Klausur ERP-Systeme und ERP 2 am 11. Februar 2011

(B_Inf2.0 v361, B_Winf1.0 v362, IAW4.0/5.0 552 – 120 Minuten)

Aufgabe 1

Entwicklungswerkzeuge im SAP-System sind u.a.: ABAP Dictionary, ABAP Editor, Class Builder, Debugger, Function Builder, Menu Painter, Object Navigator und Screen Painter.

Ordne den nachfolgenden Aussagen bzw. Funktionen das jeweils am ehesten passende der obigen acht Werkzeuge zu:

1. Benutzerfreundliche Oberfläche, die alle Entwicklungswerkzeuge intuitiv integriert.

2. Bietet Funktionen zum Editieren von ABAP-Code.

3. Ermöglicht schrittweises Ausführen einer Anwendung.

4. Erstellung und Pflege von Dynpros.

5. Erstellung und Pflege von Funktionsbausteinen.

6. Erstellung und Pflege von globalen Klassen.

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

8. Explizites Erstellen und Zurückholen von Versionen des gleichen Programms.

9. Generierung von dynamischen Ein- und Ausgabemasken zum Testen.

10. Globales Verzeichnis der „Metadaten“.

11. Pretty Printer unterstützt das Einrücken bei Verschachtelungen.

12. Überprüfung von Variablen, Strukturen, und Tabellen zur Laufzeit.

Aufgabe 2

Benenne jeweils eine möglichst charakteristische Verwendung für folgende Datentypen:

1. C

2. D

3. F

4. I

5. P

6. T

Aufgabe 3

Forme folgendes Programmfragment in eine syntaktisch korrekte Fassung um:

```
DATA A TYPE I.
DATA B TYPE I.

IF ( A - 1 ) = ( B + 1 ).
  WRITE 'PASSEND'.
ENDIF.
```

Welche Werte beinhalten die Variablen *MyString2*, *MyString3*, *MyString4* und *MyString5* nach Ausführung des folgenden Programmfragments ?

```
DATA myString1(20) TYPE C VALUE 'Dies ist ein String'.
DATA myString2(20) TYPE C.
DATA myString3(20) TYPE C.
DATA myString4(20) TYPE C.
DATA myString5(20) TYPE C VALUE 'Bergedorf'.

DATA i TYPE I VALUE 3.
DATA j TYPE I VALUE 5.

MyString2 = myString1+9(3).

MyString3 = myString1+0(4).

MyString4 = myString1+j(i).

myString5+5(5) = 'stadt'.
```

Aufgabe 4

Welche der folgenden Aussagen zu Funktionsbausteinen sind korrekt ? (Bitte Markieren)

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

Welche der folgenden Objekte sind Datentypen im Sinne des ABAP Dictionary ? (Bitte Markieren)

1. Datenbanktabelle
2. Datenelement
3. Domäne
4. Sperrobjekt
5. Struktur
6. Suchhilfe
7. Tabellentyp
8. Typgruppe
9. View

Es gilt folgende „Spielregel“: Richtige Markierung: Pluspunkt, Falsche Markierung: Minuspunkt

Aufgabe 5

Ordne nachfolgenden Aussagen bzw. Funktionen den jeweils am ehesten passenden der folgenden zwölf SAP-Begriffe zu: ALV-Grid, BAPI, Berechtigungsprofil, Nummernkreisobjekt, Paket, Rolle, Sperrobjekt, Suchhilfe, Tabstrip-Control, 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. 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

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“ ausdrücklich unberücksichtigt bleiben 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“:

- *richtiges Kreuz: Pluspunkt*
- *falsches Kreuz: Minuspunkt*
- *kein Kreuz: Weder Pluspunkt noch Minuspunkt*

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

Ablauflogik Dynpro 0200 („Anzeigen/Ändern“)	erforderlich	nicht erforderlich
`PROCESS BEFORE OUTPUT.`		
`  MODULE status_0200.`		
`  MODULE fill_table_control.`		
`PROCESS AFTER INPUT.`		
`  MODULE read_table_control.`		
`  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: itab TYPE TABLE OF zz_spfli.`		
`DATA: fill TYPE i.`		
`DATA: my_container TYPE REF TO cl_gui_custom_container.`		
`DATA: my_alv TYPE REF TO cl_gui_alv_grid.`		

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 CORRESPONDING FIELDS OF TABLE itab.`		
`  DESCRIBE TABLE itab LINES fill.`		
`  DESCRIBE TABLE itab LINES flights-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' OR 'PICK'.`		
`    WHEN 'SELECT'.`		
`      SELECT SINGLE * FROM 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 spfli INTO wa_flug`
`                    WHERE carrid = spfli-carrid AND`
`                          connid = spfli-connid AND`
`                          airpfrom = spfli-airpfrom AND`
`                          airpto = spfli-airpto.`		

`      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 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.`		
`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 LINE cursorline.`		
`  IF ( zz_spfli-selection = 'X' ).`		
`  IF ( zz_spfli-selection = 'X' ) OR` `     ( flights-current_line = cursorline ).`		
`    spfli-carrid = zz_spfli-carrid.`		
`    spfli-connid = zz_spfli-connid.`		
`  ENDIF.`		
`ENDMODULE.`		

Aufgabe 7

Entwickle eine Business Server Pages (BSP) – Applikation für einen Taschenrechner mit den drei Grundrechenarten Addition, Subtraktion und Multiplikation. Die BSP-Applikation soll aus vier Seiten bestehen. Auf der ersten Seite werden sowohl zwei Zahlen eingelesen, als auch wird über drei Buttons entweder zur zweiten, dritten oder vierten Seite verzweigt. Die zweite Seite beinhaltet die Berechnung und Ausgabe der Addition beider Zahlen, die dritte und vierte Seite jeweils analog die Subtraktion und Multiplikation. Von der zweiten, dritten und vierten Seite soll jeweils über Buttons die Möglichkeit bestehen, direkt zur Seite der beiden jeweils übrigen Grundrechenarten zu verzweigen.

Spezifiziere zur Lösung dieser Aufgabe insgesamt acht Code-Bestandteile: Jeweils das Layout und den Event-Handler der obigen vier Seiten.

Aufgabe 7 (Fort)

Aufgabe 7 (Fort)

Das Team der FH und PTL Wedel wünscht viel Erfolg