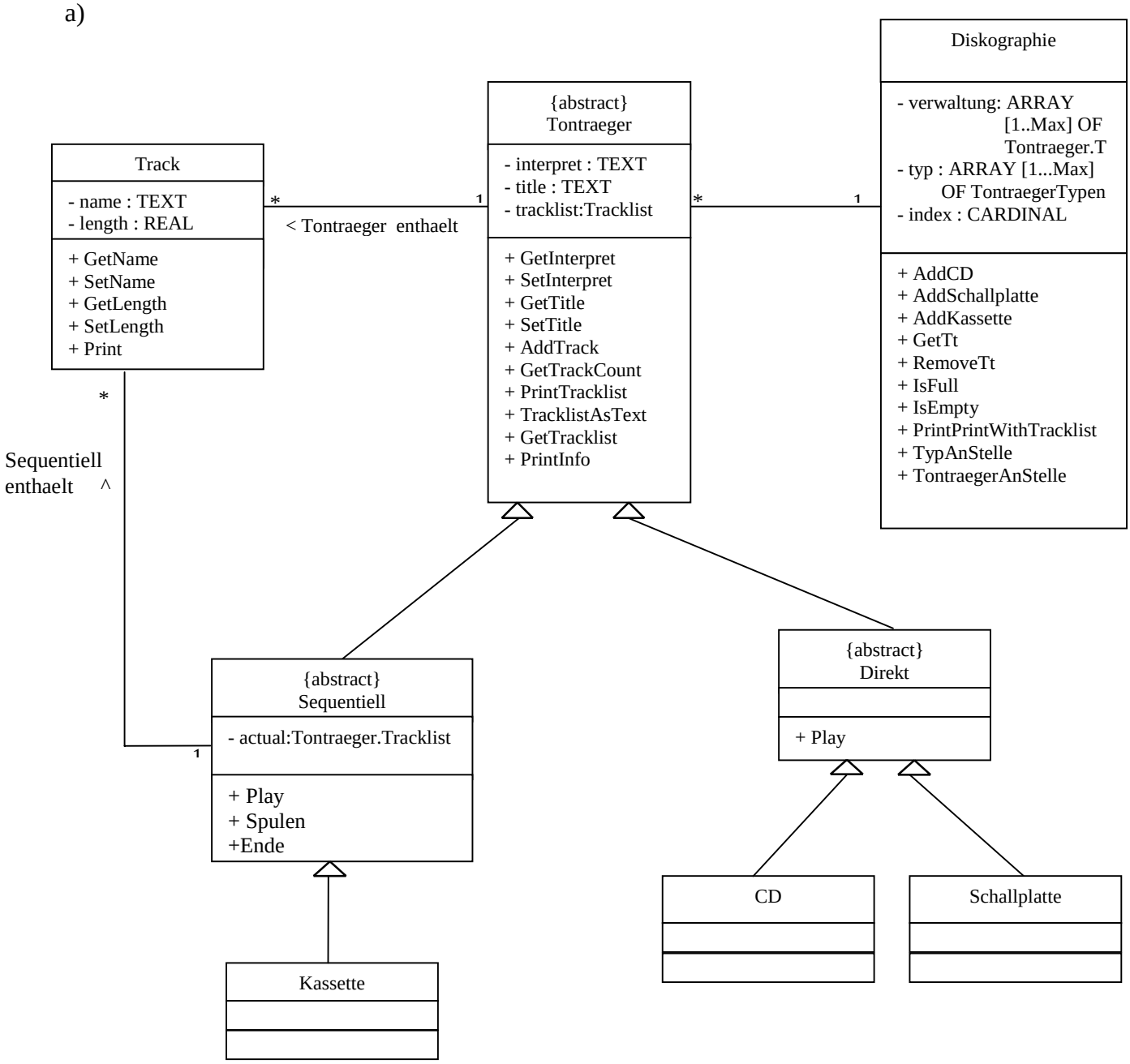
Programmierung **WS 2000 / 2001** **Prof. H. Lichter**
A. Nowack

Übung 11

Musterlösung

Aufgabe 11.1:

a)



Anmerkung:
Dies ist nur eine Möglichkeit, die Aufgabenstellung zu realisieren. Der Typ Trackliste ist definiert im Interface Tontraeger, eine Kapselung als Klasse bzw. eine Kapselung des Typs der Listenelemente sowie die Einfuehrung eines Containers wäre in objekt-orientierten

Sprachen notwendig. In Modula-3 (einer hybriden Sprache) ist dies nicht der Fall (wäre aber ein anderer möglicher Ansatz).

b)

```
INTERFACE Track;

TYPE T <: Public;
   Public = ROOT OBJECT
          METHODS
               GetName(): TEXT ;
               SetName(name: TEXT);
               GetLength(): CARDINAL ;
               SetLength(length: CARDINAL);
               Print();
          END;

END Track.
```

```
MODULE Track;

IMPORT SIO;

REVEAL
   T = Public BRANDED OBJECT
               name: TEXT;
               length: CARDINAL;
             OVERRIDES
               GetName := GetNameTrack;
               SetName := SetNameTrack;
               GetLength := GetLengthTrack;
               SetLength := SetLengthTrack;
               Print := PrintTrack;

    END;


PROCEDURE GetNameTrack(self: T): TEXT =
BEGIN
  RETURN self.name;
END GetNameTrack;

PROCEDURE SetNameTrack(self: T; n: TEXT) =
BEGIN
  self.name := n;
END SetNameTrack;
```

