

M. Caramihai, © 2020

**PROGRAMAREA
ORIENTATA
OBIECT**

CURS 10

Sistemele AS400 si SAP – o abordare OO

ERP (1)

- In anii '70 apare programul de planificare a necesarului de materiale (MRP, acronimul pentru *Material Requirements Planning*). MRP este primul program ce utilizeaza aplicatii software pentru programarea proceselor de productie ale companiei.
- In anii '80, conceptul de MRP evolueaza, transformandu'se in planificarea necesarului de productie (MRP II, acronim pentru *Manufacturing Requirements Planning*). Acesta extinde functionalitatile MRP, utilizand aplicatii software pentru coordonarea proceselor de productie,
- In anii '90 apare conceptul de planificare a resurselor intreprinderii (ERP – acronim pentru *enterprise resource planning*). Acest nou sistem extinde functionalitatile MRP II asupra mai multor sectoare ale intreprinderii, utilizand o aplicatie software extinsa, cu mai multe module, pentru imbunatatirea performantelor proceselor interne de business.

ERP (2)

- O aplicație **ERP** poate fi privită ca o cutie în care se introduc date (input-uri), care sunt prelucrate, stocate și procesate în așa fel încât să poată oferi la ieșire datele care sunt solicitate de utilizatori (Output-uri).

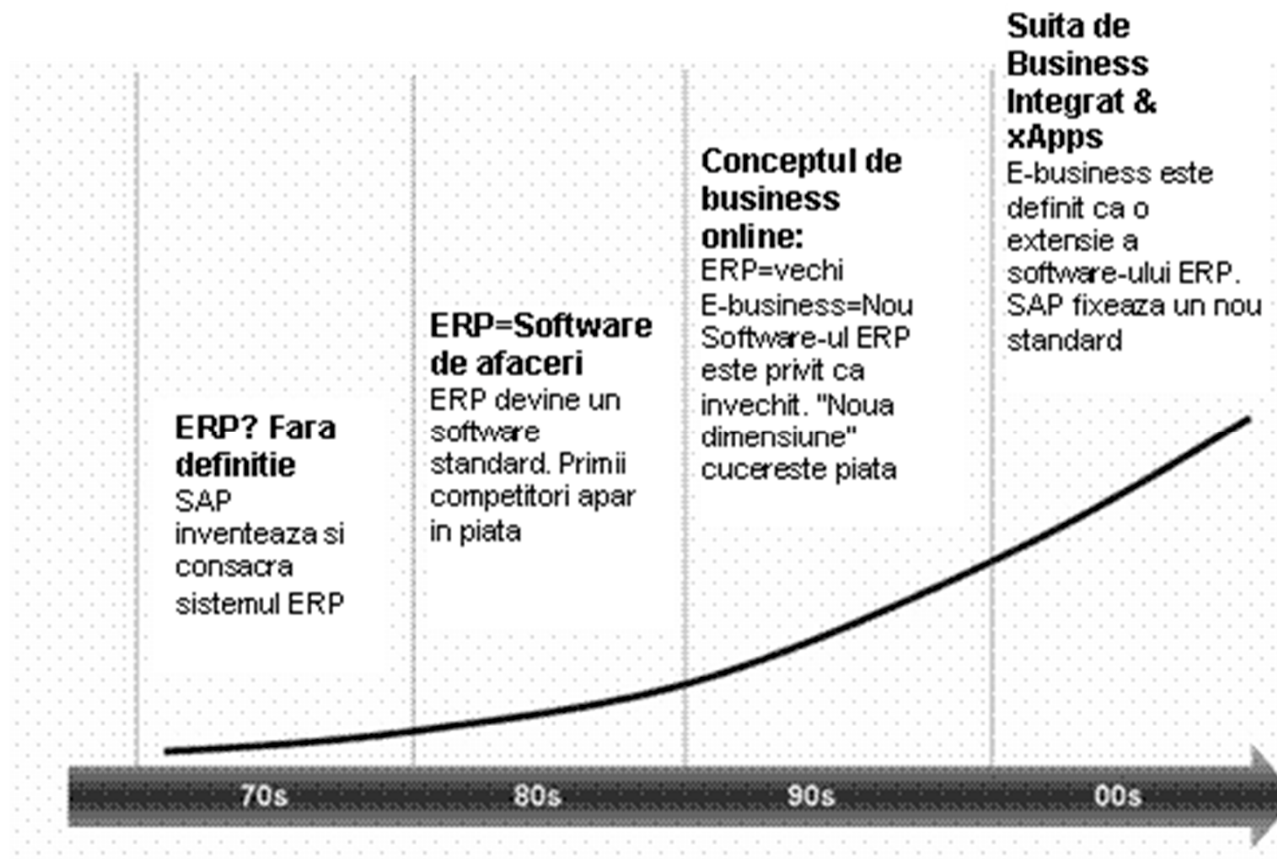


Sistemul SAP

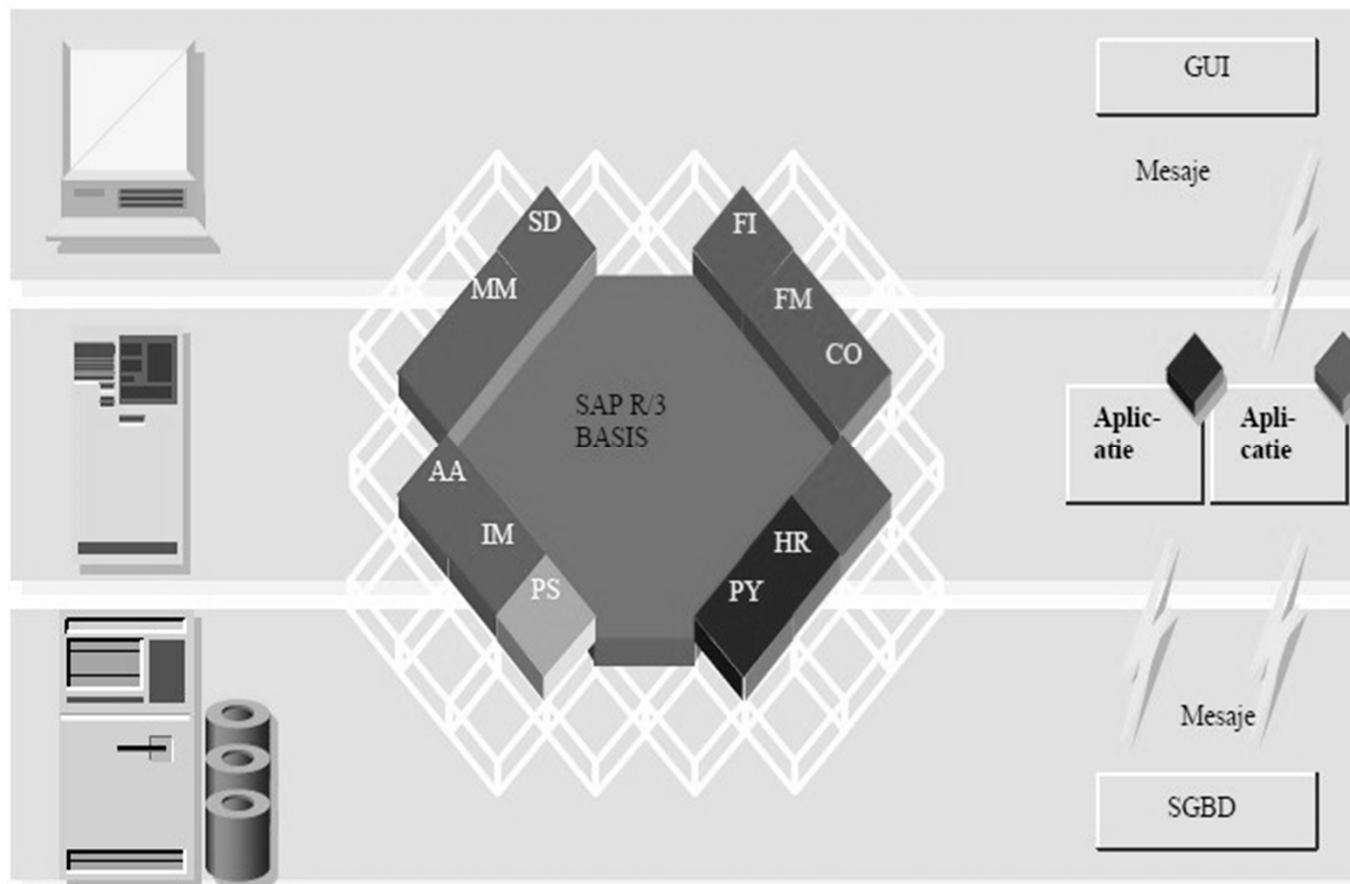
SAP (1)

