern, DiagonalCrosshatch-Pattern, DottedLinePattern, EmptyPattern, LightDotPattern, ZigZagPattern,...
Schrift | Farbe	.Font.Color	(siehe Color)
Schrift | Größe	.Font.Size	8, 9, 10, 11, 12, 16, 20, ...
Schrift | Art	.Font.Style	FontAttribBold, FontAttribItalic, FontAttribNormal,..
Laufzeit | Sichtbar	.Visible	Yes, No

Um beispielsweise die Hütte mit einem blauen Schattenrahmen zu umgeben, kann `Hütte.Frame.Color = Blue` und `Hütte.Frame.Style = ShadowFrame` programmiert werden!
Soll der über der Tür stehende Text "Meine kleine Hütte" kursiv, also in Schrägschrift geschrieben werden, hilft `Überschrift.Font.Style = FontAttribItalic`. Über weitere Schriftarten gibt die im Entwurfsmodus über Hilfe | ObjectPAL erreichbare OnLine-Hilfe mit Suchen nach Font-Attributes Auskunft (genauso hilft die Suche nach Colors oder PatternStyles). Allerdings muß man schon die ersten Buchstaben der englischen Bezeichnung einer Eigenschaft wissen, um in der Hilfe mögliche Werte finden zu können.

Übung 1c: *Erweitern Sie die Landschaft noch um die Strichzeichnung einer Katze aus zwei braun gefüllten Ellipsen und vielen Linien. Diese Katze soll nachts (abends) völlig verschwinden. Damit nicht alle Komponenten einzeln auf unsichtbar gesetzt werden müssen, empfiehlt es sich, ein Rechteck um die Katze zu zeichnen (ein solches Rechteck heißt Container-Objekt. Welche Eigenschaft von Objekten wird hier ausgenutzt?). Nach der Testphase kann der Rechteck-Rahmen im Entwurfsmodus nach Anklicken mit der rechten Maustaste auf die Hintergrundfarbe weiß verändert werden, damit das Rechteck tags nicht auffällt (warum wird nicht den Katzencontainer einfach unsichtbar machen?)*

Bei den Übungen 1b/c stört sicher noch, daß auf den einmal durch Knopfdruck ausgelösten Abend nicht wieder der Tag folgt. Deshalb soll ein Mausklick auf die Sonne (nicht auf den Schalter) diese wieder gelb färben und die Katze wieder erscheinen lassen.

Übung 1d: *Schreiben Sie ähnlich wie in Übung 1b&c eine Methode* `Sonne.mouseClick(var eventInfo MouseEvent)`*, die eine Nachricht "Es wird hell!" auf dem Bildschirm ausgibt und dann die Sonne (wieder) gelb färbt und die Katze (wieder) sichtbar macht.*

Übung 1e: *Füllen Sie die Hütte beim Mausklick auf die Tür mit einem Backstein-Muster!*
(Hinweis: Das Muster scheint auch durch Türen und Fenster, die als durchsichtige Objekte auf dem gemusterten Haus liegen. Durch Wahl einer (undurchsichtigen) Farbe für Türen und Fenster kann ein realistischerer Eindruck erreicht werden!)

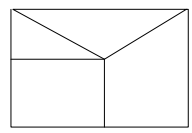
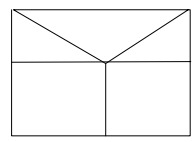
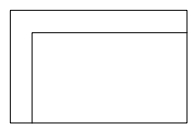
Hier wird deutlich, wie der Programmtext auf die verschiedenen Objekte bzw. ihre Methoden verteilt ist -- wobei eine Methode vom Objekt X durchaus die fremden Objekte Y und Z beeinflussen kann. Gleichzeitig läßt sich aus der Reihenfolge der beiden Zeilen in den Übungen 1b und 1d -- einmal `msginfo`.. vor, einmal nach der Farbänderung -- erkennen, daß nach Eintritt eines Ereignisses (wie Knopfdruck oder Mausklick) die in der reagierenden Methode aufgeführten Anweisungen und Befehle in der Reihenfolge von oben nach unten hintereinander ausgeführt werden (Sequenz). Durch spezielle Kontrollstrukturen wie Verzweigung (Selektion) und Wiederholung (Repetition) läßt sich aber von der linearen Reihenfolge bei der Befehlsausführung abweichen (s.u.).

