Klausur ERP-Systeme und ERP 2 am 10. Februar 2012

(B_Inf2.0 v361, B_Winf1.0 v362, IAW5.0 552 – 120 Minuten)

Aufgabe 1 *(10% Bewertungsanteil)*

Von den nachfolgenden sechzehn Aussagen sind leider vier falsch, bitte markiere jene vier fehlerhaften Aussagen, wobei mehr als vier Markierungen zu Punktabzug führt.

- ABAP ermöglicht mehrsprachige Anwendungen.
- ABAP enthält SQL-Einbettung.
- ABAP ist plattformunabhängig.
- ABAP ist datenbankunabhängig.
- ABAP richtet sich an Wiederverwendbarkeit von Code- und Datenobjekten aus.
- ABAP-Programme liegen auch als Quelltext vor.
- ABAP-Programme sind mandantenabhängig.
- In ABAP erfolgt eine automatische Typkonvertierung bei Wertzuweisung, wenn der Wert des Datenobjektes es zulässt.
- Der Inhalt von internen Tabellen befindet sich im Arbeitsspeicher, er steht demnach über die Laufzeit des Programms zur Verfügung.
- Typen interner Tabellen sind Standard-Tabellen, sortierte Tabellen und Hash-Tabellen.
- Unterprogramme können nicht auf übergeordnete („globale“) Variablen des Programms zugreifen.
- Innerhalb von Unterprogrammen können lokale Datenobjekte definiert werden.
- Der rekursive Aufruf von Unterprogrammen ist möglich.
- Eigene Pakete ermöglichen die Gruppierung verschiedener Entwicklungen.
- Transportaufträge dienen unter anderem zur Gruppierung von Entwicklungsaufgaben.
- Eine typische SAP-Systemlandschaft besteht aus Development und Production System.

Aufgabe 2 *(10% Bewertungsanteil)*

Entwickle jeweils ein ABAP-Codefragment (beinhaltend Datendeklaration und -verarbeitung) für nachfolgende vier Datentypen. Dabei sind folgende charakteristische Verwendungen zu berücksichtigen: Anzahl Minuten seit Mitternacht, Anzahl Tage zwischen zwei Kalenderdaten, Summe zweier Geldbeträge und Zählschleife für genau zehn Durchläufe.

1. D

2. I

3. P

4. T

Aufgabe 3 *(10% Bewertungsanteil)*

Forme folgendes Programmfragment in eine syntaktisch korrekte Fassung um:

```
DATA A TYPE F.
DATA B TYPE F.

IF ( A * 2 ) = ( B / 2 ).
  WRITE 'IDENTISCH'.
ENDIF.
```

Welche Werte beinhalten die Variablen *MyString2*, *MyString3*, *MyString4* und *MyString5* nach Ausführung des folgenden Programmfragments (nachfolgende Leerzeichen in den Ergebnissen bleiben unberücksichtigt) ?

```
DATA myString1(42) TYPE C VALUE 'Der Dozent wird am UCC geschult'.
DATA myString2(21) TYPE C.
DATA myString3(21) TYPE C.
DATA myString4(21) TYPE C VALUE '3'.
DATA myString5(21) TYPE C VALUE 'Magdeburg'.

DATA i TYPE I VALUE 3.
DATA j TYPE I VALUE 19.

myString2 = myString1+4(6).

myString3 = myString1+j(i).

myString4 = myString4 + j.

myString5+5(4) = 'dorf'.
```

Aufgabe 4 *(5% Bewertungsanteil)*

Markiere die drei richtigen Aussagen zu Funktionsbausteinen *(mehr als drei Markierungen führt zu Punktabzug)*:

- Kapselung von Quellcode.
- Funktionsbausteine können auf übergeordnete Variablen zugreifen.
- Organisation von Funktionsbausteinen in Funktionsgruppen.
- Import-Parameter: Dies sind die Rückgabeparameter des Funktionsbausteins.
- Export-Parameter: Es erfolgt die Angabe der Eingabeparameter des Funktionsbausteins.
- Changing-Parameter: Es handelt sich um Parameter, die gleichzeitig als Import- und Export-Parameter dienen.

