

Klausur am 12. Februar 2015

ERP-Systeme, ERP 2 und Anwendungsentwicklung in ERP-Systemen

(B_Inf2.0 v360, IAW6.1 552, M_Inf14.0 C098, KAI1.0 213, IAW6.5 552 – 120 Minuten)

Aufgabe 1

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.

Aufgabe 2

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

1. D

2. I

3. P

4. T

Aufgabe 3

1. Kennzeichne die <i>richtige</i> Behauptung (1 Kreuz von 4 Möglichkeiten)	
ABAP ist datenbankabhängig und plattformabhängig	<input type="checkbox"/>
ABAP ist datenbankabhängig und plattformunabhängig	<input type="checkbox"/>
ABAP ist datenbankunabhängig und plattformabhängig	<input type="checkbox"/>
ABAP ist datenbankunabhängig und plattformunabhängig	<input type="checkbox"/>

2. Kennzeichne die <i>richtige</i> Behauptung (1 Kreuz von 4 Möglichkeiten)	
ABAP richtet sich weder an die Wiederverwendbarkeit von Code- noch von Datenobjekten aus	<input type="checkbox"/>
ABAP richtet sich stark an die Wiederverwendbarkeit von Codeobjekten aus	<input type="checkbox"/>
ABAP richtet sich stark an die Wiederverwendbarkeit von Datenobjekten aus	<input type="checkbox"/>
ABAP richtet sich stark an die Wiederverwendbarkeit von Code- und Datenobjekten aus	<input type="checkbox"/>

3. Kennzeichne die <i>falschen</i> Behauptungen (2 Kreuze von 4 Möglichkeiten)	
ABAP-Programme liegen nur in kompilierter Form vor	<input type="checkbox"/>
ABAP-Programme liegen auch als Quelltext vor	<input type="checkbox"/>
ABAP-Programme werden beim ersten Aufruf kompiliert	<input type="checkbox"/>
ABAP-Programme werden bei jedem Aufruf kompiliert	<input type="checkbox"/>

4. Kennzeichne die <i>falschen</i> Behauptungen (2 Kreuze von 4 Möglichkeiten)	
Das Repository ist mandantenunabhängig	<input type="checkbox"/>
Das Repository ist mandantenabhängig	<input type="checkbox"/>
Die Anwendungsdaten sind mandantenunabhängig	<input type="checkbox"/>
Die Anwendungsdaten sind mandantenabhängig	<input type="checkbox"/>

5. Transporte in einer typischen SAP-Systemlandschaft erfolgen: (2 Kreuze von 6 Möglichkeiten)	
Von Development System nach Production System	<input type="checkbox"/>
Von Development System nach Quality Assurance System	<input type="checkbox"/>
Von Production System nach Development System	<input type="checkbox"/>
Von Production System nach Quality Assurance System	<input type="checkbox"/>
Von Quality Assurance System nach Development System	<input type="checkbox"/>
Von Quality Assurance System nach Production System	<input type="checkbox"/>

6. Kennzeichne die <i>falschen</i> Behauptungen (2 Kreuze von 3 Möglichkeiten)	
In ABAP ist die Konvertierung von String nach Integer nicht möglich	<input type="checkbox"/>
In ABAP ist die Konvertierung von String nach Integer in Abhängigkeit vom Inhalt möglich	<input type="checkbox"/>
In ABAP ist die Konvertierung von String nach Integer generell möglich	<input type="checkbox"/>

7. Markiere die Datentypen im Sinne des ABAP Dictionary (3 Kreuze von 9 Möglichkeiten) (<i>doppelt gewertet</i>)	
Datenbanktabelle	<input type="checkbox"/>
Datenelement	<input type="checkbox"/>
Domäne	<input type="checkbox"/>
Sperrobjekt	<input type="checkbox"/>
Struktur	<input type="checkbox"/>
Suchhilfe	<input type="checkbox"/>
Tabellentyp	<input type="checkbox"/>
Typgruppe	<input type="checkbox"/>
View	<input type="checkbox"/>

8. Kennzeichne die <i>falschen</i> Behauptungen (2 Kreuze von 4 Möglichkeiten)	
Der Inhalt von Datenbanktabellen steht nicht über die Programmlaufzeit hinaus zur Verfügung	<input type="checkbox"/>
Der Inhalt von Datenbanktabellen steht über die Programmlaufzeit hinaus zur Verfügung	<input type="checkbox"/>
Der Inhalt von internen Tabellen steht nicht über die Programmlaufzeit hinaus zur Verfügung	<input type="checkbox"/>
Der Inhalt von internen Tabellen steht über die Programmlaufzeit hinaus zur Verfügung	<input type="checkbox"/>