Entsprechend:
PROCEDURE GetLengthTrack(self: T): CARDINAL
PROCEDURE SetLengthTrack(self: T; l: CARDINAL)
PROCEDURE PrintTrack(self : T)

```
BEGIN
END Track.
```

```
INTERFACE Tontraeger;

IMPORT Track;

TYPE Tracklist = REF RECORD
                   track: Track.T;
                   next: Tracklist;
                 END;

TYPE T <: Public;
   Public = ROOT OBJECT
              METHODS
                    GetInterpret(): TEXT;
                    SetInterpret(interpret: TEXT);
                    GetTitle(): TEXT;
                    SetTitle(title: TEXT);
                    AddTrack(track: Track.T);
                    GetTrackCount(): CARDINAL;
                    PrintTracklist();
                    TracklistAsText() : TEXT;
                    GetTracklist() : Tracklist;
                    PrintInfo(): TEXT;
               END;

END Tontraeger.
```

```
MODULE Tontraeger;

IMPORT Track;
IMPORT SIO;
IMPORT Text;
IMPORT Fmt;

REVEAL
      T = Public BRANDED OBJECT
                  interpret: TEXT;
                  title: TEXT;
                  tracklist: Tracklist := NIL;
            OVERRIDES
                    GetInterpret := GetInterpretTt;
                    SetInterpret := SetInterpretTt;
                    GetTitle := GetTitleTt;
                    SetTitle := SetTitleTt;
                    AddTrack := AddTrackTt;
                    GetTrackCount := GetTrackCountTt;
                    PrintTracklist := PrintTracklistTt;
                    GetTracklist := GetTracklistTt;
```

```
                    PrintInfo := PrintTt;
                    TracklistAsText := TracklistAsTextTt;
          END;

PROCEDURE GetInterpretTt(self: T): TEXT =
BEGIN
  RETURN (self.interpret);
END GetInterpretTt;

PROCEDURE SetInterpretTt(self: T; interpret: TEXT) =
BEGIN
  self.interpret := interpret;
END SetInterpretTt;
```

Entsprechend:
PROCEDURE GetTitleTt(self: T): TEXT
PROCEDURE SetTitleTt(self: T; title: TEXT)

```
PROCEDURE AddTrackTt(self: T; track: Track.T) =
(* Fügt einen Track dem Tontraeger hinzu. Der neue Track wird
   an das Ende der Trackliste angehängt. *)
VAR newtrack, actual, prev: Tracklist;
VAR add : BOOLEAN := TRUE;
BEGIN
  newtrack := NEW(Tracklist);
  newtrack^.track := track;
  newtrack^.next := NIL;

  (* Falls Trackliste leer *)
  IF (self.tracklist = NIL) THEN
    self.tracklist := newtrack;
  ELSE
    (* Suche Ende der Trackliste *)
    actual := self.tracklist;
    prev := self.tracklist;
    WHILE (actual # NIL) DO
      (* Falls der einzufuegende Track bereits an der
         betrachteten Stelle steht *)
      IF Text.Equal(track.GetName() ,
                    (actual^.track).GetName()) AND
         track.GetLength() = (actual^.track).GetLength()
      THEN(* wird der Tontraeger nicht veraendert. *)
        actual := NIL;
        add := FALSE;
        SIO.PutLine("Der angegebene Track ist bereits
                     enthalten.");
      ELSE
        (* sonst Aenderung gemaess obiger Beschreibung *)
        prev := actual;
        actual := actual^.next;
      END;
    END;
```

```
    IF add THEN prev^.next := newtrack; END;
  END;
END AddTrackTt;
```

Entsprechend:
PROCEDURE GetTrackCountTt(self: T): CARDINAL
PROCEDURE PrintTracklistTt(self: T)
PROCEDURE GetTracklistTt(self : T) : Tracklist
PROCEDURE TracklistAsTextTt(self : T) : TEXT
(* Liefert die Trackliste des Tontraegers mit Namen und Länge der Tracks als Text. Dies wird realisiert durch einen rekursiven Durchlauf durch die Liste. *)

```
PROCEDURE PrintTt(self : T) : TEXT =
(* Liefert einen Text, der alle Informationen - inklusive
   Trackliste - ueber den Tontraeger enthaelt. *)
BEGIN
   RETURN "Interpret: " & self.interpret & "   Titel: " &
           self. title  & "\n" & self.TracklistAsText() ;
END PrintTt;

BEGIN
END Tontraeger.
```

```
INTERFACE Sequentiell;

IMPORT Tontraeger;
IMPORT Track;

TYPE T <: Public;
   Public = Tontraeger.T OBJECT
                  METHODS
                    Play() : Track.T;
                    (* Abspielen des sequentiellen
                       Tontraegers an der aktuellen Stelle *)
                    Spulen();
                    (* Spulen des Tontraegers *)
                    Ende(): BOOLEAN;
                    (* Ueberprueft, ob der Tontraeger an
                       seinem Ende angelangt ist. *)
                  END;

END Sequentiell.
```

```
MODULE Sequentiell;

IMPORT Track;
IMPORT Tontraeger;
IMPORT SIO;
IMPORT Text;
IMPORT Fmt;
```

```
REVEAL
   T = Public BRANDED OBJECT
                      actual: Tontraeger.Tracklist := NIL;
                   OVERRIDES
                    Play := PlayerSeq;
                    (* Abspielen des sequentiellen Tontraegers
                       an der aktuellen Stelle *)
                    Spulen := SpulenSeq;
                    PrintInfo := PrintSeq;
                    (* Dies ist eine Redefinition der
                       entsprechenden Methode des Typs
                       Tontraeger. *)
                    AddTrack := AddTrackSeq;
                    (* Fügt einen Track des übergebenen
                       Tontraegers hinzu. Dies ist eine
                       Redefinition der entsprechenden Methode
                       des Typen Tontraeger. *)
                    Ende := EndeSeq;
            END;


PROCEDURE SpulenSeq(self: T) =
VAR act : Tontraeger.Tracklist;
BEGIN
(* Die Trackliste wird von Anfang an so lange durchlaufen,
   bis der Vorgraenger des aktuellen Tracks gefunden
   ist. Der aktuelle Trakc wird durch seinen Vorgaenger
   ersetzt. *)
   act := self.GetTracklist();
  (* Wenn der aktuelle Track nicht der erste Track ist und
    somit der Track keinen Vorgraenger hat, der Tontraeger
    also nicht bereits am Anfang steht ... *)
  IF NOT self.Ende() THEN
    IF NOT ( Text.Equal( ((self.actual)^.track).GetName(),
                  ((self.GetTracklist())^.track).GetName())
           AND ( ((self.actual)^.track).GetLength () =
               ((self.GetTracklist())^.track).GetLength()))
    THEN
    (* ...dann suche den Vorgaenger des aktuellen Tracks ...*)
      WHILE  NOT (Text.Equal( ((act^.next)^.track).GetName(),
                              ((self.actual)^.track).GetName())
                   AND ( ((act^.next)^.track).GetLength () =
                       ((self.actual)^.track).GetLength())) DO
           act := act^.next;
        END;
        self.actual :=act;
        (* ... sonst ist Spulen nicht moeglich. *)

        ELSE SIO.PutLine("Spulen nicht moeglich! Der
                          Tontraeger ist bereits am Anfang.");
    END;
```

```
  ELSIF self.GetTracklist() = NIL
  THEN SIO.PutLine("Die Trackliste ist leer!");
  ELSE
    WHILE  NOT act.next = NIL DO
          act := act^.next;
        END;
        self.actual :=act;
  END;
END SpulenSeq;

PROCEDURE PlayerSeq(self : T) : Track.T =
VAR play: Tontraeger.Tracklist;
BEGIN
  (* Liefere den aktuellen Track zurueck ... *)
  play := self.actual;
  (* ... und ersetze den aktuellen Track durch seinen
     Vorgaenger ... *)
  self.actual := (self.actual)^.next;
  RETURN play.track;
END PlayerSeq;

PROCEDURE EndeSeq(self:T) : BOOLEAN =
BEGIN
  RETURN self.actual = NIL;
END EndeSeq;

PROCEDURE PrintSeq(self : T) : TEXT =
BEGIN
  (* Liefert zusaetzlich zu den Informationen, die die Methode
     des Supertypen Tontraeger liefert, die Zugriffsart. *)
  IF NOT self.actual = NIL THEN
    RETURN Tontraeger.T.PrintInfo(self) &   " Zugriff :
           Sequentiell  " & "\n" & "       " & " aktueller
           Track: " & (((self.actual)^.track).GetName())
           & "   Laenge: " & Fmt.Int
                     ((((self.actual)^.track).GetLength ()))
           & " Sekunden" &"\n" & "    ";
  ELSE
    RETURN Tontraeger.T.PrintInfo(self) &   " Zugriff :
           Sequentiell  " & "\n" & "       " & "Der Tontaeger
           befindet sich am Ende." &"\n" & "    ";
  END;
END PrintSeq;
```