Markiere die drei richtigen Datentypen im Sinne des ABAP Dictionarys *(mehr als drei Markierungen führt zu Punktabzug)*:

- Datenbanktabelle
- Datenelement
- Domäne
- Sperrobjekt
- Struktur
- Suchhilfe
- Tabellentyp
- Typgruppe
- View

Aufgabe 5 *(5% Bewertungsanteil)*

Ordne nachfolgenden Aussagen bzw. Funktionen den jeweils am ehesten passenden der folgenden vierzehn SAP-Begriffe zu: ALV-Grid, BAPI, Berechtigungsprofil, HTML-Viewer, Nummernkreisobjekt, Paket, Rolle, Sperrobjekt, Suchhilfe, Tabstrip-Control, Transportauftrag, User-Exit, Verbuchungsbaustein und Web-Dynpro:

1. Dient dazu, ein Menü auszuwählen und dazu passend ein Berechtigungsprofil zu erzeugen.

2. Element des Berechtigungssystems, gewährt den Benutzern Zugriff auf das System.

3. Ermöglicht Registerkarten auf Dynpros darzustellen.

4. Nach betriebswirtschaftlicher Sichtweise gekapselte Funktionsbausteine.

5. Oberflächenelement, mit dem tabellarische Daten in Anwendungen angezeigt werden können.

6. Objekt des ABAP Dictionary, mit dem Eingabehilfen (F4-Hilfen) definiert werden können.

7. Synchronisation des gleichzeitigen Zugriffs zweier Benutzer auf denselben Datenbestand.

8. Zeitpunkt im SAP-Programm, zu dem ein kundeneigener Programmteil aufgerufen werden kann.

Aufgabe 6 *(30% Bewertungsanteil)*

Gegeben sei im Data Dictionary die Datenbanktabelle SPFLI („Flugverbindungen“):

- CARRID: Fluggesellschaft (2-stellig alphanumerisch, Bestandteil des Primärschlüssels)
- CONNID: Einzelflugverbindung (4-stellig numerisch, Bestandteil des Primärschlüssels)
- AIRPFROM: Abflughafen (3-stellig alphanumerisch)
- AIRPTO: Zielflughafen (3-stellig alphanumerisch)
- DEPTIME: Abflugzeit (4-stellig numerisch)
- ARRTIME: Ankunftzeit (4-stellig numerisch)

Entwickelt wurde eine Transaktion zur auf Anzeigen und Ändern beschränkte Stammdatenpflege, die einerseits auf der obigen Datenbanktabelle basiert, und die andererseits zwei Dynpros („Auswahl“ über den Primärschlüssel und „Anzeigen/Ändern“ ohne den Primärschlüssel) beinhaltet. Spezifiziert ist der ABAP-Code der einzelnen Module (Include-Dateien) inklusive dem TOP-Include, sowie der Ablauflogik-Code der beiden Dynpros. Auf dem „Auswahl“-Dynpro wird zusätzlich ein Table-Control zum Markieren genau einer Flugverbindung verwendet, wobei das spezielle Ereignis „Doppelklick in eine Tabellenzeile“ ebenfalls berücksichtigt werden soll.

Doch Halt, der Programmierer scheint beim Spezifizieren ganz gehörig durcheinander geraten zu sein, Helft ihm durch Ankreuzen, welche Codezeilen erforderlich sind, und welche Codezeilen nicht erforderlich (bzw. falsch oder überflüssig) sind.

*Dabei gelten folgende „Spielregeln“: Sowohl in der Spalte „erforderlich“ als auch in der Spalte „nicht erforderlich“ sind genau **72** Codezeilen zu markieren. Jedes jeweils darüber hinausgehende Kreuz wird als Minuspunkt gewertet.*

Ablauflogik Dynpro 0100 („Auswahl“)	erforderlich	nicht erforderlich
PROCESS BEFORE OUTPUT.		
MODULE status_0100.		
MODULE user_command_0100.		
LOOP WITH CONTROL flights.		
LOOP WITH CONTROL my_alv.		
MODULE fill_table_control.		
MODULE read_table_control.		
MODULE status_0100.		
MODULE user_command_0100.		
ENDLOOP.		
MODULE status_0100.		
MODULE user_command_0100.		
PROCESS AFTER INPUT.		
MODULE status_0100.		
MODULE user_command_0100.		
LOOP WITH CONTROL flights.		
LOOP WITH CONTROL my_alv.		
MODULE fill_table_control.		
MODULE read_table_control.		
MODULE status_0100.		
MODULE user_command_0100.		
ENDLOOP.		
MODULE status_0100.		
MODULE user_command_0100.		

