Klausur ERP-Systeme und ERP 2 am 18. Februar 2013

(B_Inf2.0 v361, B_Winf1.0 v362, IAW6.1 552 – 120 Minuten)

Aufgabe 1 *(15% Bewertungsanteil)*

1. Kennzeichne die richtige Behauptung (1 Kreuz von 4 Möglichkeiten)	
ABAP ist datenbankabhängig und plattformabhängig	
ABAP ist datenbankabhängig und plattformunabhängig	
ABAP ist datenbankunabhängig und plattformabhängig	
ABAP ist datenbankunabhängig und plattformunabhängig	

2. Kennzeichne die richtige Behauptung (1 Kreuz von 4 Möglichkeiten)	
ABAP richtet sich weder an die Wiederverwenbarkeit von Code- noch von Datenobjekten aus	
ABAP richtet sich stark an die Wiederverwendbarkeit von Codeobjekten aus	
ABAP richtet sich stark an die Wiederverwendbarkeit von Datenobjekten aus	
ABAP richtet sich stark an die Wiederverwendbarkeit von Code- und Datenobjekten aus	

3. Kennzeichne die richtigen Behauptungen (2 Kreuze von 4 Möglichkeiten)	
ABAP-Programme liegen nur in compilierter Form vor	
ABAP-Programme liegen auch als Quelltext vor	
ABAP-Programme werden beim ersten Aufruf compiliert	
ABAP-Programme werden bei jedem Aufruf compiliert	

4. Kennzeichne die richtigen Behauptungen (2 Kreuze von 4 Möglichkeiten)	
Das Repository ist mandantenunabhängig	
Das Repository ist mandantenabhängig	
Die Anwendungsdaten sind mandantenunabhängig	
Die Anwendungsdaten sind mandatenabhängig	

5. Transporte in einer typischen SAP-Systemlandschaft erfolgen: (2 Kreuze von 6 Möglichkeiten)	
Von Development System nach Production System	
Von Development System nach Quality Assurance System	
Von Production System nach Development System	
Von Production System nach Quality Assurance System	
Von Quality Assurance System nach Development System	
Von Quality Assurance System nach Production System	

6. Kennzeichne die richtige Behauptung (1 Kreuz von 3 Möglichkeiten)	
In ABAP ist die Konvertierung von String nach Integer nicht möglich	
In ABAP ist die Konvertierung von String nach Integer in Abhängigkeit vom Inhalt möglich	
In ABAP ist die Konvertierung von String nach Integer generell möglich	

7. Markiere die Datentypen im Sinne des ABAP Dictionarys (3 Kreuze von 9 Möglichkeiten) *(doppelt gewertet)*	
Datenbanktabelle	
Datenelement	
Domäne	
Sperrobjekt	
Struktur	
Suchhilfe	
Tabellentyp	
Typgruppe	
View	

8. Kennzeichne die richtigen Behauptungen (2 Kreuze von 4 Möglichkeiten)	
Der Inhalt von Datenbanktabellen steht nicht über die Programmlaufzeit hinaus zur Verfügung	
Der Inhalt von Datenbanktabellen steht über die Programmlaufzeit hinaus zur Verfügung	
Der Inhalt von internen Tabellen steht nicht über die Programmlaufzeit hinaus zur Verfügung	
Der Inhalt von internen Tabellen steht über die Programmlaufzeit hinaus zur Verfügung	

9. Kennzeichne die richtigen Behauptungen (2 Kreuze von 4 Möglichkeiten)	
Interne Tabellen haben eine feste Länge	
Interne Tabellen können prinzipiell beliebig lang sein	
Interne Tabellen können nicht aus internen Tabellen bestehen	
Interne Tabellen können auch aus internen Tabellen bestehen	

10. Kennzeichne die richtigen Behauptungen (3 Kreuze von 6 Möglichkeiten) *(doppelt gewertet)*	
Unterprogramme können nicht auf übergeordnete ("globale") Variablen des Programms zugreifen.	
Innerhalb von Unterprogrammen können keine lokalen Variablen definiert werden.	
Der rekursive Aufruf von Unterprogrammen ist nicht möglich	
Unterprogramme können auf übergeordnete ("globale") Variablen des Programms zugreifen.	
Innerhalb von Unterprogrammen können lokale Variablen definiert werden.	
Der rekursive Aufruf von Unterprogrammen ist möglich.	

11. Markiere aus der Sicht des Funktionsbausteins die richtigen Aussagen zu Funktionsbausteinen: (3 Kreuze von 6 Möglichkeiten) *(doppelt gewertet)*	
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.	