Entsprechend: PROCEDURE AddTrackSeq(self: T; track: Track.T)
(* Fügt einen Track des übergebenen Tontraegers hinzu. Dies realisiert die Redefinition der entsprechenden Methode des Supertypen Tontraeger. Ausser der Aktualisierung der Trackliste ist hier auch die Aktualisierung der Liste notwendig, die den aktuellen Track angibt. Deren Aktualisierung wird genauso realisiert wie die der Trackliste. *)

```
BEGIN
```

```
END Sequentiell.
```

```
INTERFACE Direkt;

IMPORT Tontraeger;
IMPORT Track;

TYPE T <: Public;
   Public = Tontraeger.T  OBJECT
              METHODS
                Play(pos : CARDINAL) : Track.T;
                (* Abspielen des Tontraegers mit direktem
                   Zugriff. Der Track an der n-ten Position
                   der Trackliste wird zurueckgelierfert. *)
              END;

END Direkt.
```

```
MODULE Direkt;

IMPORT Tontraeger;
IMPORT Track;

REVEAL
   T = Public BRANDED OBJECT
                OVERRIDES
                  Play := PlayerDir;
                  (* Abspielen des Tontraegers mit direktem
                     Zugriff. Der Track an der n-ten Position
                     Der Trackliste wird zurueckgelierfert. *)
                  PrintInfo  := PrintDir;
                  (* Dies ist eine Redefinition der
                     entsprechenden Methode des Typs
                     Tontraeger. *)
        END;

PROCEDURE PlayerDir(self : T; n: CARDINAL) : Track.T =
VAR liste: Tontraeger.Tracklist;
BEGIN
   liste := Tontraeger.T.GetTracklist(self);
   (* Suche den Track an der n-ten Stelle ... *)
   FOR i := 1 TO (n-1)  DO
       liste := liste^.next;
   END;
   (* ... und liefere diesen zurueck. *)
   RETURN liste.track;
END PlayerDir;

PROCEDURE PrintDir(self : T) : TEXT =
BEGIN
```

```
(* Liefert zusaetzlich zu den Informationen, die die Methode
   des Supertypen Tontraeger liefert, die Zugriffsart. *)

  RETURN Tontraeger.T.PrintInfo(self) & " Zugriff : Direkt"
         &"\n" & "    ";
END PrintDir;

BEGIN
END Direkt.
```

```
INTERFACE Kassette;

IMPORT Sequentiell;

TYPE T <: Public;
   Public = Sequentiell.T OBJECT
             END;
END Kassette.
```

```
MODULE Kassette;

IMPORT Sequentiell;

REVEAL
   T = Public BRANDED OBJECT
     OVERRIDES
         PrintInfo := PrintKassette;
     END;

PROCEDURE PrintKassette(self : T) : TEXT =
BEGIN
  RETURN Sequentiell.T.PrintInfo(self) & " Typ : Kassette";
END PrintKassette;

BEGIN
END Kassette.
```

```
INTERFACE CD;

IMPORT Direkt;

TYPE T <: Public;
   Public = Direkt.T  OBJECT
            END;

END CD.
```

```
MODULE CD;

IMPORT Direkt;

REVEAL
   T = Public BRANDED OBJECT
          OVERRIDES
            PrintInfo := PrintCD;
            (* Ausgabe aller Informationen ueber den
               Tontraeger als Text. Dies ist eine Redefinition
               der entsprechenden Methode des Typen Direkt. *)
          END;

PROCEDURE PrintCD(self :T) : TEXT =
BEGIN
  RETURN Direkt.T.PrintInfo(self) & " Typ : CD";
END PrintCD;


BEGIN
END CD.
```

```
INTERFACE Schallplatte;

IMPORT Direkt;
TYPE T <: Public;
   Public = Direkt.T OBJECT
  END;
END Schallplatte.
```

```
MODULE Schallplatte;

IMPORT Direkt;

REVEAL
   T = Public BRANDED OBJECT
          OVERRIDES
            PrintInfo := PrintSchallplatte;
            (* Dies ist eine Redefinition der entsprechenden
               Methode des Typen Direkt. *)
          END;

PROCEDURE PrintSchallplatte(self : T) : TEXT =
BEGIN
  RETURN Direkt.T.PrintInfo(self) & " Typ : Schallplatte";
END PrintSchallplatte;

BEGIN
END Schallplatte.
```

```
INTERFACE Diskographie;

IMPORT Tontraeger;
IMPORT CD;
IMPORT Schallplatte;
IMPORT Kassette;

TYPE TontraegerTypen = {CDTyp, SchallplatteTyp, KassetteTyp};

TYPE T <: Public;
   Public = ROOT OBJECT
              METHODS
                AddCD(cd: CD.T);
                AddSchallplatte(schp: Schallplatte.T);
                AddKassette(kassette: Kassette.T);
                GetTt(index: CARDINAL): Tontraeger.T;
                RemoveTt(pos: CARDINAL;);
                IsFull(): BOOLEAN;
                IsEmpty(): BOOLEAN;
                Print();
                PrintWithTracklist();
                TypAnStelle(i : CARDINAL) : TontraegerTypen;
                TontraegerAnStelle(i : CARDINAL):Tontraeger.T;
              END;

END Diskographie.
```

```
MODULE Diskographie;

IMPORT SIO;
IMPORT CD;   (* Importiere den ADT CD *)
IMPORT Tontraeger;
IMPORT Schallplatte;
IMPORT Kassette;

CONST Max = 5;

REVEAL
      T = Public BRANDED OBJECT
                  verwaltung:  ARRAY [1 .. Max] OF
                               Tontraeger.T;
                  typ : ARRAY [1..Max] OF TontraegerTypen;
                  index : CARDINAL := 0;
            OVERRIDES
                AddCD := AddCDDisk;
                AddSchallplatte := AddSchallplatteDisk;
                AddKassette := AddKassetteDisk;
                GetTt := GetTtDisk;
                RemoveTt:= RemoveTtDisk;
                IsFull := IsFullDisk;
```

```
                IsEmpty := IsEmptyDisk;
                Print := PrintDisk;
                PrintWithTracklist := PrintWithTracklistDisk;
                TypAnStelle := TypAnStelleDisk;
                TontraegerAnStelle := TontraegerAnStelleDisk;

            END;

PROCEDURE AddCDDisk(self: T;  cd: CD.T)=
BEGIN
  (* CD wird im nächsten Speicherplatz abgelegt *)
  IF NOT self.IsFull() THEN
    self.index := self.index + 1;
    self.verwaltung[self.index] := cd;
    self.typ[self.index] := TontraegerTypen.CDTyp;
  ELSE
    SIO.PutLine("Diskographie ist voll!!!");
  END;
END AddCDDisk;
```