- **SAP AG** a fost fondată în 1972 ca „Systemanalyse und Programmentwicklung” de către cinci fost ingineri de la IBM în Mannheim, Baden-Württemberg.
- **SAP** este cel mai important program de tip ERP [Enterprise Resource Planning] (sistem integrat de gestiunea resurselor întreprinderii) din lume.
- Programul s’a numit inițial **SAP R3**, unde abrevierea **SAP** provine de la „Sisteme, Aplicații și Produse în prelucrarea datelor”, **R** vine de la timp real, iar numărul **3** se referă la trei niveluri de arhitectura client-server: GUI, Server de aplicații și Baza de date.
- La momentul de față SAP AG este cea mai mare companie de software din Europa și a treia ca mărime din lume

SAP (2)



SAP (3)



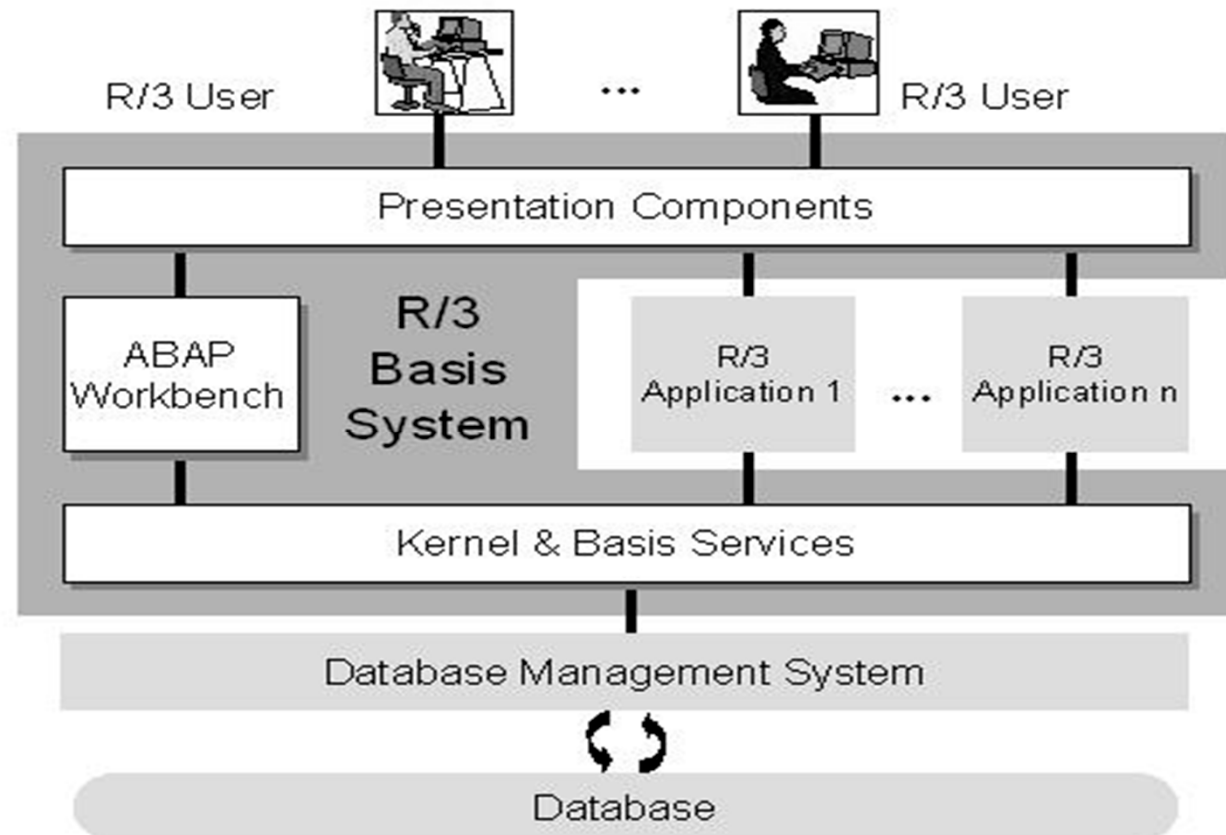
SAP (4)

Module

- managementul materialelor (MM);
- vanzari si distributie (SD);
- planificare productie (PP);
- financiar/contabilitate (FI/CO);
- contabilitate mijloace fixe (AA);
- service clienti (CS);
- gestiunea depozitelor (WM);
- managementul calitatii (QM); I
- ntretinere echipamente (PM);
- resurse umane (HR).

Arhitectura SAP (1)

Arhitectura C/S: 3 nivele – pot fi pe masini diferite → scalabil



Arhitectura SAP (2)

Nivelul bazei de date - Cel mai de jos nivel. Aici sunt gestionate datele cu ajutorul unui sistem de baze de date relational (RDBMS). Tot aici sunt gestionate datele de baza (nomenclatoare), datele tranzactionale, programele si metadatele. Toate poarta numele generic: **obiecte SAP**.

