Programmierung **WS 2000/2001**

Prof. H. Lichter
Th. von der Maßen

Übung 10
Musterlösung

Aufgabe 10.1

a)

```
INTERFACE CD;

(* Schnittstelle des abstrakten Datentyps CD.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)

IMPORT Track;   (* Importiere den ADT Track *)

TYPE T <: REFANY;

PROCEDURE GetInterpret(cd: T): TEXT;
(* Liefert den Interpreten einer CD *)

PROCEDURE SetInterpret(VAR cd: T; interpret: TEXT);
(* Setzt den Interpreten der angegebenen CD *)

PROCEDURE GetTitle(cd: T): TEXT;
(* Liefert den Titel der übergebenen CD *)

PROCEDURE SetTitle(VAR cd: T; title: TEXT);
(* Setzt den Titel der übergebenen CD *)

PROCEDURE GetTotalTime(cd: T): CARDINAL;
(* Liefert die Gesamtspielzeit der CD in Sekunden *)

PROCEDURE AddTrack(VAR cd: T; track: Track.T);
(* Fügt einen Track der übergebenen CD hinzu *)

PROCEDURE RemoveTrack(VAR cd: T);
(* Löscht den letzten Track der übergebenen CD *)

PROCEDURE GetTrackCount(cd: T): CARDINAL;
(* Liefert die Anzahl der Tracks der übergebenen CD *)

PROCEDURE PrintTracklist(cd: T);
(* Zeigt die Trackliste der CD auf dem Bildschirm an *)

PROCEDURE CreateCD(): T;
(* Erstellt eine neue CD und initalisiert diese *)

END CD.
```

```
INTERFACE Track;

(* Schnittstelle des abstrakten Datentyps Track.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)

TYPE T <: REFANY;

PROCEDURE GetName(track: T): TEXT;
(* Liefert den Namen des Tracks *)

PROCEDURE SetName(VAR track: T; name: TEXT);
(* Setzt den Namen des Tracks auf den übergebenen String *)

PROCEDURE GetLength(track: T): CARDINAL;
(* Liefert die Länge des Tracks in Sekunden *)

PROCEDURE SetLength(VAR track: T; length: CARDINAL);
(* Setzt die Länge des übergebenen Tracks in Sekunden *)

PROCEDURE CreateTrack(): T;
(* Erstellt einen neuen Track und initialisiert diesen *)

END Track.
```

```
INTERFACE Diskographie;

(* Schnittstelle des Objektmoduls zur Verwaltung einer
Diskographie.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)

IMPORT CD;  (* Importiere den ADT CD *)

PROCEDURE Init();
(* Initialisiert die Diskographie. Diese enthält dann keine CDs
*)

PROCEDURE AddCD(cd: CD.T);
(* Fügt die übergebene CD der Diskographie hinzu *)

PROCEDURE GetCD(index: CARDINAL): CD.T;
(* Liefert die CD der Diskographie mit dem übergebenen Index *)

PROCEDURE RemoveCD(pos: CARDINAL);
(* Löscht die CD an der übergebenen Indexposition *)
```

```
PROCEDURE IsFull(): BOOLEAN;
(* Prüft, ob die Diskographie voll ist *)

PROCEDURE IsEmpty(): BOOLEAN;
(* Prüft, ob die Diskographie leer ist *)

PROCEDURE Print();
(* Listet den Inhalt der Diskographie auf dem Bildschirm *)

PROCEDURE PrintWithTracklist();
(* Listet den Inhalt der Diskographie inklusive der Trackliste
jeder CD auf dem Bildschirm *)

END Diskographie.