Entsprechend:
PROCEDURE AddSchallplatteDisk(self: T; sch: Schallplatte.T)
PROCEDURE AddKassetteDisk(self: T; kass: Kassette.T)
PROCEDURE RemoveTtDisk(self:T; pos: CARDINAL)
PROCEDURE IsFullDisk(self: T): BOOLEAN
PROCEDURE IsEmptyDisk(self:T): BOOLEAN
PROCEDURE PrintDisk(self : T)=

```
PROCEDURE PrintWithTracklistDisk(self:T)=
BEGIN
  SIO.PutLine(" *** Diskographieverwaltung *** ");
  SIO.Nl();
  FOR i:=1 TO self.index DO
    SIO.PutInt(i); SIO.PutText(".   ");
    IF self.typ[i] = TontraegerTypen.CDTyp
    THEN SIO.PutLine(CD.T.PrintInfo(self.verwaltung[i])) END;
    IF self.typ[i] = TontraegerTypen.SchallplatteTyp
    THEN
     SIO.PutLine(Schallplatte.T.PrintInfo(self.verwaltung[i]))
    END;
    IF self.typ[i] = TontraegerTypen.KassetteTyp
    THEN SIO.PutLine(Kassette.T.PrintInfo(self.verwaltung[i]))
END;
  END;
END PrintWithTracklistDisk;
```

Entsprechend:
PROCEDURE GetTtDisk(self: T; i: CARDINAL): Tontraeger.T =
PROCEDURE TypAnStelleDisk(self : T; i : CARDINAL) : TontraegerTypen =
PROCEDURE TontraegerAnStelleDisk(self: T; i: CARDINAL): Tontraeger.T =

```
BEGIN
```

```
END Diskographie.
```

```
MODULE Verwaltung EXPORTS Main;

IMPORT SIO;
IMPORT Diskographie;  (* Importiere ADO Diskographie *)
IMPORT Tontraeger;
IMPORT CD;            (* Importiere ADT CD *)
IMPORT Schallplatte;
IMPORT Kassette;
IMPORT Track;         (* Importiere ADT Track *)

VAR dummy : TEXT;
    auswahl : CHAR;
    cd : CD.T;
    sch : Schallplatte.T;
    kass : Kassette.T;
    tt : Tontraeger.T;
    track: Track.T;
    nummer: CARDINAL;
    disk : Diskographie.T;
    i,j : CARDINAL;
```