12. Das Abfangen von Laufzeitfehlern unter Umgehung der objektorientierten Programmierung ist möglich mit: (1 Kreuz von 5 Möglichkeiten)	
Befehl CATCH SYSTEM-EXCEPTIONS	
Variable Sy-dbcnt	
Variable Sy-fdpos	
Befehl TRY CATCH	
Variable Sy-subrc	

Aufgabe 2 *(12% Bewertungsanteil)*

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

1. D

2. I

3. P

4. T

Aufgabe 3 *(8% Bewertungsanteil)*

Forme folgendes Programmfragment in eine syntaktisch korrekte Fassung um:

```
DATA X TYPE F.
DATA Y TYPE F.

IF ( X * 2 ) = ( Y / 2 ).
  WRITE 'MOINSEN'.
ENDIF.
```

Welche Werte beinhalten die Variablen *MyString1*, *MyString2*, *MyString3* und *MyString4* nach Ausführung des folgenden Programmfragments (nachfolgende Leerzeichen in den Ergebnissen bleiben unberücksichtigt) ?

```
DATA myString1(42) TYPE C VALUE 'ERP-Software und ERP-Systeme'.
DATA myString2(42) TYPE C VALUE 'ERP-Software und ERP-Systeme'.
DATA myString3(4)  TYPE C.
DATA myString4(4)  TYPE C.

DATA i TYPE I VALUE 17.
DATA j TYPE I VALUE 3.

myString1+i(j) = 'SAP'.

myString2+j(i) = '-Systeme  und SAP'.

myString3 = i.

j = i + myString3.

myString4 = j.
```

Aufgabe 4 *(15% Bewertungsanteil)*

Entwickle einen Report für eine einfache Textanalyse („Berechnung und Ausgabe“): Erste *n* Zeichen, mittlere *n* Zeichen und letzte *n* Zeichen einer statischen Zeichenkette von genau 64 Zeichen. Mittels eines „Selektionsbildschirms“ werden die Zeichenkette und die Anzahl der Zeichen („*n*“) eingegeben. Bei der Berechnung der mittleren *n* Zeichen soll eine Fehlermeldung ausgegeben werden, sofern *n* nicht gerade ist.

Aufgabe 4 (Fort.)

Aufgabe 5 *(15% Bewertungsanteil)*

Entwickle einen Report der für zwei Brüche (bestehend jeweils aus zwei natürlichen Zahlen, dem Zähler und dem Nenner) einerseits die Addition der beiden Brüche und andererseits die Multiplikation der beiden Brüche berechnet und ausgibt. Mittels eines „Selektionsbildschirms“ werden die beiden Brüche eingegeben. Dabei mögen die folgenden (mathematischen) Hinweise hilfreich sein:

1. Zum Erweitern zweier Brüche ist ein Hauptnenner erforderlich, der sich aus dem kleinsten gemeinsamen Vielfachen (kgV) der beiden Nenner bildet. Zur Berechnung des kgV zweier natürlicher Zahlen steht bereits der Funktionsbaustein KGV mit zwei Eingabeparametern (PIN1, PIN2) und einem Ausgabeparameter (POUT) zur Verfügung.

2. Zum Kürzen eines Bruches ist der größte gemeinsame Teiler (ggT) des Zählers und Nenners erforderlich. Zur Berechnung des ggT zweier natürlicher Zahlen steht bereits der Funktionsbaustein GGT mit zwei Eingabeparametern (PIN1, PIN2) und einem Ausgabeparameter (POUT) zur Verfügung.

Aufgabe 5 (Fort.)

Aufgabe 6 *(10% Bewertungsanteil)*

Ordne nachfolgenden Aussagen bzw. Funktionen den jeweils am ehesten passenden der folgenden achtzehn SAP-Begriffe zu: ABAP Editor, ALV-Grid, BAPI, Berechtigungsprofil, Class Builder, Debugger, Function Builder, Menu Painter, Nummernkreisobjekt, Object Navigator, Paket, Rolle, Screen Painter, Sperrobjekt, Suchhilfe, Transportauftrag, User-Exit und Verbuchungsbaustein:

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. Benutzerfreundliche Oberfläche, die alle Entwicklungswerkzeuge intuitiv integriert.

4. Nach betriebswirtschaftlicher Sichtweise gekapselte Funktionsbausteine.

5. Erstellung und Pflege von Dynpros.

6. Erstellung und Pflege von Menüs, Überschriften und Symbolleisten in ABAP-Programmen.

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

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

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

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

Aufgabe 7 *(20% 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 damit auch, welche Codezeilen nicht erforderlich (bzw. falsch oder überflüssig) sind.

Dabei gilt folgende „Spielregel“: Die Anzahl der erforderlichen Kreuze steht jeweils in Klammern hinter der Blocknummer, jeder richtig bearbeiteter Block erzielt einen Pluspunkt, jeder falsch bearbeiteter Block bleibt unberücksichtigt.