```

b)

```
MODULE CD;

(* Implementierung des abstrakten Datentyps CD.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)

IMPORT SIO;
IMPORT Track;    (* Importiere den ADT Track *)


(* Die Trackliste wird als lineare Liste realisiert *)
TYPE Tracklist = REF RECORD
                   track: Track.T;
                   next: Tracklist;
                 END;

(* Eine CD besteht aus einem Interpreten, einem Titel und einer
Liste von
Musikstücken *)
     REVEAL T = BRANDED REF RECORD
                  interpret: TEXT;
                  title: TEXT;
                  tracklist: Tracklist;
                END;

PROCEDURE GetInterpret(cd: T): TEXT =
(* Liefert den Interpreten einer CD *)
BEGIN
  RETURN (cd^.interpret);
END GetInterpret;


PROCEDURE SetInterpret(VAR cd: T; interpret: TEXT) =
(* Setzt den Interpreten der angegebenen CD *)
BEGIN
  cd^.interpret := interpret;
END SetInterpret;


PROCEDURE GetTitle(cd: T): TEXT =
(* Liefert den Titel der übergebenen CD *)
BEGIN
  RETURN (cd^.title);
END GetTitle;


PROCEDURE SetTitle(VAR cd: T; title: TEXT) =
(* Setzt den Titel der übergebenen CD *)
BEGIN
  cd^.title := title;
```

```
END SetTitle;


PROCEDURE GetTotalTime(cd: T): CARDINAL =
(* Liefert die Gesamtspielzeit der CD in Sekunden *)
VAR totaltime: CARDINAL;
    tlist : Tracklist;
    t : Track.T;
BEGIN
  tlist := cd^.tracklist;
  totaltime := 0;
  (* Summiere die Längen der einzelnen Tracks der CD *)
  WHILE (tlist # NIL) DO
    t := tlist^.track;
    totaltime := totaltime + Track.GetLength(t);
    tlist := tlist^.next;
  END;
  RETURN totaltime;
END GetTotalTime;

PROCEDURE AddTrack(VAR cd: T; track: Track.T) =
(* Fügt einen Track der übergebenen CD hinzu. Der neue Track wird
an das Ende der Trackliste angehängt. *)
VAR newtrack, actual, prev: Tracklist;
BEGIN
  newtrack := NEW(Tracklist);
  newtrack^.track := track;
  newtrack^.next := NIL;

  (* Falls Trackliste leer *)
  IF (cd^.tracklist = NIL) THEN
    cd^.tracklist := newtrack;
  ELSE
    (* Suche Ende der Trackliste *)
    actual := cd^.tracklist;
    prev := cd^.tracklist;

    WHILE (actual # NIL) DO
      prev := actual;
      actual := actual^.next;
    END;

    prev^.next := newtrack;
  END;
END AddTrack;


PROCEDURE RemoveTrack(VAR cd: T) =
(* Löscht den letzten Track der übergebenen CD *)
VAR actual, prev: Tracklist;
BEGIN
  IF (cd^.tracklist = NIL) THEN
    SIO.PutLine("Trackliste ist leer!");
  ELSIF (cd^.tracklist.next = NIL) THEN
    cd^.tracklist := NIL;
  ELSE
    actual := cd^.tracklist;
    prev := cd^.tracklist;
    WHILE (actual^.next # NIL) DO
      prev := actual;
      actual := actual^.next;
    END;
    prev^.next := NIL;
  END;
END RemoveTrack;


PROCEDURE GetTrackCount(cd: T): CARDINAL =
(* Liefert die Anzahl der Tracks der übergebenen CD *)
VAR count: CARDINAL;
    tl: Tracklist;
BEGIN
  count := 0;
  tl := cd^.tracklist;

  WHILE (tl # NIL) DO
    count := count + 1;
    tl := tl^.next;
  END;
  RETURN count;
END GetTrackCount;


PROCEDURE PrintTracklist(cd: T) =
(* Zeigt die Trackliste der CD mit Namen und Länge der Tracks
auf dem Bildschirm an *)
VAR tl: Tracklist;
    count : CARDINAL;
BEGIN
  tl := cd^.tracklist;
  count := 1;
  WHILE (tl # NIL) DO
    SIO.PutInt(count); SIO.PutText(". ");
    SIO.PutText(Track.GetName(tl^.track) & " Dauer: ");
    SIO.PutInt(Track.GetLength(tl^.track)); SIO.Nl();
    tl := tl^.next;
    count := count + 1;
  END;
END PrintTracklist;


PROCEDURE CreateCD(): T =
(* Erstellt eine neue CD und initalisiert diese *)
VAR newcd: T;
BEGIN
  newcd := NEW(T);
```

```
  newcd^.interpret := "";   (* Initialisiere Interpreten *)
  newcd^.title := "";       (* Initialisiere Titel *)
  newcd^.tracklist := NIL;  (* Initialisiere Trackliste *)
  RETURN newcd;
END CreateCD;

BEGIN
END CD.