Die Implementierung folgender Hilfsprozedur zur Ausgabe des Menüs ist offensichtlich:
PROCEDURE PrintMenu()

```
BEGIN
  disk := NEW(Diskographie.T);

  REPEAT
    PrintMenu();

    (* Frage Benutzer nach seiner Auswahl *)
    SIO.PutText("Auswahl: ");
    auswahl := SIO.GetChar();
    dummy := SIO.GetLine();
    (* Lies RETURN aus dem Eingabepuffer! *)
    SIO.Nl();

    CASE auswahl OF
    (* Erstellt eine neue CD *)
    |  'c', 'C' => cd :=NEW( CD.T);
                   SIO.PutText("Bitte geben Sie den CD-Titel
                                ein: ");
                   cd.SetTitle(SIO.GetLine());
                   SIO.PutText("Bitte geben Sie den
                                Interpreten ein: ");
                   cd.SetInterpret(SIO.GetLine());
                   disk.AddCD(cd);

    |  's', 'S' => ... Erstellung einer Schallplatte ... wie für CDs
```

```
  |  'k', 'K' => ... Erstellung einer Kassette ... wie für CDs


|  'L', 'l' =>  SIO.PutLine("Bitte geben Sie die Nummer des
                           Tontraegers ein: ");
              i := SIO.GetInt();
              disk.RemoveTt(i);


  |  'Z', 'z' => disk.PrintWithTracklist();

  |  'T', 't' => Diskographie.T.Print(disk);
              SIO.PutText("Geben Sie die Tontrager-Nummer
                           ein: ");
              nummer := SIO.GetInt();
              dummy := SIO.GetLine();
              tt := disk.GetTt(nummer);
              IF (tt # NIL) THEN
                REPEAT
                  SIO.PutText("Möchten Sie einen Track
                               eingeben (j/n)?");
                  auswahl := SIO.GetChar();
                  dummy := SIO.GetLine();
                  IF (auswahl = 'j') OR (auswahl = 'J')
                  THEN
                    track := NEW(Track.T);
                    SIO.PutText("Geben sie den Namen des
                                 Tracks ein: ");
                    track.SetName(SIO.GetLine());
                    SIO.PutText("Geben Sie die Laenge des
                             Tracks in Sekunden ein: ");
                    track.SetLength(SIO.GetInt());
                    dummy := SIO.GetLine();
                    tt.AddTrack(track);
                  END;
                UNTIL (auswahl = 'n') OR (auswahl = 'N');
              END;


  | 'P', 'p' => Diskographie.T.Print(disk);
              SIO.PutLine("Bitte die Nummer des
                           Tontraegers eingeben: ");
              i := SIO.GetInt();
              IF disk.TypAnStelle(i) =
                       Diskographie.TontraegerTypen.CDTyp
              THEN
                cd := disk.TontraegerAnStelle(i);
                SIO.PutLine("Bitte die Tracknummer
                             eingeben:");
                j := SIO.GetInt();
```

```
                    (cd.Play(j)).Print();
                  ELSIF disk.TypAnStelle(i) =
                           Diskographie.
                             TontraegerTypen.SchallplatteTyp
                  THEN
                    Abspielen einer Schallplatte ... wie für CDs
                  THEN
                    kass := disk.TontraegerAnStelle(i);
                    IF kass.Ende()
                    THEN SIO.PutLine("Die Kassette ist am
                                      Ende. Abspielen nicht
                                      moeglich!");
                    ELSE
                       (kass.Play()).Print();
                     END;
                  END;
    | 'B', 'b' => Diskographie.T.Print(disk);
                     SIO.PutLine("Bitte geben Sie die Nummer
                                  des Tontraegers ein");
                     i := SIO.GetInt();
                     IF disk.TypAnStelle(i) =
                         Diskographie.TontraegerTypen.
                           KassetteTyp
                     THEN
                        kass := disk.TontraegerAnStelle(i);
                        kass.Spulen();
                     ELSE
                        SIO.PutLine("Der Tontraeger hat
                                     keinen sequentiellem
                                     Zugriff!");
                     END;

    (* Das Programm beenden *)
    |  'E', 'e' => (* Nichts machen, gleich ist ohnehin alles
                     zu Ende! *)
    ELSE
      SIO.PutLine("Ungueltiger Befehl!");
    END (* CASE *)
  UNTIL (auswahl = 'e') OR (auswahl = 'E');
END Verwaltung.
```

Testlauf:

C : Fuege neue CD ein
S : Fuege neue Schallplatte ein
K : Fuege neue Kassette ein
L : Loesche Tontraeger
Z : Zeige Diskographie an
T : Fuege Track in Tontraeger ein
X : Loesche Track von Tontraeger
P : Track von einem Tontraeger spielen
B : Spule Kassette zurueck

E : Exit

Auswahl: c, dann Auswahl: s, dann Auswahl: k jeweils mit Eingabe, so daß sich die untenstehende Diskographie ergibt

... Menü ...
Auswahl: t
*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 0
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 0
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 0

Geben Sie die Tontrager-Nummer ein: 1
Moechten Sie einen Track eingeben (j/n)?j
Geben sie den Namen des Tracks ein: cdtrack
Geben Sie die Laenge des Tracks in Sekunden ein: 1
Moechten Sie einen Track eingeben (j/n)?n

... Menü ...

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 0
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 0

Geben Sie die Tontrager-Nummer ein: 2
Moechten Sie einen Track eingeben (j/n)?j
Geben sie den Namen des Tracks ein: schallplattentrack
Geben Sie die Laenge des Tracks in Sekunden ein: 2
Moechten Sie einen Track eingeben (j/n)?n

... Menü ...
Auswahl: t

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 0

Geben Sie die Tontraeger-Nummer ein: 3
Moechten Sie einen Track eingeben (j/n)?j
Geben sie den Namen des Tracks ein: kassettentrack1
Geben Sie die Laenge des Tracks in Sekunden ein: 4
Moechten Sie einen Track eingeben (j/n)?j
Geben sie den Namen des Tracks ein: kassettentrack2
Geben Sie die Laenge des Tracks in Sekunden ein: 5
Moechten Sie einen Track eingeben (j/n)?n