Die bisherigen Übungen sind -- trotz aller Schönheit -- noch nicht vollständig zufriedenstellend: Wird mehrfach auf die Sonne geklickt oder auf den Schalter gedrückt, erscheint eine Änderungsmeldung, ohne daß sich etwas tut. Das soll anders werden: Weil der Schalter "Drück mich" so einladend wirkt, soll jeder Knopfdruck die Tageszeit wechseln. Auf die Mausklick-Methode der Sonne kann dann verzichtet werden.

Übung 1f: *Kopieren Sie den Programmtext von der Sonne zum Schalter (s.u.) und testen Sie!*

Dieser erste Versuch in 1f, einfach den Programmtext der Methode `Sonne.mouseClick` mit Bearbeiten | Kopieren und Bearbeiten | Einfügen (oder mit den entsprechenden Knöpfen in der Werkzeugleiste) einfach unter den Text in der Methode `DrückMichKnopf.PushButton` zu setzen, bringt nicht den gewünschten Erfolg: Auf die Nacht folgt sofort wieder der Tag. Es wäre

wünschenswert, wenn die Methode über so viel "Intelligenz" verfügen könnte, daß sie den gegenwärtigen Zustand erkennt (z.B. an der Sonnenfarbe) und bei Tag auf die Nacht bzw. bei Nacht auf den Tag umschaltet. Je nach vorherigem Zustand muß also mal der erste, mal der zweite Teil des zusammenkopierten Programmtextes ausgeführt werden. Einen Wippschalter mit zwei verschiedenen Schaltzuständen gibt es nämlich in Objekt-PAL nicht. Abhilfe schafft eine Kontrollstruktur. Einige Möglichkeiten sind in der Tabelle genannt:

Kontrollstruktur		Objekt-PAL	Struktogramm
Verzweigung	einseitige Verzweigung	`if` *Bedingung* `then` *Anweisung1* *(Anweisung2, ...)* `endif`	
	zweiseitige Verzweigung	`if` *Bedingung* `then` *Anweisung1 (Anweisung2, ...)* `else` *AnweisungA (AnweisungB, ...)* `endif`	
	mehrseitige Verzweigung	`switch` ... `case` ... `endswitch`	
Wiederholung	Wdh. mit vorgeschalteter Kontrolle	`while` *Bedingung* *Anweisung1 (Anweisung2, ..)* `endwhile`	

	Zählschleife	`for` *Zähler* `from` *Startwert* `to` *Ende* `[step` *Schrittweite*`]`
		Anweisung1 (Anweisung2, ..)
		`endfor`	

Erläuterungen, Querverweise und Beispiele -- allerdings mit vielen hier noch nicht behandelten Befehlen -- hält die Hilfe unter ObjectPAL bei Suchen nach einem der Schlüsselwörter wie `if`, `switch`, `for` oder `while` bereit. Offensichtlich sind Verzweigungen unserem Problem angemessen:

Übung 1g: *Fügen Sie in den schon in Übung 1f zusam-mengeschobenen Text beim DrückMichKnopf zwei Verzweigungen ein. Erleichtern Sie die Lesbarkeit durch Einrücken und Kommentare in {} oder nach ;*
Testen Sie die Wirkungsweise und erklären Sie das (unerwartete?) Verhalten durch Nachvollziehen des Programmablaufs!

```
method pushButton(var eventInfo Event)
  if Sonne.Color = Yellow {bisher ist Tag}
    then {dann soll es Nacht werden!}
      Sonne.Color = Red
      msgInfo ("Hinweis", "Es wird Abend!")
      KatzenContainer.Visible = No
  endif
  if Sonne.Color = Red {bisher war Nacht}
    then {dann soll es Tag werden}
      msgInfo ("Und jetzt..", "..wird es hell!")
      Sonne.Color = Yellow
      KatzenContainer.Visible = Yes
  endif
endMethod
```

Haben Sie die Erklärung für das merkwürdige Verhalten gefunden, daß sich gegenüber 1f gar nicht

verbessert hat? Dann überlegen Sie bitte, wie/ob eine zweiseitige Verzweigung (anstelle zweier einseitiger Verzweigungen) helfen kann!