Ablauflogik Dynpro 0100 („Auswahl“)	erforder-lich	Block (Kreuzanzahl)
PROCESS BEFORE OUTPUT.	X	
MODULE status_0100.		1 (4)
MODULE user_command_0100.		1 (4)
LOOP WITH CONTROL flights.		1 (4)
LOOP WITH CONTROL my_alv.		1 (4)
MODULE fill_table_control.		1 (4)
MODULE read_table_control.		1 (4)
MODULE status_0100.		1 (4)
MODULE user_command_0100.		1 (4)
ENDLOOP.		1 (4)
MODULE status_0100.		1 (4)
MODULE user_command_0100.		1 (4)
PROCESS AFTER INPUT.	X	
MODULE status_0100.		2 (4)
MODULE user_command_0100.		2 (4)
LOOP WITH CONTROL flights.		2 (4)
LOOP WITH CONTROL my_alv.		2 (4)
MODULE fill_table_control.		2 (4)
MODULE read_table_control.		2 (4)
MODULE status_0100.		2 (4)
MODULE user_command_0100.		2 (4)
ENDLOOP.		2 (4)
MODULE status_0100.		2 (4)
MODULE user_command_0100.		2 (4)

Ablauflogik Dynpro 0200 („Anzeigen/Ändern“)	erforder-lich	Block (Kreuzanzahl)
`PROCESS BEFORE OUTPUT.`	X	
`  MODULE fill_table_control.`		3 (1)
`  MODULE read_table_control.`		3 (1)
`  MODULE status_0200.`		3 (1)
`  MODULE user_command_0200.`		3 (1)
`PROCESS AFTER INPUT.`	X	
`  MODULE fill_table_control.`		4 (1)
`  MODULE read_table_control.`		4 (1)
`  MODULE status_0200.`		4 (1)
`  MODULE user_command_0200.`		4 (1)

TOP-Include	erforder-lich	Block (Kreuzanzahl)
`PROGRAM zz_dynpro.`		5 (5)
`CONTROLS flights TYPE TABLEVIEW USING SCREEN 100.`		5 (5)
`CONTROLS flights TYPE TABLEVIEW USING SCREEN 200.`		5 (5)
`TABLES spfli.`		5 (5)
`TABLES zz_spfli.`		5 (5)
`DATA: ok_code LIKE sy-ucomm.`		5 (5)
`DATA: wa_flug TYPE spfli.`		6 (2)
`DATA: wa_flug TYPE zz_spfli.`		6 (2)
`DATA: itab TYPE TABLE OF spfli.`		6 (2)
`DATA: itab TYPE TABLE OF zz_spfli.`		6 (2)
`DATA: my_container TYPE REF TO cl_gui_custom_container.`		6 (2)
`DATA: my_alv TYPE REF TO cl_gui_alv_grid.`		6 (2)

Include zu MODULE status_0100	erforder-lich	Block (Kreuzanzahl)
`MODULE status_0100 OUTPUT.`	X	
`  SET PF-STATUS '100'.`		7 (3)
`  SET PF-STATUS '200'.`		7 (3)
`  SET TITLEBAR '100'.`		7 (3)
`  SET TITLEBAR '200'.`		7 (3)
`  CLEAR ok_code.`		7 (3)
`  SELECT * FROM spfli INTO TABLE itab.`		8 (2)
`  SELECT * FROM spfli INTO CORRESPONDING FIELDS OF TABLE itab.`		8 (2)
`  SELECT * FROM zz_spfli INTO TABLE itab.`		8 (2)
`  SELECT * FROM zz_spfli INTO CORRESPONDING FIELDS OF TABLE itab.`		8 (2)
`  DESCRIBE TABLE itab LINES flights-lines.`		8 (2)
`  DESCRIBE TABLE itab LINES my_alv-lines.`		8 (2)
`ENDMODULE.`	X	

Include zu MODULE user_command_0100	erforder-lich	Block (Kreuzanzahl)
`MODULE user_command_0100 INPUT.`	X	
`  CASE ok_code.`	X	
`    WHEN 'BACK'.`	X	
`      LEAVE PROGRAM.`	X	
`    WHEN 'SELECT'`		9 (2)
`    WHEN 'SELECT' OR 'PICK'.`		9 (2)
`      SELECT SINGLE * FROM spfli INTO wa_flug` `                    WHERE carrid = spfli-carrid.`		9 (2)
`      SELECT SINGLE * FROM zz_spfli INTO wa_flug` `                    WHERE carrid = spfli-carrid.`		9 (2)

`      SELECT SINGLE * FROM spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid.`
		9 (2)

`      SELECT SINGLE * FROM zz_spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid.`
		9 (2)

`      SELECT SINGLE * FROM spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid AND`
`                          airpfrom = spfli-airpfrom AND`
`                          airpto = spfli-airpto.`
		9 (2)