Nivelul aplicatiilor – Aici se executa, prin intermediul unui dispecer, cererile utilizatorilor (tranzactii). Aceste tranzactii (de fapt, programe ABAP) “solicita” date de la nivelul inferior si, dupa prelucrare, le “expediaza” tot la acel nivel pe cele noi sau modificate

Nivelul de prezentare (SAPGUI) - Contine interfetele catre utilizatori, in care un end-user poate accesa o aplicatie prin intermediul tranzactiilor.

Distributia tehnica este independenta de locatia fizica a hardware-ului.

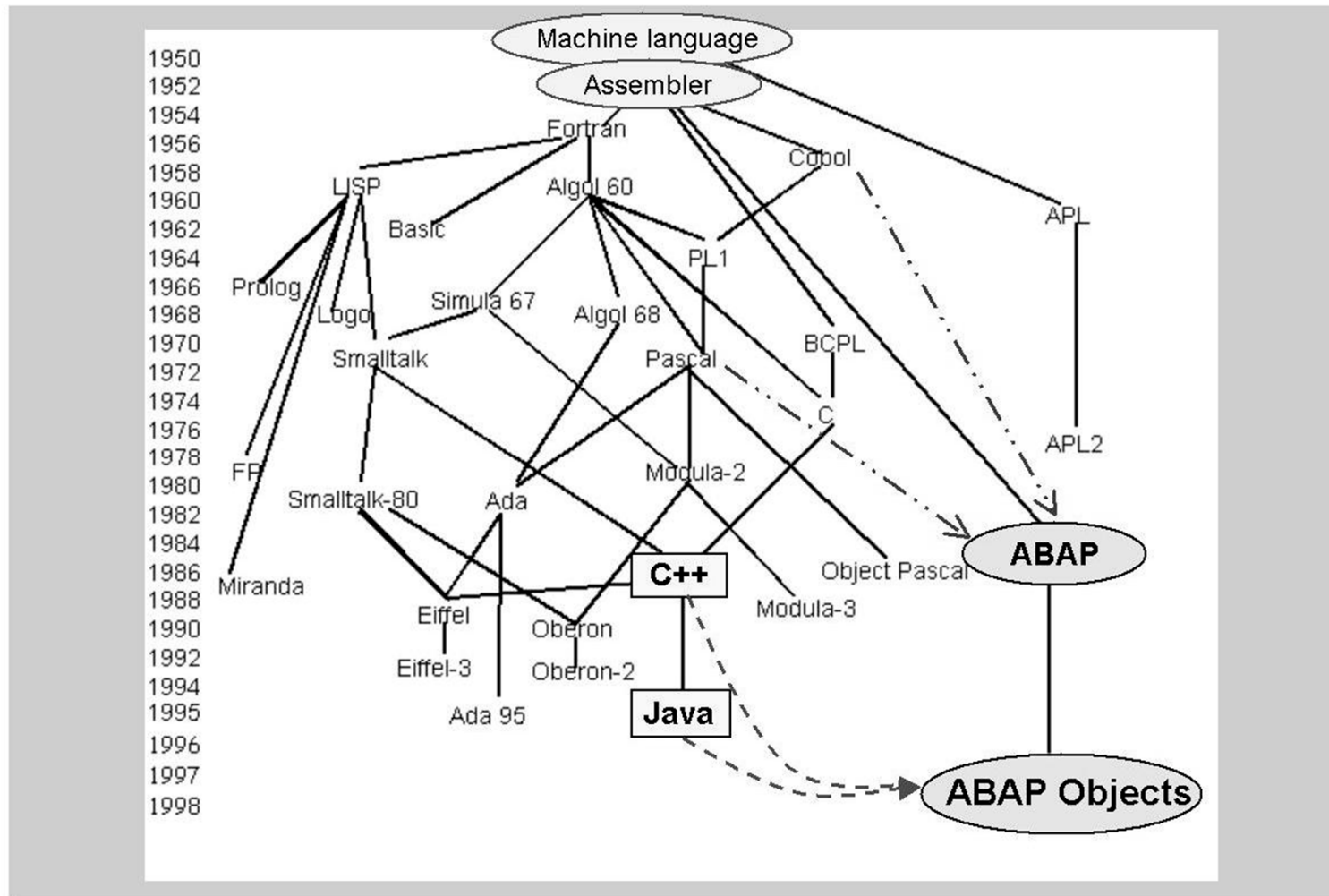
Vertical, toate nivelele pot fi instalate pe acelasi hard sau pe hard-uri diferite.

Orizantal, nivelele de aplicatie si prezentare pot fi divizate intre mai multe statii de lucru. Distributia orizontala a componentelor bazei de date, depinde de tipul de baza de date instalat.

Limbajul ABAP (1)

- **ABAP** – Advanced Business Application Programming
- Dezvoltat pe baza limbajului 4GL
- Specializat in “business applications”
- ABAP Objects: - Extensia OO pentru ABAP

Limbajul ABAP (2)



Clase ABAP (1)

```
CLASS class_name DEFINITION.
```

```
....
```

```
ENDCLASS.
```

```
CLASS class_name IMPLEMENTATION.
```

```
....
```

```
ENDCLASS.
```

Clase ABAP (2)

◆ Vizibilitate

➤ **Public**

- toate componentele sunt publice.

➤ **Protected**

- componentele sunt protejate

➤ **Private**

- componentele sunt private

Clase ABAP (3)

- Definire vizibilitate

```
CLASS class_name DEFINITION.  
    PUBLIC SECTION.  
  
        ....  
    PROTECTED SECTION.  
  
        ....  
    PRIVATE SECTION.  
  
        ....  
ENDCLASS.
```


Clase ABAP (4)

● Componentele clasei

1. **Atribute** – obiectele din cadrul clasei
2. **Metode** – blocul de procesare

Observatie: **NU** exista metoda de tip *Destructor*

3. **Evenimente** – permit obiectelor aferente unei clase sa'si publice statusul. Alte obiecte pot raspunde ref aceasta schimbare de status.