Übung 1h: *Strukturieren Sie den Programmtext beim DrückMich-Knopf wie angegeben! Warum ist die Wirkung anders als bei 1g ?*

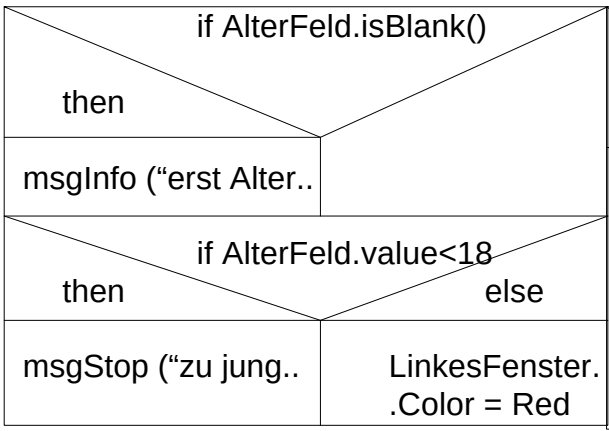
```
method pushButton(var eventInfo Event)
  if Sonne.Color = Yellow {bisher ist Tag}
    then {dann soll es Nacht werden!}
      Sonne.Color = Red
      msgInfo ("Hinweis", "Es wird Abend!")
      KatzenContainer.Visible = No
    else {war noch kein Tag, so soll es Tag werden}
      msgInfo ("Und jetzt..", "..wird es hell!")
      Sonne.Color = Yellow
      KatzenContainer.Visible = Yes
  endif
endMethod
```

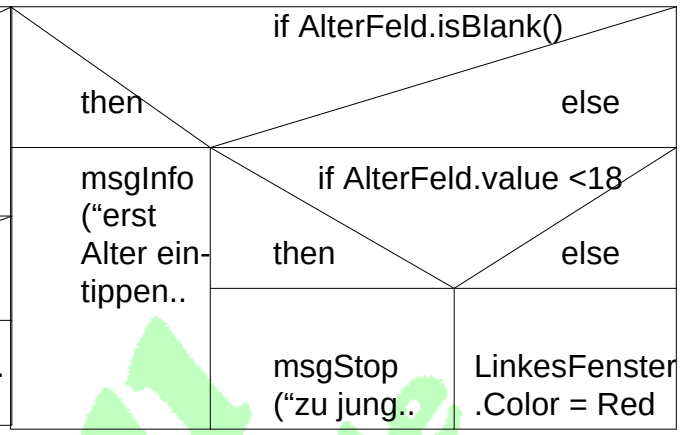
Kontrollstrukturen können übrigens auch verschachtelt werden. Dies soll an folgendem Beispiel gezeigt werden: Bisher wurde in unserer Landschaft nicht darauf geachtet, ob ein Alter (und wenn, welches) eingegeben wurde. Wenn versucht wird, durch Mausklick auf das linke Fenster der Hütte ins Innere zu blicken, so soll dieser Blick jugendlichen Zuschauern verwehrt bleiben. Genauer: falls noch kein Alter oder ein Alter unter 18 eingegeben wurde, sollen entsprechende Warnungen auf dem Bildschirm erscheinen.