... Menü ...
Auswahl: z

*** Diskographieverwaltung ***

1. Interpret: CDinterpret Titel: CD
   1. CDtrack Dauer: 1 Sekunden

   Zugriff : Direkt
   Typ : CD
2. Interpret: Schallplatteninterpret Titel: Schallplatte
   1. Schallplattentrack Dauer: 2 Sekunden

   Zugriff : Direkt
   Typ : Schallplatte
3. Interpret: Kassetteninterpret Titel: Kassette
   1. Kassettentrack1 Dauer: 4 Sekunden
   2. Kassettentrack2 Dauer: 5 Sekunden

   Zugriff : Sequentiell
   aktueller Track: Kassettentrack1 Laenge: 4 Sekunden
   Typ : Kassette

... Menü ...
Auswahl: P

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 2

Bitte die Nummer des Tontraegers eingeben:
1
Bitte die Tracknummer eingeben:
1
Titel: CDtrack Laenge: 1 Sekunden

... Menü ...
Auswahl: p

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 2

Bitte die Nummer des Tontraegers eingeben:
3
Titel: Kassettentrack1 Laenge: 4 Sekunden

... Menü ...
Auswahl: p

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks: 2

Bitte die Nummer des Tontraegers eingeben:
3
Titel: Kassettentrack2 Laenge: 5 Sekunden

... Menü ...
Auswahl: b

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassetteninterpret Interpret: Kassetteninterpret Anzahl Tracks: 2

Bitte geben Sie die Nummer des Tontraegers ein
3

... Menü ...
Auswahl: p

*** Diskographieverwaltung ***

1. Titel: CD Interpret: CDinterpret Anzahl Tracks: 1
2. Titel: Schallplatte Interpret: Schallplatteninterpret Anzahl Tracks: 1
3. Titel: Kassette Interpret: Kassetteninterpret Anzahl Tracks:

2
Bitte die Nummer des Tontraegers eingeben:
3
Titel: Kassettentrack2 Laenge: 5 Sekunden

... Menü ...
Auswahl: e

Aufgabe 11.2:

```
INTERFACE Boxer;

TYPE T <: REFANY;  (* Abstrakter Datentyp Spieler *)

PROCEDURE LeseEin(nachricht: TEXT): T;
PROCEDURE Drucke(boxer: T);
PROCEDURE Gleich(spielerA, spielerB: T): BOOLEAN;

END Boxer.
```

```
MODULE Boxer;

IMPORT SIO;  (* Importiere Operationen fuer Ein-/Ausgabe *)
```

```
IMPORT Text; (* Importiere Operationen fuer Textmanipulation
              *)

TYPE Name = RECORD
              vorname : TEXT;
              nachname: TEXT;
            END;

     (* Definiere nach aussen nur als abstrakten Datentyp
        sichtbaren Datentyp fuer Boxer *)

     REVEAL T = BRANDED REF RECORD
                  name       : Name;
                  nation     : TEXT;
                  gewicht    : CARDINAL;
                END;

PROCEDURE LeseEin(nachricht: TEXT): T=
(* Fragt den Benutzer nach den Daten eines Boxers und gibt
   anschliessend einen entsprechenden Boxer zurueck. *)

VAR resultat: T;
    dummy   : TEXT;

BEGIN
  (* Erzeuge einen neuen Boxer auf der Halde *)
  resultat := NEW(T);

  (* Erfrage Daten vom Benutzer *)
  SIO.PutLine(nachricht);
  SIO.PutText("Nachname   : ");
  resultat^.name.nachname := SIO.GetLine();
  SIO.PutText("Vorname    : ");
  resultat^.name.vorname := SIO.GetLine();
  SIO.PutText("Nation     : ");
  resultat^.nation       := SIO.GetLine();
  SIO.PutText("Gewicht: ");
  resultat^.gewicht := SIO.GetInt();

  (* Lies RETURN nach Gewicht aus der Eingabe! *)
  dummy := SIO.GetLine();

  RETURN resultat;
END LeseEin;
```

Entsprechend:
PROCEDURE Drucke(boxer: T)
PROCEDURE Gleich(boxerA, boxerB: T): BOOLEAN

```
BEGIN
END Boxer.
```

```
INTERFACE Boxen;

IMPORT Boxer;  (* Importiere den ADT Boxer *)
IMPORT Assertion AS As;

PROCEDURE VollRL(): BOOLEAN;
PROCEDURE BoxerExistiertRL(boxer: Boxer.T): BOOLEAN;
PROCEDURE EinfuegenBoxerRL(boxer: Boxer.T)RAISES{As.Violated};
(* Fuegt den gegebenen Boxer am Ende der Rangliste ein.
   Vorbedingung: Der Boxer ist noch nicht in der Rangliste
   enthalten und die Rangliste ist noch nicht  voll.
   Vorbedingung: Der einzufuegende Boxer darf noch nicht in
   der Liste enthalten sein.
   Nachbedingung: Der Boxer muss in der Liste enthalten
   sein.*)

PROCEDURE LoescheBoxerRL(boxer:Boxer.T) RAISES{As.Violated};
(* Loescht den gegebenen Boxer aus der Rangliste.
   Vorbedingung:
   Der Boxer ist in der Rangliste enthalten.
   Vorbedingung: Der zu loeschende Boxer muss in der Rangliste
   enthalten sein.
   Nachbedingung: Der zu loeschende Boxer darf nach dem
   Loeschen nicht mehr in der Rangliste enthalten sein. *)

PROCEDURE AktualisiereRL(gewinner, verlierer: Boxer.T)
                                          RAISES{As.Violated};
(* Aendert die Rangfolge in der Rangliste entsprechend dem
   gegebenen Kampfergebnis (Gewinner, Verlierer).
   Vorbedingung: Gewinner und Verlierer sind in der Rangliste
   enthalten
   Vorbedingung: Gewinner und Verlierer muessen beide in der
   Liste enthalten sein. *)

PROCEDURE DruckeRL();

END Boxen.
```

```
MODULE BoxListe EXPORTS Boxen;
(* Dieses Modul implementiert die Verwaltung einer
   Boxrangliste wie in der Schnittstelle Boxen definiert, mit
   Hilfe einer einfach verketteten Liste. *)

IMPORT SIO; (* Importiere Operationen fuer die Ein-/Ausgabe *)
IMPORT Boxer;  (* Importiere den ADT Boxer *)
IMPORT Assertion AS As;

TYPE RanglisteRef = REF RanglistenElement;

     RanglistenElement = RECORD
```

```
                        boxer  : Boxer.T;
                        naechster: RanglisteRef;
                      END;

VAR rangliste    : RanglisteRef; (* Anker der Boxangliste *)
```