Mosteniri

● Doar mosteniri simple

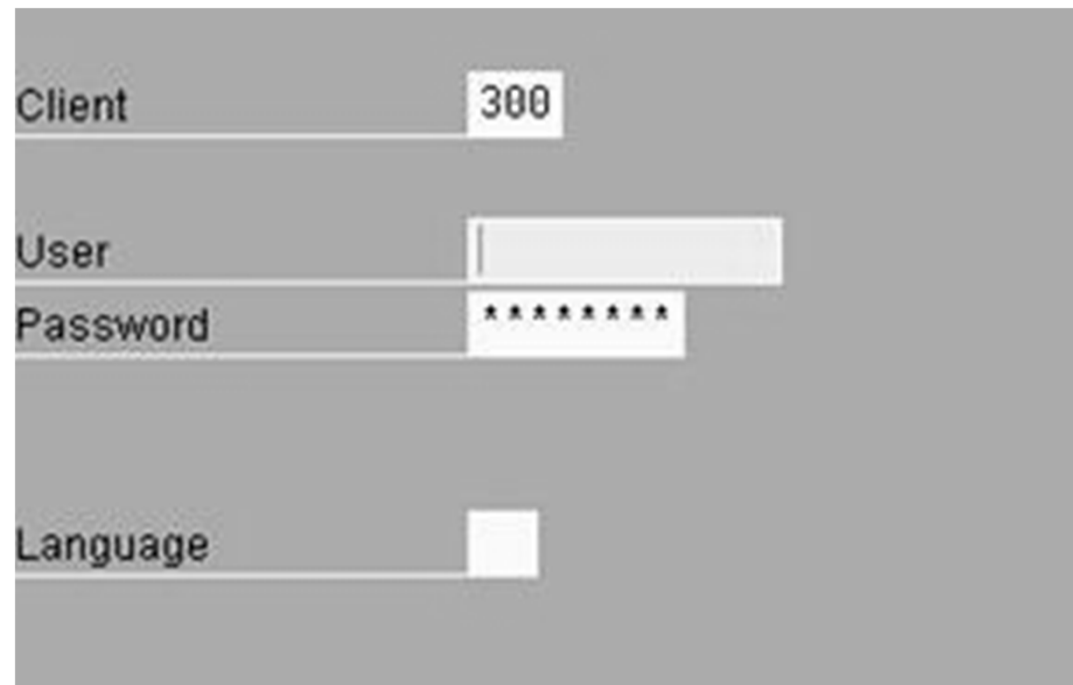
```
CLASS c1 DEFINITION.  
    ...  
CLASS c2 DEFINITION INHERITING FROM c1.  
    ...  
CLASS c3 DEFINITION INHERITING FROM c1.  
    ...
```

Interfete

- Pot fi definite mai multe interfete pentru o aceeași clasă

```
INTERFACE i1.  
    ...  
INTERFACE i2.  
    ...  
CLASS c1 DEFINITION.  
    PUBLIC SECTION.  
        INTERFACES: i1, i2.  
    ...
```

Logare in SAP

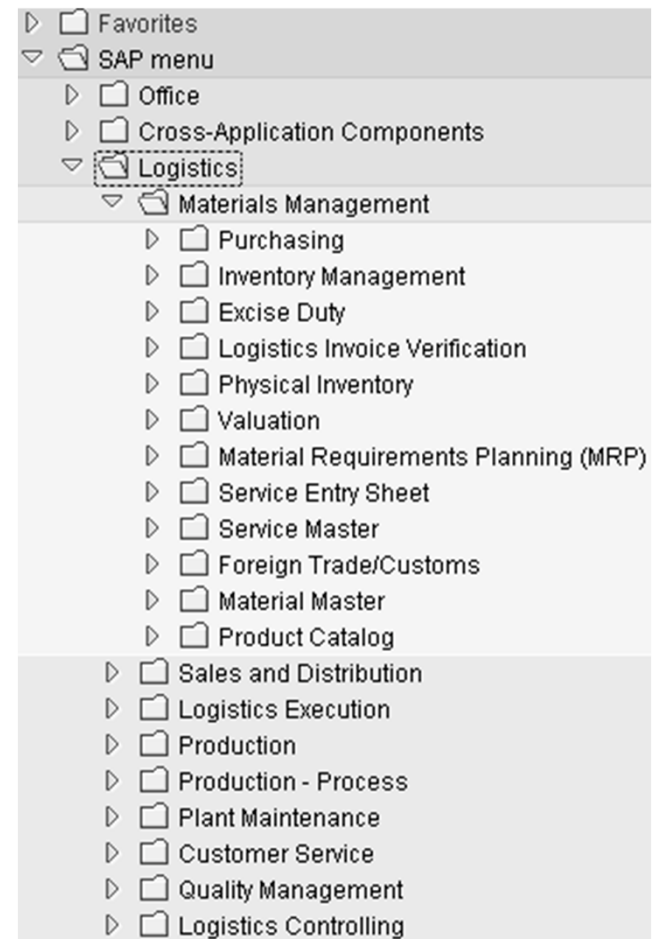


A screenshot of the SAP login form. It features four input fields on a gray background. The 'Client' field contains the value '300'. The 'User' field is empty. The 'Password' field contains ten asterisks. The 'Language' field is empty.

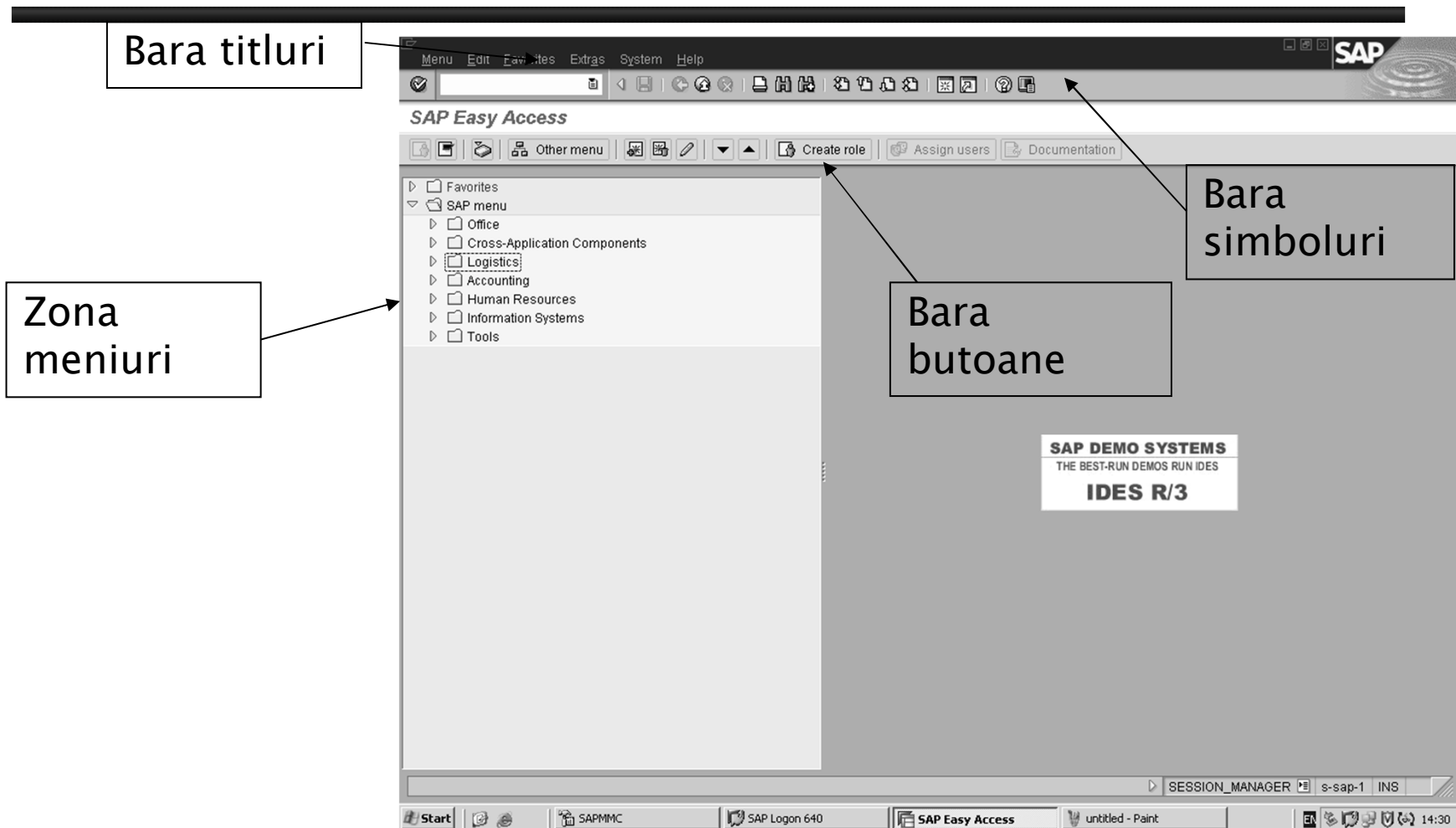
Client	300
User	
Password	*****
Language	