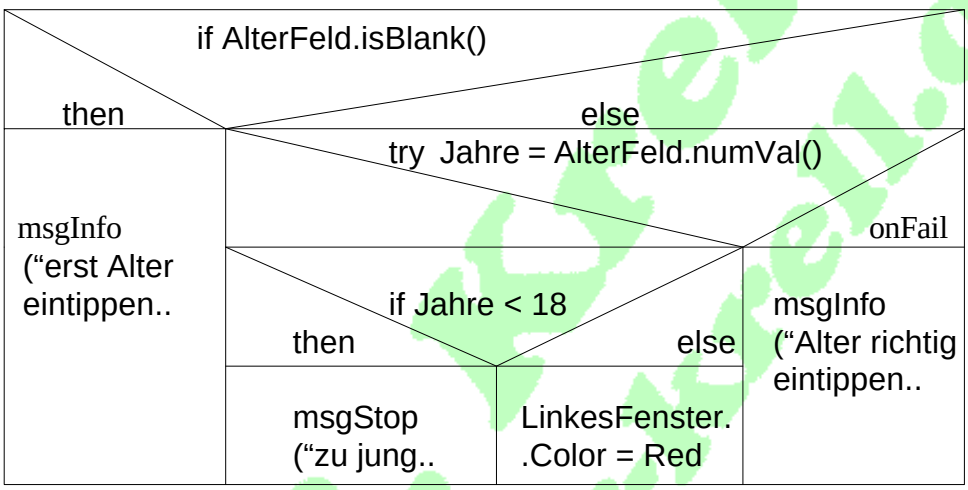
Übung 1i: *Fügen Sie beim Objekt LinkesFenster die Methode mouseclick wie skizziert ein. Dabei bezeichnet AlterFeld das gesamte, mit "Mein Alter" beschriftete Objekt oben links in der Landschaft (Namen notfalls entsprechend ändern). Überlegen Sie selbst, wo im Text noch die zweite Abfrage* `if AlterFeld.value < 18 then msgStop ("Holla", "Noch reichlich jung` *.. untergebracht werden muß! Oder wäre* `if AlterFeld.value < "18"` *.. besser? Experimentieren Sie mit verschiedenen Möglichkeiten und entwerfen Sie jeweils ein Struktogramm (damit Sie als Mensch den Durchblick behalten)! Lassen Sie bei ausreichendem Alter das Fenster nach dem Mausklick rot erscheinen.*
Was passiert, wenn Sie als Alter z.B. '2a1' eingeben? `try` *(s. Hilfe) kann zur Ausgabe einer Fehlermeldung benutzt werden!*

```
method mouseClick (var eventInfo MouseEvent)
  if Alterfeld.isblank() {noch nichts eingetippt}
    then
      msgInfo ("Ohne Alter", "Bitte eingeben")
  endif
endMethod
```

Im folgenden sind einige mögliche Struktogramme aufgezeigt. Wie arbeiten die dargestellten Programme? Und wie müssen sie in Objekt-PAL geschrieben werden? (Weil die Kontrollstrukturen hier durch Rahmen begrenzt sind, läßt man `endIf` bzw. `endTry` im Struktogramm weg. Ebenso fehlen die Überschriften bei `msgInfo`). Beim dritten Diagramm fällt auf, daß jeweils außen links und außen rechts ähnliche Meldungen ausgegeben werden. Läßt sich da etwas zusammenfassen oder vereinfachen?







Zu Beginn der Methode muß Jahre als Zahlen-Variable definiert werden:

```
var
  Jahre  number
endVar
```

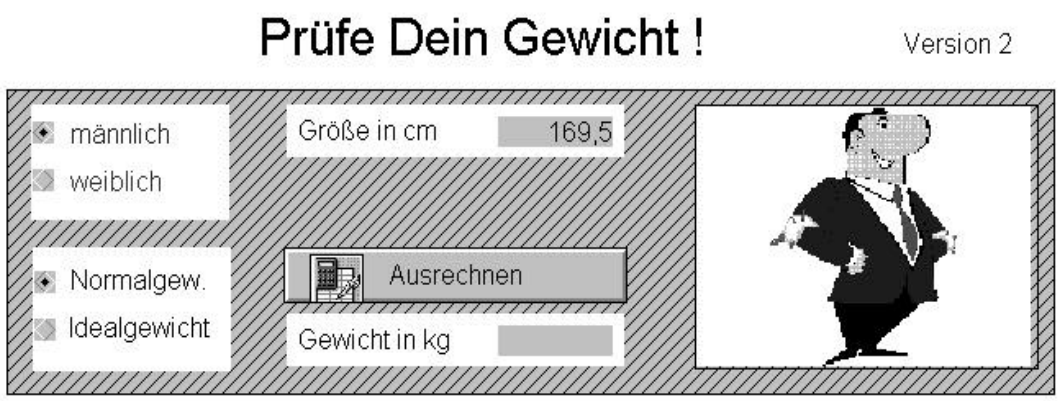
Nach kurzer Gewöhnungszeit erleichtern die Struktogramme die Übersicht erheblich!

Damit unsere schöne Landschaft nicht überstrapaziert wird, soll die nächste Übung ein etwas weniger spielerisches Problem berühren und zur Anwendung des Gelernten auf neue Aufgaben dienen.