Die Implementierung der folgenden Prozeduren sollte klar sein:
PROCEDURE SucheBoxerRL(boxer: Boxer.T): RanglisteRef
(* Sucht den gegebenen Boxer in der Rangliste und gibt einen Zeiger auf ihn zurueck. Der Zeiger hat den Wert NIL, wenn der Boxer nicht in der Liste vorhanden ist. *)
PROCEDURE SucheVorgaengerRL(boxer: Boxer.T): RanglisteRef
(* Sucht den Vorgaenger des gegebenen Boxers in der Rangliste und gibt einen Zeiger auf ihn zurueck. Der Zeiger hat den Wert NIL, wenn der Boxer keinen Vorgaenger hat, d.h. wenn er das erste Element oder nicht in der Liste ist. *)
PROCEDURE VollRL(): BOOLEAN
PROCEDURE BoxerExistiertRL(boxer: Boxer.T): BOOLEAN

```
PROCEDURE EinfuegenBoxerRL(boxer: Boxer.T)RAISES{As.Violated}=

VAR ranglisteRef, aktRLRef: RanglisteRef;

BEGIN
  (* Vorbedingungen *)
  As.Require(NOT BoxerExistiertRL(boxer), "Der Boxer ist
              schon in der Liste enthalten!");
  As.Require(NOT VollRL(), "Die Liste ist voll!!!");
  (* Erzeuge neues Ranglisten-Element fuer den Boxer *)
  ranglisteRef := NEW(RanglisteRef);
  ranglisteRef^.boxer := boxer;
  ranglisteRef^.naechster := NIL;

  (* Fuege Element an Ende der Liste an *)
  IF (rangliste = NIL) THEN
    (* Liste ist noch leer *)
    rangliste := ranglisteRef;
  ELSE
    (* Suche Ende der Liste *)
    aktRLRef := rangliste;
    WHILE (aktRLRef^.naechster # NIL) DO
      aktRLRef := aktRLRef^.naechster;
    END;
    (* Haenge Boxer ans Ende an *)
    aktRLRef^.naechster := ranglisteRef;
  END;
  (* Nachbedingung *)
  As.Ensure(BoxerExistiertRL(boxer), "Fehler beim Einfuegen
   des Boxers!! Der Boxer ist nicht in der Liste enthalten!");

END EinfuegenBoxerRL;

PROCEDURE LoescheBoxerRL(boxer: Boxer.T) RAISES{As.Violated}=
```

```
VAR ranglisteRef,
    vorgaengerRef: RanglisteRef;

BEGIN
   (* Vorbedingung *)
   As.Require(BoxerExistiertRL(boxer), "Der Boxer ist nicht in
   der Liste enthalten!");
   (* Suche den Boxer in der Liste *)
   ranglisteRef := SucheBoxerRL(boxer);

  IF (ranglisteRef # NIL) THEN

    (* Vorhandenen Boxer aus der Rangliste loeschen *)
    vorgaengerRef := SucheVorgaengerRL(boxer);
    IF vorgaengerRef = NIL THEN
      (* Erstes Element in Rangliste, daher kein Vorgaenger *)
      rangliste := ranglisteRef^.naechster;
    ELSE
      vorgaengerRef^.naechster := ranglisteRef^.naechster;
    END;
  END;
  (* Nachbedingung *)
  As.Ensure(NOT BoxerExistiertRL(boxer), "Fehler beim
              Loeschen! Der Boxer ist noch immer enthalten1");
END LoescheBoxerRL;
```

Entsprechend: PROCEDURE AKommtHinterBInRL(aRef, bRef: RanglisteRef):BOOLEAN

```
PROCEDURE AktualisiereRL(gewinner, verlierer: Boxer.T)
                                          RAISES{As.Violated}=

VAR gewinnerRef, verliererRef      : RanglisteRef;
    vorVerliererRef: RanglisteRef;

BEGIN
  (* Vorbedingungen *)
  As.Require(BoxerExistiertRL(gewinner), "Der Gewinner ist
             nicht in der Liste enthalten!");
  As.Require(BoxerExistiertRL(verlierer), "Der Verlierer ist
             nicht in der Liste enthalten!");

  (* Suche zunaechst Gewinner und Verlierer in der Rangliste
   *)
  gewinnerRef := SucheBoxerRL(gewinner);
  verliererRef := SucheBoxerRL( verlierer);

  (* Pruefe, ob nicht Gewinner ohnehin vor Verlierer in der
     Rangliste! *)
  IF NOT AKommtHinterBInRL(gewinnerRef, verliererRef) THEN
    SIO.PutLine("Keine Veraenderung in der Rangliste!");
  ELSE(* Fuege Verlierer hinter Gewinner in Rangliste ein *)
```

```
      (* Suche die jeweiligen Vorgaenger in der Rangliste *)
      vorVerliererRef := SucheVorgaengerRL(verlierer);

      (* Entferne Verlierer aus ursprünglicher Position *)
      IF ( vorVerliererRef = NIL) THEN
        (* Kein Vorgaenger, da ganz am Anfang der Liste *)
        rangliste := verliererRef^.naechster;
      ELSE
        vorVerliererRef^.naechster := verliererRef^.naechster;
      END;

      (* Hänge Verlierer hinter Gewinner ein *)
      verliererRef^.naechster := gewinnerRef^.naechster;
      gewinnerRef^.naechster := verliererRef;
       (* Nachbedingung *)
      As.Ensure(AKommtHinterBInRL(verliererRef, gewinnerRef),
                "Fehler beim Bearbeiten des Kampfes");
    END;
END AktualisiereRL;
```