Navigare in SAP

- SAP foloseste sesiuni - nr max stabilit de administratorul de sistem
- Utilizatorii pot sa isi personalizeze meniul
- Meniul poate fi afisat in mai multe limbi



Ecranul initial SAP



Sistemul AS400

Prezentare AS/400 (1)

- Proiectul **SilverLake**

- 1985 data de inceput
- Un nou tip de masina
- Sucesor al System/38

- *Application System/400*
(AS/400)

- Anuntat in Iunie 1988
- Imbunatatiri intr'o rata 10% / an
- Sistem independent de hardware incepand cu 1995
- CISC (Complex Instruction Set Computer)
 - Lungime variabila a instructiunii

The logo for the IBM AS/400 system, featuring the text "IBM®" above "AS/400®" in a white, sans-serif font, set against a dark gray rectangular background.

IBM®
AS/400®

Prezentare AS/400 (2)

- *Upgrading*
 - RISC (*Reduced Instruction Set Computer*)
 - Instrucțiuni de aceeași lungime
 - Trecere de la 48-byte la 64-byte
 - Independența de hardware
- *Enhanced Series* sau ***e-series***
 - Dezvoltare începând cu 1994
 - Lansare: August 1997
- Rebranding: **System i**, 2006
- Rebranding: **IBM Power System**, 2008

Prezentare AS/400 (3)

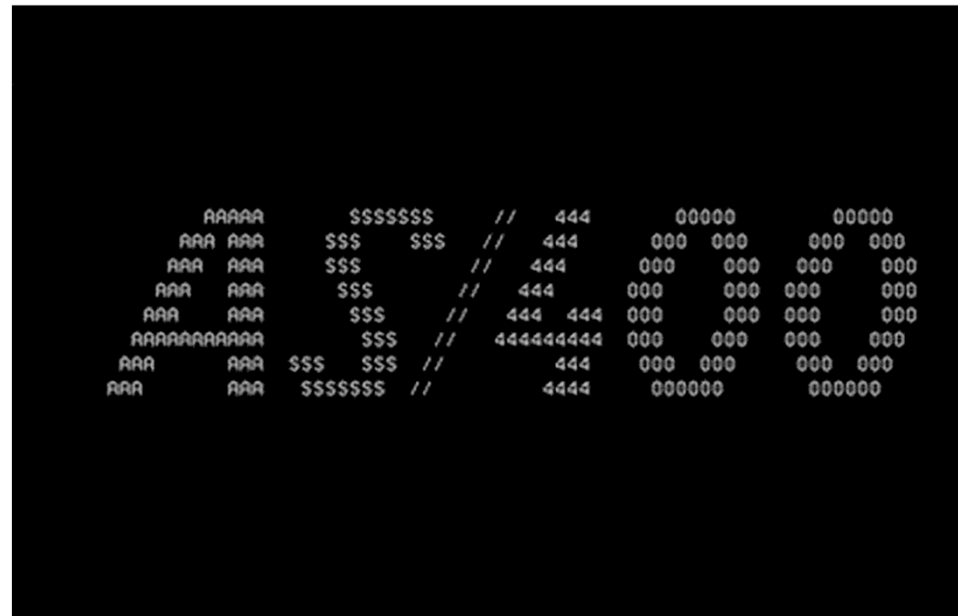


Obiecte si stocari de date

- Sistemul are la baza "obiectul":
 - ➔ Orice este stocat pe un AS/400 este un obiect
- Obiectele sunt stocate in biblioteci de obiecte (*Libraries*)
- Datele sunt stocate in format:
 - ➔ **EBCDIC** - Extended Binary Coded Decimal Interchange Code
 - ➔ **ASCII** - American Standard Code for Information Interchange
- Nu se face nici o deosebire intre stocarea pe disc si cea din memorie
- Obiectele trebuie sa fie in memorie pentru a putea fi utilizate

Ce este un obiect AS400?

Un obiect este un element care **exista**,
ocupa un spatiu de memorie si asupra
caruia pot actiona **comenzi** (ale sistemului).



Comanda WRKOBJ

Work with Objects (WRKOBJ)

Type choices, press Enter.

Object	_____	Name, generic*, *ALL
Library	<u>*LIBL</u>	Name, *LIBL, *CURLIB...
Object type	<u>*ALL</u>	*ALL, *ALRTBL, *AUTL...