Übung 2a: *Legen Sie ein neues Formular an. Erzeugen Sie die gezeichneten Objekte.*

Übung 2b: *Schreiben Sie für den Schalter “Ausrechnen” eine Methode* `pushbutton`*, die 1. überprüft, ob eine Größe eingegeben wurde und 2. wenn ja, nach der (umstrittenen) Formel* `Normalgewicht` *= Körpergröße - 100 das Gewicht berechnet.*

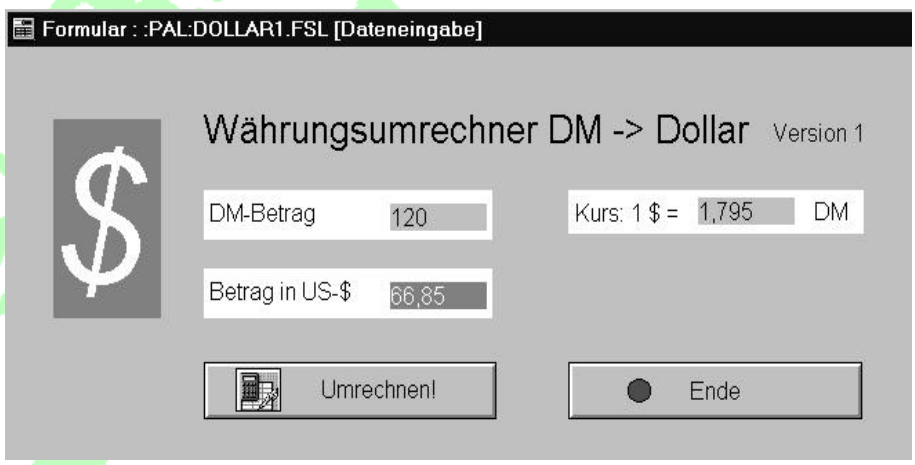
Auch hier kann die Verwendung von `try` hilfreich sein. Damit auch Kommazahlen wie 178,5 als Größe akzeptiert werden, ist eine explizite Verwandlung des Wertes in eine Zahl sinnvoll (mit der Funktion `number(Eingabe)` bzw. besser noch in OOP-Schreibweise mit der Methode `Eingabe.numVal()`).
Bei komplexeren Berechnungen empfiehlt sich außerdem die Verwendung von Variablen, z.B. wenn das Normalgewicht für Frauen nochmal 10% unter der bisher berechneten Empfehlung liegen soll und das Idealgewicht für beide Geschlechter angeblich nochmal um 10% unter dem Normalgewicht.

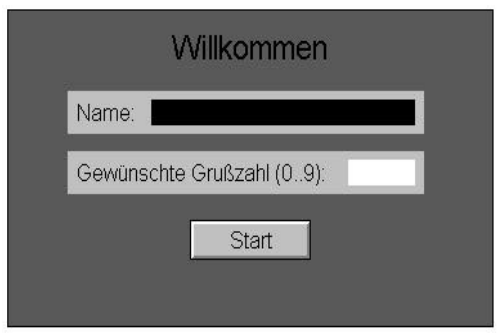


Übung 2c: *Verbessern Sie die Gewichtsberechnung!*

Wenn Ihnen dieses Problem keine Herausforderung mehr bietet, sollten Sie sich der harten Finanzmathematik zuwenden.

Übung 3: *Entwickeln Sie ein Programm zur Währungsumrechnung! (Entwickeln Sie zunächst eine erste lauffähige Fassung, die anschließend noch verbessert werden kann. U.a. könnte später sogar auf den Umrechnen-Knopf verzichtet werden.)*





In den bisherigen Beispielen mußten keine Wiederholstrukturen verwendet werden; es reichte der Gebrauch von Verzweigungen. Sollen hingegen Programmteile mehrfach ausgeführt werden, sind Wiederholungen nötig. Objekt-PAL bietet zwei Wiederholstrukturen mit vorgeschalteter Kontrolle, nämlich die Zählschleife (`for .. from .. to ... endfor`) für eine vor Eintritt in die Struktur bekannte Anzahl von Wiederholungen, sowie die `while`-Schleife, die auf Änderungen während des Wiederholens reagieren kann. Zwei Beispiele mögen das erläutern:

Im nachfolgenden Formular kann der Benutzer außer seinem Namen eine Zahl eintragen. Dadurch "weiß" das Programm, wie oft gegrüßt werden soll, und kann -- auf Knopfdruck -- eine entsprechende Zählschleife starten, die immer wieder eine entsprechende Meldung auf den Bildschirm schickt. Probieren Sie unbedingt Grußzahlen wie 0 oder -1 aus und machen Sie sich klar, was Wiederholung mit "vorgeschalteter Kontrolle" bedeutet!

```
method pushButton(var eventInfo Event)
var
  Zahl, wieoft smallint
endvar
if Name.isblank ()
  then msginfo ("Hinweis","Bitte erst den Namen eintippen")
  else if Anzahl.isblank ()
    then msginfo ("Hinweis","Bitte eine Zahl eintippen")
    else
      try wieoft = Anzahl
        if wieoft>9
          then msginfo ("Hinweis","maximal 9 erlaubt")
          else
            for Zahl from 1 to wieoft
              msginfo ("Gruß Nr. "+string(Zahl),"Hallo, "+Name+"!")
            endfor
        endif {wieoft}
      onfail msginfo ("Hinweis","Nur Ziffern erlaubt
            -- kein Komma und keine zu großen Zahlen!")
      endtry
    endif {Anzahl}
  endif {Name}
endMethod
```

Übung 4a: *Berechnen Sie Fakultäten (mit einer* for*-Schleife), d.h. lassen Sie nach Eingabe von z.B. 3 und Knopfdruck das Ergebnis 6 (=3!) erscheinen. Testen Sie den Zahlbereich von Objekt-PAL!*

Übung 4b: *Schreiben Sie ein Programm, das die natürlichen Zahlen von 1 bis zu einer eingetippten Endzahl addiert und das Ergebnis nennt (z.B. sollte bei der Endzahl 10 die Summe 55 angezeigt werden).*

Überlegen Sie, ob sich eine for-Schleife auch für Kettenrechnungen eignet, d.h. um mehrere verschiedene Zahlen zu addieren. Könnten so auch an der Supermarktkasse die Preise aller Artikel im Einkaufswagen zusammengezählt werden?

Im zweiten Beispiel des Grußprogramms kann während des Grüßens der Name des/der Begrüßten verändert oder ganz gelöscht werden. Im letzen Fall soll die Wiederholung und damit das Programm enden.

```
method pushButton(var eventInfo Event)
var
  Zahl smallint
endvar
if Name.isblank ()
  then msginfo ("Hinweis","Bitte erst den Namen eintippen")
  else Textzeile.Font.Color = white
       Zahl = 1
       Endehinweis = "Namen löschen um zu beenden!"
       while not Name.isblank()
        Textzeile = "Gruß Nr. "+string(Zahl)+": Hallo, "+Name+"!"
         sleep (500)
         Textzeile = "       "
         sleep (500)
         Zahl = Zahl + 1
       endwhile
       Endehinweis = "            "
       Textzeile.Font.Color = black
       Textzeile = "(hier erscheinen die Grüße!)"
endif {Name}
endMethod
```

Abgedruckt ist die Methode pushbutton für die mit "Los geht's" beschriftete Taste.

Die Variable Zahl dient hier nicht der Programmsteuerung, sondern zählt lediglich mit, wie oft gegrüßt wurde. Das Programm funktioniert auch, wenn auf die Zahl und alle damit zusammenhängenden Programmzeilen verzichtet wird.

Bei Programmen mit Wiederholstrukturen empfiehlt es sich übrigens immer, den Programmtext vor dem Start zu speichern: Bei einer Endlos-Schleife hilft manchmal nur der Griff zum Ausschalter.

Übung 5: *Geben Sie nacheinander verschiedene Zahlen ins gleiche Eingabefeld ein und addieren Sie alle diese Zahlen. Wie können Sie dem Computer klar machen, daß 1. Ihre Zahl fertig ist (und keine weiteren Ziffern folgen), 2. Ihre Rechnung beendet ist und Sie das Ergebnis sehen wollen?*
(Suchen Sie nach einer benutzerfreundlichen Lösung: Der Kassiererin im Supermarkt ist nicht zuzumuten, nach jedem Artikel auf die Frage "Wollen Sie wirklich noch einen Preis eintippen (j/n)?" antworten zu müssen).

