Klausur ERP-Systeme und ERP 2 am 13. Februar 2014

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

Aufgabe 1+2 *(25% 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 des „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.

… 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 des „Selektionsbildschirms“ werden die beiden Brüche eingegeben.

Folgende (mathematische) Hinweise mögen 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 1+2 (Fort.)

Aufgabe 1+2 (Fort.)

Aufgabe 3 *(25% 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 3 (Fort.)

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 4 *(25% Bewertungsanteil)*

Gegeben sei im Data Dictionary die Datenbanktabelle ZZ_PERSONEN („Mitarbeiter“):

- PID: Mitarbeiteridentifikation (6-stellig alphanumerisch, Primärschlüssel)
- VORNAME: Vorname des Mitarbeiters (30-stellig alphanumerisch)
- NACHNAME: Nachname des Mitarbeiters (30-stellig alphanumerisch)
- TITEL: Titel des Mitarbeiters (15-stellig alphanumerisch, Fremdschlüssel)

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 eines Mitarbeiters 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 personen.		1 (4)
LOOP WITH CONTROL mein_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 personen.		2 (4)
LOOP WITH CONTROL mein_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 personen TYPE TABLEVIEW USING SCREEN 100.`		5 (5)
`CONTROLS personen TYPE TABLEVIEW USING SCREEN 200.`		5 (5)
`TABLES zz_personen.`		5 (5)
`TABLES zz_personen_t.`		5 (5)
`DATA: ok_code LIKE sy-ucomm.`		5 (5)
`DATA: wa_person TYPE zz_personen.`		6 (2)
`DATA: wa_person TYPE zz_personen_t.`		6 (2)
`DATA: itab TYPE TABLE OF zz_personen.`		6 (2)
`DATA: itab TYPE TABLE OF zz_personen_t.`		6 (2)
`DATA: mein_container TYPE REF TO cl_gui_custom_container.`		6 (2)
`DATA: mein_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 zz_personen INTO` `  TABLE itab.`		8 (2)
`  SELECT * FROM zz_personen INTO` `  CORRESPONDING FIELDS OF TABLE itab.`		8 (2)
`  SELECT * FROM zz_personen_t INTO` `  TABLE itab.`		8 (2)
`  SELECT * FROM zz_personen_t INTO` `  CORRESPONDING FIELDS OF TABLE itab.`		8 (2)
`  DESCRIBE TABLE itab LINES personen-lines.`		8 (2)
`  DESCRIBE TABLE itab LINES mein_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 zz_personen INTO wa_person` `                    WHERE pid = zz_personen-pid.`		9 (2)
`      SELECT SINGLE * FROM zz_personen_t INTO wa_person` `                    WHERE pid = zz_personen-pid.`		9 (2)

`      SELECT SINGLE * FROM zz_personen INTO wa_person`
`                    WHERE pid = zz_personen-pid AND`
`                          titel = zz_personen-titel.`
		9 (2)

`      SELECT SINGLE * FROM zz_personen_t INTO wa_person`
`                    WHERE pid = zz_personen-pid AND`
`                          titel = zz_personen-titel.`
		9 (2)

`      SELECT SINGLE * FROM zz_personen INTO wa_person`
`                    WHERE pid = zz_personen-pid AND`
`                          titel = zz_personen-titel AND`
`                          vorname = zz_personen-vorname AND`
`                          nachname = zz_personen-nachname.`
		9 (2)

`      SELECT SINGLE * FROM zz_personen_t INTO wa_person`
`                    WHERE pid = zz_personen-pid AND`
`                          titel = zz_personen-titel AND`
`                          vorname = zz_personen-vorname AND`
`                          nachname = zz_personen-nachname.`
		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_person = zz_personen.`		12 (1)
`  zz_personen = wa_person.`		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 zz_personen FROM zz_personen.`		14 (1)
`      MODIFY zz_personen_t FROM zz_personen.`		14 (1)
`      MODIFY zz_personen FROM wa_person.`		14 (1)
`      MODIFY zz_personen_t FROM wa_person.`		14 (1)
`  ENDCASE.`	X	
`  zz_personen = wa_person.`		15 (1)
`  wa_person = zz_personen.`		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_personen_t` `  INDEX personen-current_line.`		16 (1)
`  READ TABLE itab INTO zz_personen_t` `  INDEX sy-index.`		16 (1)
`  READ TABLE zz_personen INTO zz_personen_t` `  INDEX personen-current_line.`		16 (1)
`  READ TABLE zz_personen INTO zz_personen_t` `  INDEX sy-index.`		16 (1)
`  READ TABLE zz_personen_t INTO zz_personen_t` `  INDEX personen-current_line.`		16 (1)
`  READ TABLE zz_personen_t INTO zz_personen_t` `  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 = personen-top_line + cursorline - 1.`		18 (1)
`    cursorline = personen-top_line + cursorline.`		18 (1)
`    cursorline = personen-top_line + cursorline + 1.`		18 (1)
`  ELSE.`	X	
`    cursorline = -1.`		
`    cursorline = 0.`	X	
`    cursorline = 1.`		
`  ENDIF.`	X	
`  IF ( zz_personen_t-selection = 'X' ).`		19 (1)
`  IF ( zz_personen_t-selection = 'X' ) OR` `     ( personen-current_line = cursorline ).`		19 (1)
`  IF ( zz_personen_t-selection = 'X' ) OR` `     ( mein_alv-current_line = cursorline ).`		19 (1)
`  IF ( personen-current_line = cursorline ).`		19 (1)
`  IF ( mein_alv-current_line = cursorline ).`		19 (1)
`    zz_personen_t-pid = zz_personen-pid.`		20 (1)
`    zz_personen_t-title = zz_personen-title.`		20 (1)
`    zz_personen-pid = zz_personen_t-pid.`		20 (1)
`    zz_personen-title = zz_personen_t-title.`		20 (1)
`  ENDIF.`	X	
`ENDMODULE.`	X	