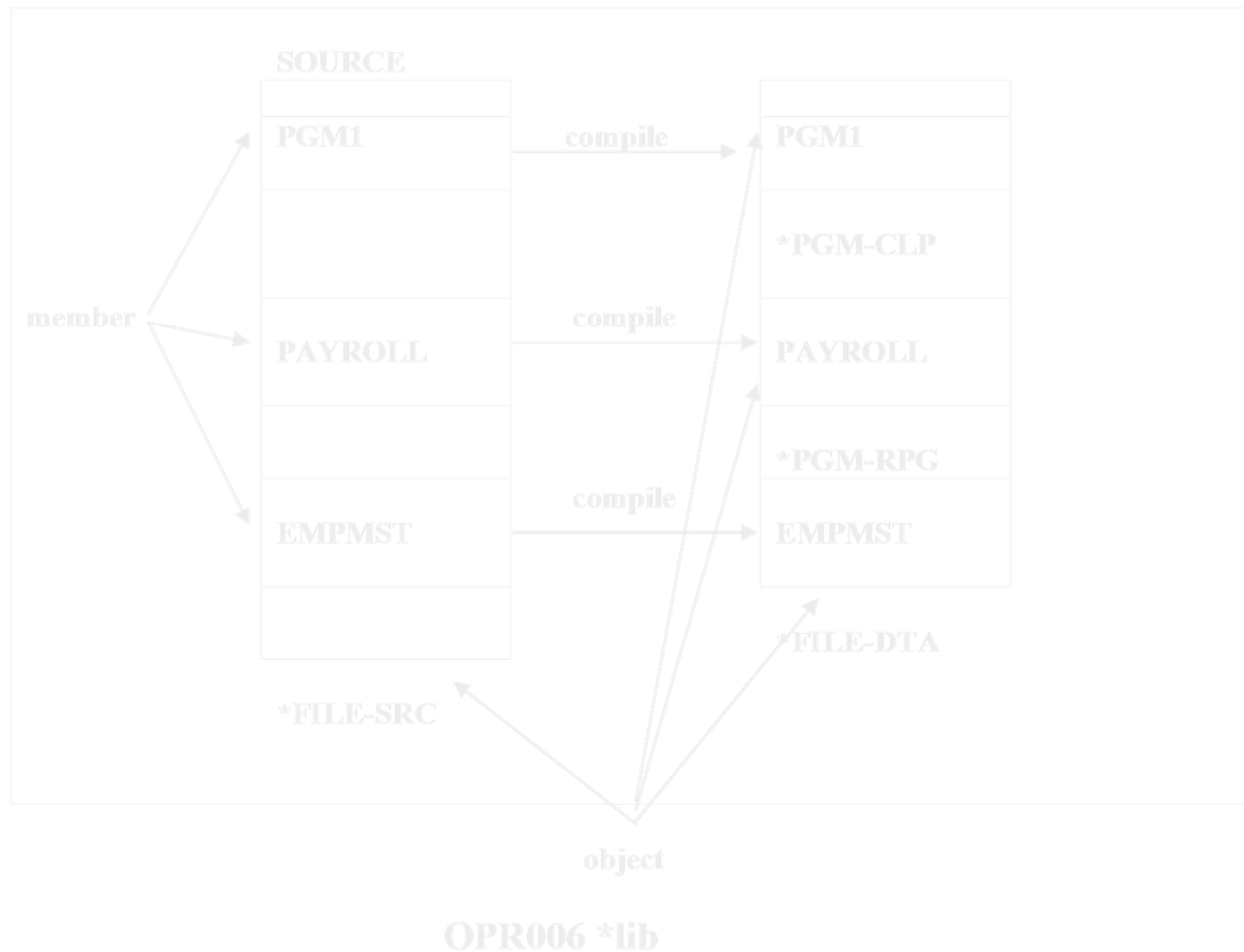
Bottom

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this
display F24=More keys

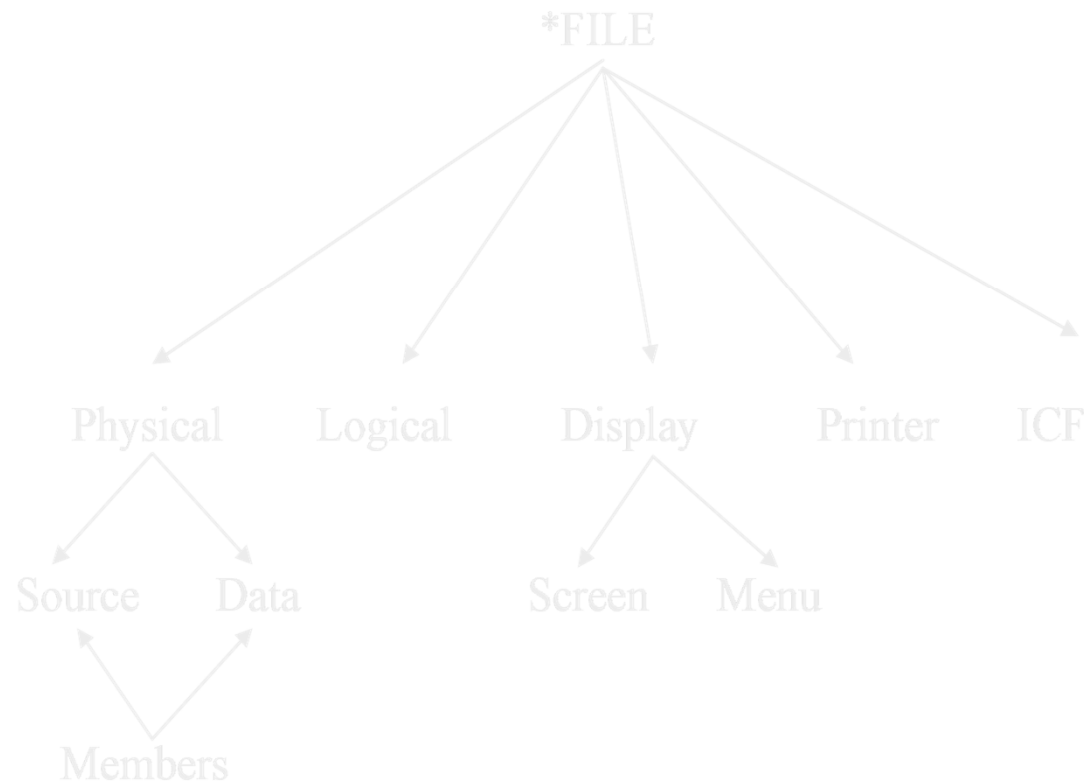
Atributele obiectelor

- Nume
- Tip
- Nume proprietar
- Data crearii
- Data salvarii
- Text explicativ
- Biblioteca
- Atribute
- Ora crearii
- Ora salvarii
- Data modificarii