Die geschickte Kombination von Befehlsfolgen, Verzweigungen und Wiederholungen ist die Grundlage aller Programme. Zum Abschluß sollen Sie Ihre Kenntnisse noch in einigen Projekten erproben bzw. vertiefen:

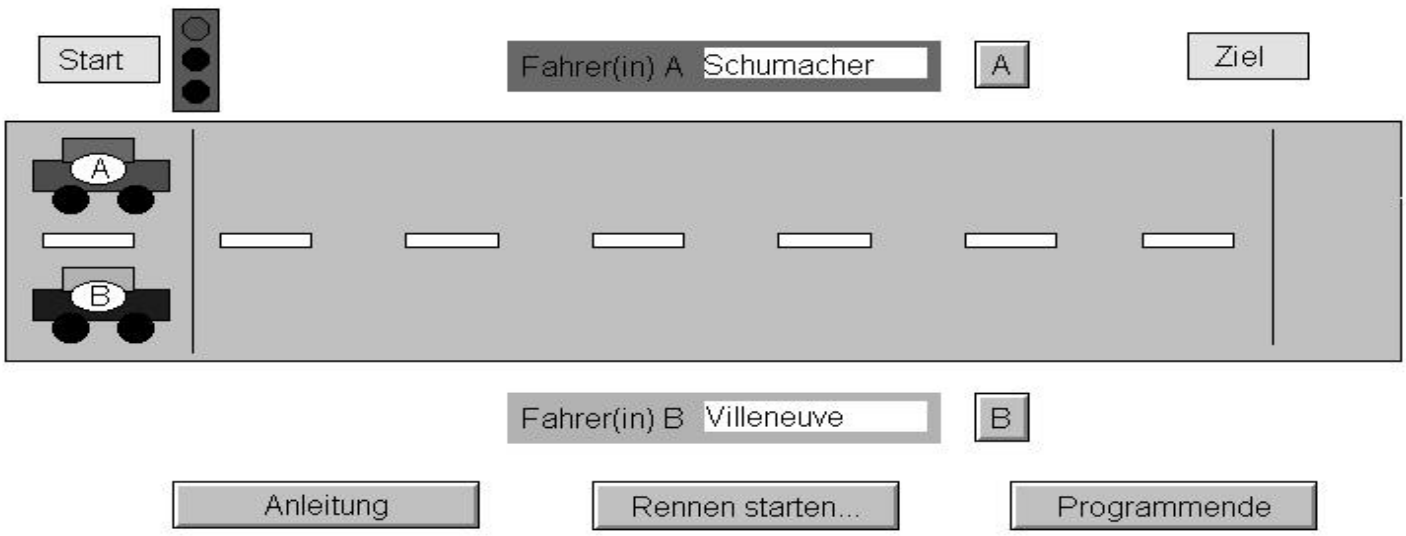
Rechner bitte nur mit der Maus bedienen (nicht per Tastatur!)

%-Taste arbeitet manchmal noch komisch!

Krell, 12/97

Projekt A: *Schreiben Sie ein Programm, das einen Taschenrechner simuliert. Der Taschenrechner soll ausschließlich mit der Maus bedient werden (wenn Sie die Tastatur verwenden, erscheinen eingetippte Rechenzeichen wie + oder * in der Anzeige und werden nicht als Operatoren erkannt -- oder Sie müssen diesen Effekt durch zusätzlichen Programmieraufwand abfangen!)*
Hinweis: Überlegen Sie, was sich der Taschenrechner alles merken muß und vereinbaren Sie dafür geeignete Variablen.

Projekt B: *Schreiben Sie ein Programm, das ein Autorennen simuliert. Zur Bewegung der Autos sollen verschiedene Möglichkeiten ausprobiert werden (zunächst das automatische Bewegen eines Wagens immer um die gleiche kleine Entfernung [z.B. 50 "mips"] bis zum Ziel, gleichzeitige oder abwechselnde Bewegung der Autos, schließlich Eingreifmöglichkeit seitens des Menschen per Mausklick oder Tastendruck...)*



Projekt C: *Wenden Sie Ihre Programmierkenntnisse an, um die Büchereiverwaltung durch Formulare zu erweitern, die PAL-Eingabefelder und mit Knöpfe enthalten, die mit verschiedenen Funktionen belegt sind!*