Aufgabe 5 *(25% Bewertungsanteil)*

Erläutere mit eigenen Worten stichwortartig möglichst exakt entweder die drei Programmfragmente auf dieser Seite oder den Report auf der nächsten Seite.

Top-Include

```
PROGRAM  zz_dynpro.

TABLES spfli.
DATA ok_code LIKE sy-ucomm.
DATA wa_flug TYPE spfli.
```

Process After Input (PAI) Dynpro 100

```
MODULE user_command_0100 INPUT.

  CASE ok_code.
    WHEN 'BACK'.
      LEAVE PROGRAM.
    WHEN 'SELECT'.
      SELECT SINGLE * FROM spfli INTO wa_flug WHERE carrid = spfli-carrid AND
                                                    connid = spfli-connid.
      IF sy-dbcnt = 1.
        CALL FUNCTION 'ENQUEUE_EZ_SPFLI'
          EXPORTING
            mode_spfli = 'E'
            carrid     = spfli-carrid
            connid     = spfli-connid
          EXCEPTIONS
            FOREIGN_LOCK = 1.
        IF sy-subrc = 0.
          LEAVE TO SCREEN 200.
        ENDIF.
      ENDIF.
  ENDCASE.
  CLEAR ok_code.

ENDMODULE.
```

Process After Input (PAI) Dynpro 200

```
MODULE user_command_0200 INPUT.

  CASE ok_code.
    WHEN 'LEAVE'.
      LEAVE PROGRAM.
    WHEN 'BACK'.
      CALL FUNCTION 'DEQUEUE_EZ_SPFLI'
        EXPORTING
          mode_spfli = 'E'
          carrid     = spfli-carrid
          connid     = spfli-connid.
      LEAVE TO SCREEN 100.
    WHEN 'SAVE'.
      MODIFY spfli FROM spfli.
  ENDCASE.
  CLEAR ok_code.
  wa_flug = spfli.

ENDMODULE.
```

Aufgabe 5 (Fort.)

Report ZZ_OBJECTS

```
REPORT  ZZ_OBJECTS.

CLASS lcl_airplane DEFINITION.

  PUBLIC SECTION.
    METHODS: attribute_setzen IMPORTING
               im_name TYPE string
               im_planetype TYPE string,
             attribute_anzeigen.

  PRIVATE SECTION.
    DATA: name TYPE string,
          planetype TYPE string.

ENDCLASS.

CLASS lcl_airplane IMPLEMENTATION.

  METHOD attribute_setzen.
    name = im_name.
    planetype = im_planetype.
  ENDMETHOD.

  METHOD attribute_anzeigen.
    WRITE: / 'Name des Flugzeugs: ', name,
           / 'Typ des Flugzeugs: ', planetype.
  ENDMETHOD.

ENDCLASS.

DATA: r_plane TYPE REF TO lcl_airplane,
      it_plane_list TYPE TABLE OF REF TO lcl_airplane.

START-OF-SELECTION.

CREATE OBJECT r_plane.
r_plane->attribute_setzen( im_name = 'München'
                           im_planetype = 'Boeing 737' ).
APPEND r_plane TO it_plane_list.

CREATE OBJECT r_plane.
r_plane->attribute_setzen( im_name = 'Hamburg'
                           im_planetype = 'Airbus 380' ).
APPEND r_plane TO it_plane_list.

LOOP AT it_plane_list INTO r_plane.
  r_plane->attribute_anzeigen( ).
ENDLOOP.
```

Aufgabe 5 (Fort.)

Aufgabe 5 (Fort.)

Das Team der FH und PTL Wedel wünscht viel Erfolg