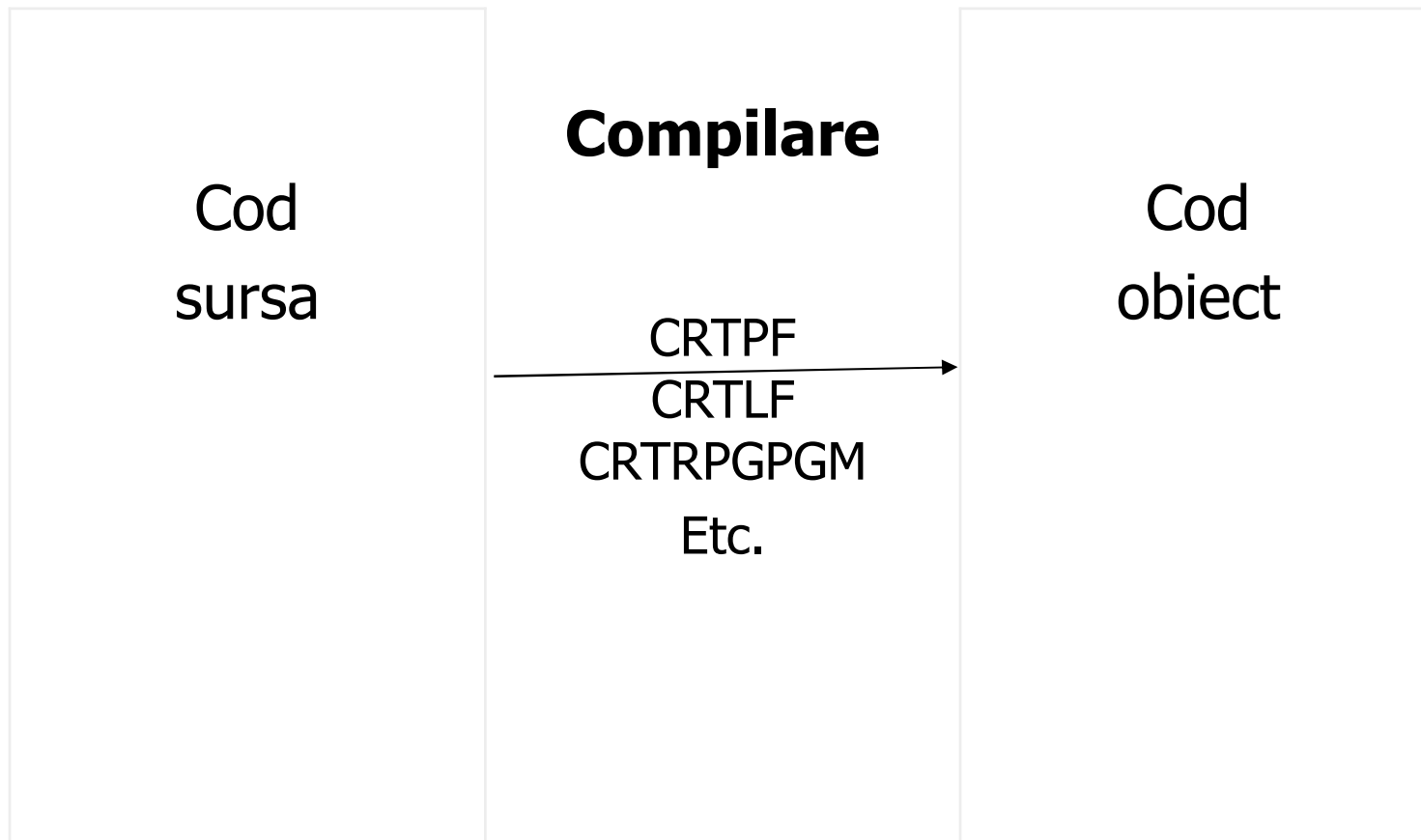
Continutul unei biblioteci



Atributele obiectelor



Definire obiect



Comenzi de compilare

Tip obiect	Comanda
<i>Physical File</i>	CRTPF
<i>Logical File</i>	CRTLFL
Program RPG (nu ILE)	CRTRPGPGM
<i>Display file</i>	CRTDSPF
<i>Program</i>	CRTPGM
Etc...	

Exemple tipuri de obiecte

(*FILE)	<i>FILE</i>
(*FLR)	<i>Folder</i>
(*JOBQ)	<i>Job Queue</i>
(*JRN & *JRNRCV)	<i>Journal and Journal Receiver</i>
(*LIB)	<i>Library</i>
(*LIND)	<i>Line Description</i>
(*MENU)	<i>Menu</i>

Securitatea

- Securitate fizica.
- Securitate a datelor.
- Nivele de securitate:
 - sistem
 - *user profiles.*
 - obiecte

Nivele de securitate

● QSECURITY:

➤ Security Level 20: (SL20).

- User ID si parola.
- Full access.

➤ Security Level 30: (SL30).

- Securitate completa a sistemului
- user ID si parola.
- Acces la obiecte pe baza de autoritate.

System Security Level

- Security Level 40: (SL40).
 - Level 30 plus.
 - Previne utilizarea neautorizata a programelor (inclusiv low-level).
- Security Level 50: (SL50).
 - Sistem special de securitate → militar
 - Orice comanda este verificata.

Profile implicite IBM

● ***Security Officer.***

➔ **QSECOFR.**

➔ Acces nelimitat la obiecte.

● ***System Administrator.***

➔ Creare / mentinere *user profiles*.

● ***System Operator.***

➔ **QSYSOPR.**

➔ Control al joburilor

➔ Functii Backup & restore.

● ***Programmers.***

➔ **QPGMR.**

➔ Acces la biblioteci de obiecte.

--

Security Menu

SECURITY

Security

System: BIGBLUE

Select one of the following:

1. Work with object authority
2. Work with authorization lists
3. Office security
4. Change your password
5. Change your user profile
6. Work with user profiles
7. Work with system values
8. Security tools