Ablauflogik Dynpro 0200 („Anzeigen/Ändern“)	erforderlich	nicht erforderlich
PROCESS BEFORE OUTPUT.		
MODULE fill_table_control.		
MODULE read_table_control.		
MODULE status_0200.		
MODULE user_command_0200.		
PROCESS AFTER INPUT.		
MODULE fill_table_control.		
MODULE read_table_control.		
MODULE status_0200.		
MODULE user_command_0200.		

TOP-Include	erforderlich	nicht erforderlich
`PROGRAM zz_dynpro.`		
`CONTROLS flights TYPE TABLEVIEW USING SCREEN 100.`		
`CONTROLS flights TYPE TABLEVIEW USING SCREEN 200.`		
`CONTROLS mytabstrip TYPE TABSTRIP.`		
`TABLES spfli.`		
`TABLES zz_spfli.`		
`DATA: ok_code LIKE sy-ucomm.`		
`DATA: wa_flug TYPE spfli.`		
`DATA: wa_flug TYPE zz_spfli.`		
`DATA: itab TYPE TABLE OF spfli.`		
`DATA: itab TYPE TABLE OF zz_spfli.`		
`DATA: my_container TYPE REF TO cl_gui_custom_container.`		
`DATA: my_alv TYPE REF TO cl_gui_alv_grid.`		
`DATA: my_html TYPE REF TO cl_gui_html_viewer.`		

Include zu MODULE status_0100	erforderlich	nicht erforderlich
MODULE status_0100 INPUT.		
MODULE status_0100 OUTPUT.		
SET PF-STATUS '100'.		
SET PF-STATUS '200'.		
SET TITLEBAR '100'.		
SET TITLEBAR '200'.		
CLEAR ok_code.		
SELECT * FROM spfli INTO TABLE itab.		
SELECT * FROM spfli INTO CORRESPONDING FIELDS OF TABLE itab.		
SELECT * FROM zz_spfli INTO TABLE itab.		
SELECT * FROM zz_spfli INTO CORRESPONDING FIELDS OF TABLE itab.		
DESCRIBE TABLE itab LINES flights-lines.		
DESCRIBE TABLE itab LINES my_alv-lines.		
ENDMODULE.		

Include zu MODULE user_command_0100	erforderlich	nicht erforderlich
`MODULE user_command_0100 INPUT.`		
`MODULE user_command_0100 OUTPUT.`		
`  CASE ok_code.`		
`    WHEN 'BACK'.`		
`      LEAVE PROGRAM.`		
`    WHEN 'SELECT'`		
`    WHEN 'SELECT' OR 'PICK'.`		
`      SELECT SINGLE * FROM spfli INTO wa_flug` `                    WHERE carrid = spfli-carrid.`		
`      SELECT SINGLE * FROM zz_spfli INTO wa_flug` `                    WHERE carrid = spfli-carrid.`		

`      SELECT SINGLE * FROM spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid.`		

`      SELECT SINGLE * FROM zz_spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid.`		

`      SELECT SINGLE * FROM spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid AND`
`                          airpfrom = spfli-airpfrom AND`
`                          airpto = spfli-airpto.`		

`      SELECT SINGLE * FROM zz_spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid AND`
`                          airpfrom = spfli-airpfrom AND`
`                          airpto = spfli-airpto.`		

`      IF sy-dbcnt = 0.`		
`      IF sy-dbcnt = 1.`		
`        LEAVE TO SCREEN 100.`		
`        LEAVE TO SCREEN 200.`		
`      ENDIF.`		
`  ENDCASE.`		
`ENDMODULE.`		

Include zu MODULE status_0200	erforderlich	nicht erforderlich
`MODULE status_0200 INPUT.`		
`MODULE status_0200 OUTPUT.`		
`  SET PF-STATUS '100'.`		
`  SET PF-STATUS '200'.`		
`  SET TITLEBAR '100'.`		
`  SET TITLEBAR '200'.`		
`  CLEAR ok_code.`		
`  wa_flug = spfli.`		
`  spfli = wa_flug.`		
`ENDMODULE.`		

Include zu MODULE user_command_0200	erforderlich	nicht erforderlich
MODULE user_command_0200 INPUT.		
MODULE user_command_0200 OUTPUT.		
CASE ok_code.		
WHEN 'LEAVE'.		
LEAVE PROGRAM.		
WHEN 'BACK'.		
LEAVE TO SCREEN 100.		
LEAVE TO SCREEN 200.		
WHEN 'SAVE'.		
MODIFY spfli FROM spfli.		
MODIFY zz_spfli FROM spfli.		
MODIFY spfli FROM wa_flug.		
MODIFY zz_spfli FROM wa_flug.		
ENDCASE.		
spfli = wa_flug.		
wa_flug = spfli.		
ENDMODULE.		