Entsprechend: PROCEDURE DruckeRL()

```
BEGIN
  (* Initialisiere Liste mit leerer Rangliste *)
  rangliste := NIL;
END BoxListe.
```

```
MODULE BoxerRl EXPORTS Main;

IMPORT SIO; (* Importiere Operationen fuer die Ein-/Ausgabe *)
IMPORT Boxer; (* Importiere den ADT Boxer *)
IMPORT Boxen; (* Importiere das Objektmodul Boxen *)
IMPORT Assertion AS As;

VAR boxer, gewinner, verlierer  : Boxer.T; (* Variablen des
                                              ADTs Boxer *)
    auswahl                     : CHAR;
    dummy                       : TEXT;
```

Die Implementierung folgender Hilfsprozedur, die Hilfsprozedur, die das Befehlsmenü ausgibt, ist offensichtlich: PROCEDURE DruckeMenue()

```
PROCEDURE BoxerRl() RAISES {As.Terminate}=
  BEGIN
    REPEAT
      TRY
        DruckeMenue();
        (* Frage Benutzer nach seiner Auswahl *)
        SIO.PutText("Auswahl: ");
        auswahl := SIO.GetChar();
        dummy := SIO.GetLine(); (* Lies RETURN aus dem
```

```
                                    Eingabepuffer! *)
        SIO.Nl();

        CASE auswahl OF
        |  'F', 'f' => boxer := Boxer.LeseEin("Boxer
                      Einfuegen"); SIO.Nl();
                      Boxen.EinfuegenBoxerRL(boxer);

        |  'L', 'l' => boxer := Boxer.LeseEin("Boxer
                      Loeschen"); SIO.Nl();
                      Boxen.LoescheBoxerRL(boxer);

        |  'G', 'g' => gewinner := Boxer.LeseEin("Sieger des
                      Kampfes"); SIO.Nl();
                      verlierer := Boxer.LeseEin("Verlierer
                      des Kampfes"); SIO.Nl();
                      Boxen.AktualisiereRL(gewinner,
                      verlierer);

        |  'Z', 'z' => Boxen.DruckeRL();

        |  'E', 'e' => (* Nichts machen, gleich ist ohnehin
                         alles zu Ende! *)
        ELSE
          SIO.PutLine("Ungueltiger Befehl!");
        END (* CASE *)

    EXCEPT
      | As.Violated =>
                SIO.PutLine ("*********** Fehler beim
                              Bearbeiten der Boxrangliste
                              ************");
                RAISE As.Terminate;
      | SIO.Error => SIO.PutLine("SIO-Fehler!!");
    END;

    UNTIL (auswahl = 'e') OR (auswahl = 'E');
  END BoxerRl;

BEGIN
  As.EnableAssertions();
  TRY
     BoxerRl();
  EXCEPT
    |As.Terminate =>
      SIO.PutLine("***  Programm wird beendet wegen des obigen
                   Fehlers ***");
   END;
END BoxerRl.
```

Testlauf

F : Fuege einen Boxer ein
L : Loesche einen Boxer
G : Kampf gemacht
Z : Zeige Rangliste an
E : Exit

Auswahl: f

Boxer Einfuegen
Nachname : Schwer
Vorname : Arno
Nation : D
Gewicht: 200

... Menü ...
Auswahl: f

Boxer Einfuegen
Nachname : Mittel
Vorname : Bert
Nation : Aut
Gewicht: 125

... Menü ...
Auswahl: f

Boxer Einfuegen
Nachname : Leicht
Vorname : Christian
Nation : CH
Gewicht: 70

... Menü ...
Auswahl: z

*** Box-Rangliste ***

1. Schwer Arno Nation D Gewicht 200
2. Mittel Bert Nation Aut Gewicht 125
3. Leicht Christian Nation CH Gewicht 70

... Menü ...

Boxer Loeschen
Nachname : Unbekannt
Vorname : Unbekannt
Nation : Unbekannt
Gewicht: 100

*** Precondition violated: Der Boxer ist nicht in der Liste enthalten!
********** Fehler beim Bearbeiten der Boxrangliste **********
*** Programm wird beendet wegen des obigen Fehlers ***

.... Neu starten mit gleichen Eingaben, wobei die gleichen Boxer wie zuvor in der gleichen Reihenfolge wie zuvor eingegeben werden ...
... Menü ...
Auswahl: z

Sieger des Kampfes
Nachname   : Unbekannt
Vorname   : Unbekannt
Nation   : Unbekannt
Gewicht: 100

Verlierer des Kampfes
Nachname   : a
Vorname   : a
Nation   : a
Gewicht: 1

*** Precondition violated: Der Gewinner ist nicht in der Liste enthalten!
********** Fehler beim Bearbeiten der Boxrangliste **********
*** Programm wird beendet wegen des obigen Fehlers ***

.... Neu starten mit gleichen Eingaben, wobei die gleichen Boxer wie zuvor in der gleichen Reihenfolge wie zuvor eingegeben werden ...
... Menü ...
Auswahl: g

Sieger des Kampfes
Nachname   : a
Vorname   : a
Nation   : a
Gewicht: 1

Verlierer des Kampfes
Nachname   : Unbekannt
Vorname   : Unbekannt
Nation   : Unbekannt
Gewicht: 100

*** Precondition violated: Der Verlierer ist nicht in der Liste enthalten!
********** Fehler beim Bearbeiten der Boxrangliste **********
*** Programm wird beendet wegen des obigen Fehlers ***

Ansonsten, d.h. wenn die Eingaben den Forderungen entsprechen, sehen die Ausgaben wie in der 7. Übung aus.

Anmerkung: Die vollständige Implementierung steht im Netz zur Verfügung.