```

```
MODULE Track;

(* Modulrumpf des abstrakten Datentyps Track.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)


(* Ein Track besteht aus einem Namen und der Länge in Sekunden *)
TYPE REVEAL T = BRANDED REF RECORD
                  name: TEXT;
                  length: CARDINAL;
                END;


PROCEDURE GetName(track: T): TEXT =
(* Liefert den Namen des Tracks *)
BEGIN
  RETURN (track^.name);
END GetName;


PROCEDURE SetName(VAR track: T; n: TEXT) =
(* Setzt den Namen des Tracks auf den übergebenen String *)
BEGIN
  track^.name := n;
END SetName;


PROCEDURE GetLength(track: T): CARDINAL =
(* Liefert die Länge des Tracks in Sekunden *)
BEGIN
  RETURN (track^.length);
END GetLength;


PROCEDURE SetLength(VAR track: T; l: CARDINAL) =
(* Setzt die Länge des übergebenen Tracks in Sekunden *)
BEGIN
  track^.length := l;
END SetLength;
```

```
PROCEDURE CreateTrack(): T =
(* Erstellt einen neuen Track und initialisiert diesen *)
VAR newtrack: T;
BEGIN
  newtrack := NEW(T);
  newtrack^.name := "";   (* Initialisiere den Namen *)
  newtrack^.length := 0;  (* Initialisiere die Länge *)
  RETURN newtrack;
END CreateTrack;

BEGIN
END Track.
```

```
MODULE Diskographie;

(* Dieses Module implementiert die Diskographie-Verwaltung, wie
in der Schnittstelle
   Diskographie definiert. Die Verwaltung erfolgt mit Hilfe eines
Feldes begrenzter Kapazitaet.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 19.12.00
*)

IMPORT SIO;
IMPORT CD;   (* Importiere den ADT CD *)

CONST Max = 5;

TYPE DiskographieIndex = [0 .. Max];
     Diskographieverwaltung = ARRAY [1 .. Max] OF CD.T;

VAR index : DiskographieIndex;
    verwaltung: Diskographieverwaltung;


PROCEDURE Init()=
(* Initialisiert die Diskographie. Diese enthält dann keine CDs
*)
BEGIN
  index := 0;
END Init;


PROCEDURE AddCD(cd: CD.T)=
(* Fügt die übergebene CD der Diskographie hinzu *)
BEGIN
  (* CD wird im nächsten Speicherplatz abgelegt *)
  IF NOT IsFull() THEN
    index := index + 1;
    verwaltung[index] := cd;
```

```
  ELSE
    SIO.PutLine("Diskographie ist voll!!!");
  END;
END AddCD;


PROCEDURE RemoveCD(pos: CARDINAL)=
(* Löscht die CD an der übergebenen Indexposition *)
BEGIN
  IF IsEmpty() THEN
    SIO.PutLine("Keine CD zum loeschen vorhanden!!!");
  ELSE
    IF (pos > 0) AND (pos <= index) THEN
      FOR i:=pos TO index - 1 DO
        verwaltung[i] := verwaltung[i+1];
      END;
      (* Indexposition wird um 1 zurückgesetzt *)
      index := index - 1;
    END;
  END;
END RemoveCD;


PROCEDURE IsFull(): BOOLEAN =
(* Prüft, ob die Diskographie voll ist *)
BEGIN
  RETURN (index = Max);
END IsFull;


PROCEDURE IsEmpty(): BOOLEAN =
(* Prüft, ob die Diskographie leer ist *)
BEGIN
  RETURN (index = 0);
END IsEmpty;


PROCEDURE Print()=
(* Listet den Inhalt der Diskographie auf dem Bildschirm *)
BEGIN
  SIO.PutLine("------------------- Diskographieverwaltung ------
----------------------------------------");
  SIO.Nl();
  FOR i:=1 TO index DO
    SIO.PutInt(i); SIO.PutText(".   ");
    SIO.PutText("Titel: ");
SIO.PutText(CD.GetTitle(verwaltung[i]));
    SIO.PutText("   Interpret: ");
SIO.PutText(CD.GetInterpret(verwaltung[i]));
    SIO.PutText("   Anzahl Tracks: ");
SIO.PutInt(CD.GetTrackCount(verwaltung[i]));
    SIO.PutText("   Gesamtspieldauer: ");
SIO.PutInt(CD.GetTotalTime(verwaltung[i]));
    SIO.Nl();
  END;
  SIO.PutLine("------------------------------------------------
----------------------------------------");
END Print;


PROCEDURE PrintWithTracklist()=
(* Listet den Inhalt der Diskographie mit den Tracks jeder CD auf
dem Bildschirm *)
BEGIN
  SIO.PutLine("------------------- Diskographieverwaltung ------
----------------------------------------");
  SIO.Nl();
  FOR i:=1 TO index DO
    SIO.PutInt(i); SIO.PutText(".   ");
    SIO.PutText("Titel: ");
SIO.PutText(CD.GetTitle(verwaltung[i]));
    SIO.PutText("   Interpret: ");
SIO.PutText(CD.GetInterpret(verwaltung[i]));
    SIO.PutText("   Anzahl Tracks: ");
SIO.PutInt(CD.GetTrackCount(verwaltung[i]));
    SIO.PutText("   Gesamtspieldauer: ");
SIO.PutInt(CD.GetTotalTime(verwaltung[i]));
    SIO.Nl();
    SIO.PutLine("Tracks:");
    CD.PrintTracklist(verwaltung[i]);
    SIO.Nl();
  END;
  SIO.PutLine("------------------------------------------------
----------------------------------------");
END PrintWithTracklist;

PROCEDURE GetCD(i: CARDINAL): CD.T =
(* Liefert die CD der Diskographie mit dem übergebenen Index *)
BEGIN
  IF (i > 0) AND (i <= index) THEN
    RETURN (verwaltung[i]);
  ELSE
    SIO.PutLine("Keine gueltige CD angegeben");
    RETURN NIL;
  END;
END GetCD;

BEGIN
END Diskographie.