`      SELECT SINGLE * FROM zz_spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid AND`
`                          airpfrom = spfli-airpfrom AND`
`                          airpto = spfli-airpto.`
		9 (2)

`      IF sy-dbcnt = 0.`		10 (2)
`      IF sy-dbcnt = 1.`		10 (2)
`        LEAVE TO SCREEN 100.`		10 (2)
`        LEAVE TO SCREEN 200.`		10 (2)
`      ENDIF.`	X	
`  ENDCASE.`	X	
`ENDMODULE.`	X	

Include zu MODULE status_0200	erforder-lich	Block (Kreuzanzahl)
`MODULE status_0200 OUTPUT.`	X	
`  SET PF-STATUS '100'.`		11 (3)
`  SET PF-STATUS '200'.`		11 (3)
`  SET TITLEBAR '100'.`		11 (3)
`  SET TITLEBAR '200'.`		11 (3)
`  CLEAR ok_code.`		11 (3)
`  wa_flug = spfli.`		12 (1)
`  spfli = wa_flug.`		12 (1)
`ENDMODULE.`	X	

Include zu MODULE user_command_0200	erforder-lich	Block (Kreuzanzahl)
`MODULE user_command_0200 INPUT.`	X	
`  CASE ok_code.`	X	
`    WHEN 'LEAVE'.`		13 (4)
`      LEAVE PROGRAM.`		13 (4)
`    WHEN 'BACK'.`		13 (4)
`      LEAVE TO SCREEN 100.`		13 (4)
`      LEAVE TO SCREEN 200.`		13 (4)
`    WHEN 'SAVE'.`	X	
`      MODIFY spfli FROM spfli.`		14 (1)
`      MODIFY zz_spfli FROM spfli.`		14 (1)
`      MODIFY spfli FROM wa_flug.`		14 (1)
`      MODIFY zz_spfli FROM wa_flug.`		14 (1)
`  ENDCASE.`	X	
`  spfli = wa_flug.`		15 (1)
`  wa_flug = spfli.`		15 (1)
`ENDMODULE.`	X	

Include zu MODULE fill_table_control	erforder-lich	Block (Kreuzanzahl)
`MODULE fill_table_control OUTPUT.`	X	
`  READ TABLE itab INTO zz_spfli INDEX flights-current_line.`		16 (1)
`  READ TABLE itab INTO zz_spfli INDEX sy-index.`		16 (1)
`  READ TABLE spfli INTO zz_spfli INDEX flights-current_line.`		16 (1)
`  READ TABLE spfli INTO zz_spfli INDEX sy-index.`		16 (1)
`  READ TABLE zz_spfli INTO zz_spfli INDEX flights-current_line.`		16 (1)
`  READ TABLE zz_spfli INTO zz_spfli INDEX sy-index.`		16 (1)
`ENDMODULE.`	X	

Include zu MODULE read_table_control	erforder-lich	Block (Kreuzanzahl)
`MODULE read_table_control INPUT.`	X	
`  DATA cursorline TYPE i.`	X	
`  GET CURSOR cursorline.`		17 (1)
`  GET CURSOR LINE cursorline.`		17 (1)
`  GET LINE cursorline.`		17 (1)
`  IF sy-subrc = 0.`	X	
`    cursorline = flights-top_line + cursorline - 1.`		18 (1)
`    cursorline = flights-top_line + cursorline.`		18 (1)
`    cursorline = flights-top_line + cursorline + 1.`		18 (1)
`  ELSE.`	X	
`    cursorline = -1.`		
`    cursorline = 0.`	X	
`    cursorline = 1.`		
`  ENDIF.`	X	
`  IF ( zz_spfli-selection = 'X' ).`		19 (1)
`  IF ( zz_spfli-selection = 'X' ) OR` `     ( flights-current_line = cursorline ).`		19 (1)
`  IF ( zz_spfli-selection = 'X' ) OR` `     ( my_alv-current_line = cursorline ).`		19 (1)
`  IF ( flights-current_line = cursorline ).`		19 (1)
`  IF ( my_alv-current_line = cursorline ).`		19 (1)
`    zz_spfli-carrid = spfli-carrid.`		20 (2)
`    zz_spfli-connid = spfli-connid.`		20 (2)
`    spfli-carrid = zz_spfli-carrid.`		20 (2)
`    spfli-connid = zz_spfli-connid.`		20 (2)
`  ENDIF.`	X	
`ENDMODULE.`	X	

Aufgabe 8 *(5% Bewertungsanteil)*

Charakterisiere mit jeweils wenigen eigenen Stichworten die folgenden Webdynpro-Begriffe:

1. Component

2. View

3. Window

4. Controller

5. Context

Das Team der FH und PTL Wedel wünscht viel Erfolg