9. Kennzeichne die <i>falschen</i> 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 <i>richtigen</i> Behauptungen (3 Kreuze von 6 Möglichkeiten) (<i>doppelt gewertet</i>)	
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 <i>falschen</i> Aussagen zu Funktionsbausteinen: (3 Kreuze von 6 Möglichkeiten) (<i>doppelt gewertet</i>)	
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. Abfangen von Laufzeitfehlern mithilfe der objektorientierten Programmierung möglich durch: (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

Erläutere mit eigenen Worten stichwortartig möglichst exakt nachfolgende Programmfragmente:

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 4 (Fort.)

Aufgabe 5

Erläutere mit eigenen Worten stichwortartig möglichst exakt entweder die drei BSP-Bestandteile oder den Report ZZ_OBJECTS.

Layout eingabe.htm

```
<%@page language="abap"%>
<HTML>
  <HEAD>
    <TITLE>Selektionskriterien SPFLI</TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Formular">
      <TABLE>
        <TR>
          <TD>Fluggesellschaft:</TD>
          <TD><INPUT TYPE="text" NAME="param_carrid"></TD>
        </TR><TR>
          <TD>Verbindung: </TD>
          <TD><INPUT TYPE="text" NAME="param_connid"></TD>
        </TR><TR>
          <TD>Testmodus:</TD>
          <TD><INPUT TYPE="checkbox" NAME="param_modus" value="X"></TD>
        </TR><TR>
          <TD COLSPAN="2">&nbsp;</TD>
        </TR><TR>
          <TD COLSPAN="2"><INPUT TYPE="submit" NAME="onInputProcessing(event_1)"
            VALUE="Flugdaten anzeigen"></TD>
        </TR>
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

Eventhandler OnInputProcessing eingabe.htm

```
CASE event_id.
  WHEN 'event_1'.
    navigation->set_parameter( 'param_carrid' ).
    navigation->set_parameter( 'param_connid' ).
    navigation->set_parameter( 'param_modus' ).
    navigation->goto_page( 'ausgabe.htm' ).
ENDCASE.
```

Aufgabe 5 (Fort.)

Layout ausgabe.htm

```
<%@page language="abap" %>
<HTML>
  <HEAD>
    <TITLE>Selektionsergebnis SPFLI</TITLE>
  <%
    DATA: it_fluege TYPE TABLE OF spfli,
           wa_flug TYPE spfli.
    IF param_carrid = '' AND param_connid = ''.
      SELECT * FROM spfli INTO TABLE it_fluege.
    ELSEIF param_carrid = ''.
      SELECT * FROM spfli INTO TABLE it_fluege WHERE connid = param_connid.
    ELSEIF param_connid = ''.
      SELECT * FROM SPFLI INTO TABLE it_fluege WHERE carrid = param_carrid.
    ELSE.
      SELECT * FROM spfli INTO TABLE it_fluege WHERE carrid = param_carrid
              AND connid = param_connid.
    ENDIF.
  %>
</HEAD>
<BODY>
  <H2>Ausgabe von Daten aus SPFLI</H2>
  <TABLE BORDER="1">
    <TR>
      <TH>Fluggesellschaft</TH>
      <TH>Verbindung</TH>
      <TH>Startflughafen</TH>
      <TH>Zielflughafen</TH>
      <TH>Abflugszeit</TH>
      <TH>Ankunftszeit</TH>
    </TR>
    <%
      LOOP AT it_fluege INTO wa_flug.
    %>
    <TR>
      <TD><%= wa_flug-carrid %></TD>
      <TD><%= wa_flug-connid %></TD>
      <TD><%= wa_flug-airpfrom %></TD>
      <TD><%= wa_flug-airpto %></TD>
      <TD><%= wa_flug-deptime %></TD>
      <TD><%= wa_flug-aritime %></TD>
    </TR>
    <%
      ENDLOOP.
    %>
  </TABLE>
  <%
    IF param_modus = 'X'.
  %>
  <H3>Testmodus</H3>
  <%
    ELSE.
  %>
  <H3>Produktivmodus</H3>
  <%
    ENDIF.
  %>
</BODY>
</HTML>
```

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