```

```
MODULE CDVerwaltung EXPORTS Main;

(* Hauptprogramm der CD-Verwaltung. Das Programm stellt ein Menü
bereit, welches es ermöglicht CDs in die Diskographie einzufügen
```

```
und zu löschen. Ebenso können Musikstücke zu den verschiedenen
enthaltenen CDs eingefügt und auch wieder gelöscht werden.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3 Windows 2000
   Erstellt        : 19.12.00  Letzte Aenderung: 02.01.01
*)

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

VAR dummy : TEXT;
    auswahl : CHAR;
    cd : CD.T;
    track: Track.T;
    nummer: CARDINAL;


PROCEDURE PrintMenu()=
(* Hilfsprozedur, die das Befehlsmenue ausgibt *)
BEGIN
  SIO.Nl();
  SIO.PutLine("F : Fuege neue CD ein");
  SIO.PutLine("L : Loesche CD");
  SIO.PutLine("Z : Zeige Diskographie an");
  SIO.PutLine("T : Fuege Track in CD ein");
  SIO.PutLine("X : Loesche Track von CD");
  SIO.PutLine("E : Exit");
  SIO.Nl();
END PrintMenu;


BEGIN
  Diskographie.Init();   (* Initialisiere die Diskographie *)

  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 *)
    |  'F', 'f' => cd := CD.CreateCD();
                   SIO.PutText("Bitte geben Sie den CD-Titel ein:
");
                   CD.SetTitle(cd, SIO.GetLine());
                   SIO.PutText("Bitte geben Sie den Interpreten
ein: ");
                   CD.SetInterpret(cd, SIO.GetLine());
                   Diskographie.AddCD(cd);

    (* Löschen der letzten CD *)
    |  'L', 'l' => Diskographie.Print();
                   SIO.PutText("Geben Sie die CD-Nummer ein: ");
                   nummer := SIO.GetInt();
                   dummy := SIO.GetLine();
                   Diskographie.RemoveCD(nummer)

    (* Ausgeben der aktuellen Diskographie *)
    |  'Z', 'z' => Diskographie.PrintWithTracklist();

    (* Einen Track einer CD hinzufügen *)
    |  'T', 't' => Diskographie.Print();
                   SIO.PutText("Geben Sie die CD-Nummer ein: ");
                   nummer := SIO.GetInt();
                   dummy := SIO.GetLine();
                   cd := Diskographie.GetCD(nummer);
                   IF (cd # NIL) THEN
                     REPEAT
                       track := Track.CreateTrack();
                       SIO.PutText("Geben sie den Namen des
Tracks ein: ");
                       Track.SetName(track, SIO.GetLine());
                       SIO.PutText("Geben Sie die Laenge des
Tracks in Sekunden ein: ");
                       Track.SetLength(track, SIO.GetInt());
                       dummy := SIO.GetLine();
                       CD.AddTrack(cd, track);
                       SIO.PutText("Moechten Sie einen weiteren
Track eingeben (j/n)?");
                       auswahl := SIO.GetChar();
                       dummy := SIO.GetLine();
                     UNTIL (auswahl = 'n') OR (auswahl = 'N');
                   END;

    (* Einen Track von einer CD löschen *)
    |  'X', 'x' => Diskographie.Print();
                   SIO.PutText("Geben Sie die CD-Nummer ein: ");
                   nummer := SIO.GetInt();
                   dummy := SIO.GetLine();
                   cd := Diskographie.GetCD(nummer);
                   IF (cd # NIL) THEN
                     REPEAT
                       CD.RemoveTrack(cd);
                       SIO.PutText("Moechten Sie einen weiteren
Track loeschen (j/n)?");
                       auswahl := SIO.GetChar();
                       dummy := SIO.GetLine();
                     UNTIL (auswahl = 'n') OR (auswahl ='N');
```

```
                    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 CDVerwaltung.
```