Include zu MODULE fill_table_control	erforderlich	nicht erforderlich
`MODULE fill_table_control OUTPUT.`		
`MODULE read_table_control OUTPUT.`		
`  READ TABLE itab INTO zz_spfli INDEX flights-current_line.`		
`  READ TABLE itab INTO zz_spfli INDEX sy-index.`		
`  READ TABLE spfli INTO zz_spfli INDEX flights-current_line.`		
`  READ TABLE spfli INTO zz_spfli INDEX sy-index.`		
`  READ TABLE zz_spfli INTO zz_spfli INDEX flights-current_line.`		
`  READ TABLE zz_spfli INTO zz_spfli INDEX sy-index.`		
`ENDMODULE.`		

Include zu MODULE read_table_control	erforderlich	nicht erforderlich
`MODULE fill_table_control INPUT.`		
`MODULE read_table_control INPUT.`		
`  DATA cursorline TYPE i.`		
`  GET CURSOR cursorline.`		
`  GET CURSOR LINE cursorline.`		
`  GET LINE cursorline.`		
`  IF sy-subrc = 0.`		
`    cursorline = flights-top_line + cursorline - 1.`		
`    cursorline = flights-top_line + cursorline.`		
`    cursorline = flights-top_line + cursorline + 1.`		
`  ELSE.`		
`    cursorline = -1.`		
`    cursorline = 0.`		
`    cursorline = 1.`		
`  ENDIF.`		
`  IF ( zz_spfli-selection = 'X' ).`		
`  IF ( zz_spfli-selection = 'X' ) OR` `     ( flights-current_line = cursorline ).`		
`  IF ( zz_spfli-selection = 'X' ) OR` `     ( my_alv-current_line = cursorline ).`		
`  IF ( flights-current_line = cursorline ).`		
`  IF ( my_alv-current_line = cursorline ).`		
`    zz_spfli-carrid = spfli-carrid.`		
`    zz_spfli-connid = spfli-connid.`		
`    spfli-carrid = zz_spfli-carrid.`		
`    spfli-connid = zz_spfli-connid.`		
`  ENDIF.`		
`ENDMODULE.`		

Aufgabe 7 *(25% Bewertungsanteil)*

Entwickle eine Business Server Pages (BSP) – Applikation für eine einfache Textanalyse: Erste *n* Zeichen, mittlere *n* Zeichen und letzte *n* Zeichen einer statischen Zeichenkette von genau 64 Zeichen. Die BSP-Applikation soll aus vier Seiten bestehen. Auf der ersten Seite (Haupt- bzw. Startseite) werden sowohl die Zeichenkette und die Anzahl der Zeichen („*n*“) eingegeben, als auch wird über drei Buttons entweder zur zweiten, dritten oder vierten Seite verzweigt. Die zweite Seite beinhaltet die „Berechnung“ und Ausgabe der ersten *n* Zeichen der Zeichenkette, die dritte und vierte Seite jeweils analog die mittleren *n* Zeichen (nur wenn *n* gerade, ansonsten Fehlermeldung) und letzten *n* Zeichen. Von der zweiten, dritten und vierten Seite soll jeweils über Buttons die Möglichkeit bestehen, direkt zur Seite der beiden jeweils übrigen Zeichenkettenoperationen zu verzweigen, jedoch nicht zurück zur Haupt- bzw. Startseite.

Spezifiziere zur Lösung dieser Aufgabe insgesamt *fünf* Code-Bestandteile: Das Layout der obigen *vier* Seiten und *einen* Event-Handler, der für alle *vier* Seiten gemeinsam geeignet ist. Welche Einstellungen sind bei der Entwicklung zusätzlich noch erforderlich ?

Aufgabe 7 (Fortsetzung)

Aufgabe 7 (Fortsetzung)

Aufgabe 7 (Fortsetzung)

Aufgabe 7 (Fortsetzung)

Das Team der FH und PTL Wedel wünscht viel Erfolg