70. Related commands

Selection or command

==> _____

F3=Exit F4=Prompt F9=Retrieve F12=Cancel F13=Information Assistant

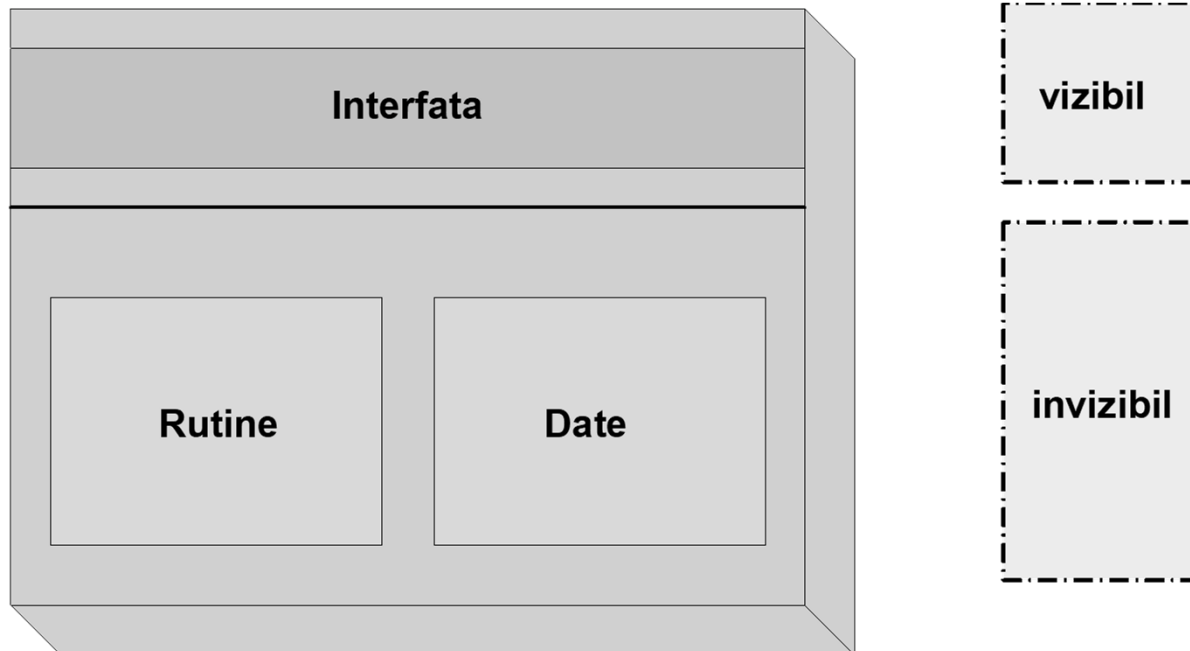
Comenzi *User-profile*

- CRTUSRPRF - Creare *user profile*.
- CHGUSRPRF - Modificare *user profile*.
- DLTUSRPRF - Stergere *user profile*.
- DSPUSRPRF - *Display user profile*.
- RSTUSRPRF - Restore *user profile*.

AS400 – abordare obiectuala

Un obiect este in mod esential o cutie neagra (*black box*) ce contine doua componente:

- O componenta ascunsa
- O componenta vizibila



Modelul obiectual - concept

Prin modelul obiectual:

- Pot fi schimbate datele fara modificarea codului,
- Se simplifica modul de reprezentare al sistemelor(pana la nivel de obiect, fara a mai fi necesara interpretarea lui in totalitate),
- Se reduce interdependenta dintre elementele unui sistem informatic,
- Creste robustetea aplicatiei soft
- Se simplifica modul de depanare a unei aplicatii: daca un obiect este "defect", doar un mic numar de rutine (aferente acestuia) pot fi "defecte"

Obiecte AS400

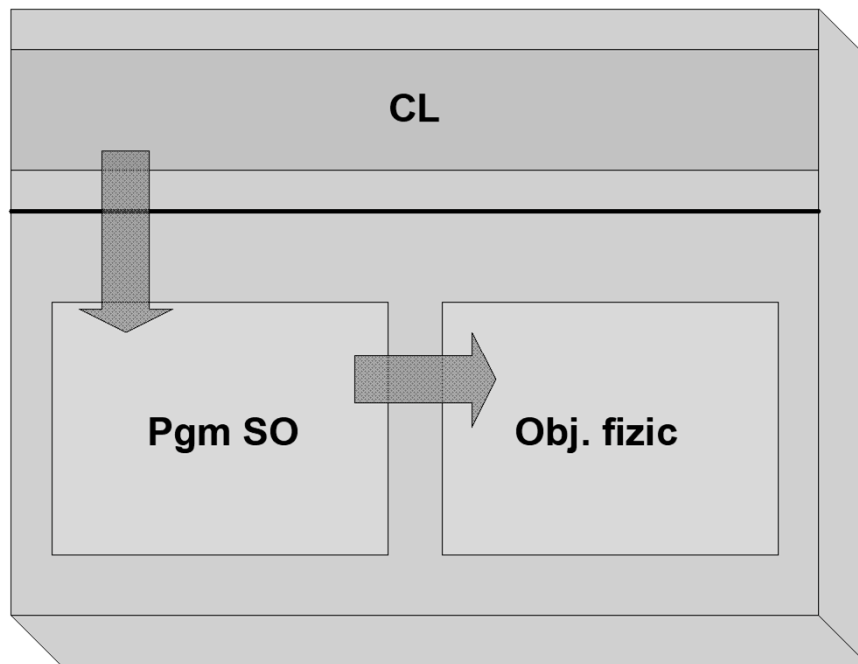
In AS400 obiectele sunt asemeni unor «cutii negre»:

- Componenta vizibila (operatii necesare manipularii obiectelor)
- Componenta ascunsa (*hidden*): rutine + date
- O variabila declarata ca tip predefinit al unui limbaj de pgm. (d.e. RPG) = **obiect**
- Reprezentarea fizica a unei asemenea variabile este *hidden*.

Accesul la obiecte

- In RPG accesul la un fisier se poate face prin intermediul a doua operatii:
 - ➔ WRITE
 - ➔ CHAIN
- Aceste operatii sunt definite la nivel abstract (i.e. structura interna a fisierului nu poate fi modificata) → doar codul intern rezultat in urma compilarii va fi dependent de acesta reprezentare:
 - ➔ Operatia standard WRITE este utilizata in cazul oricarui tip de fisier, indiferent de destinatie: PRINTER, DISK
- Structurile de date nu sunt tratate obiectual: subcampurile nu sunt capsulate, i.e. ele pot fi accesate in diferite moduri

Obiecte AS400 (1)



Componenta:

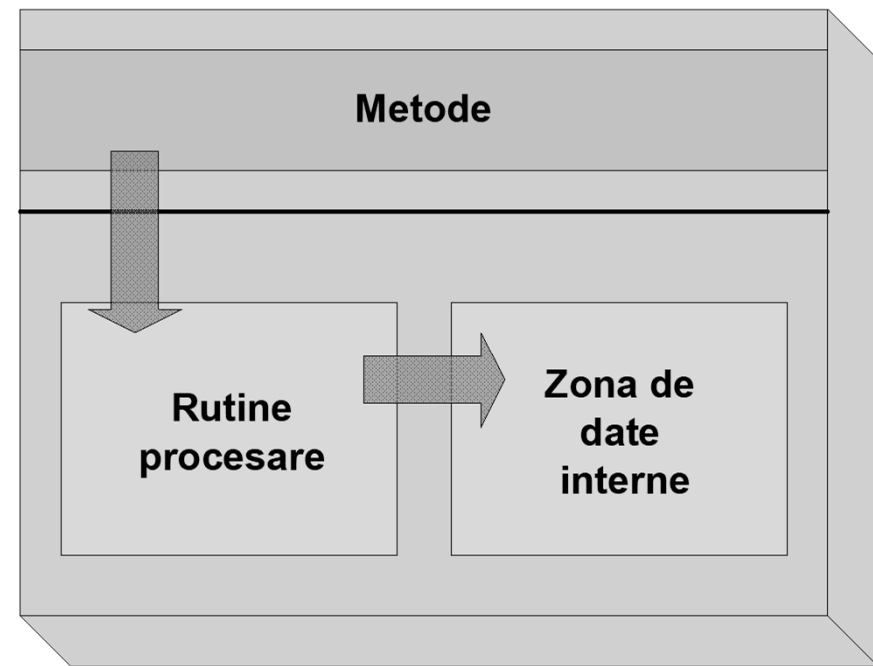
- Obiectul fizic pp-zis
- Programul (componenta SO) ce opereaza asupra fisierului fizic
- Comenzi CL ce apeleaza pgm SO

Obiecte AS400 (2)

- Singura modalitate de actiune asupra obiectelor fizice (fisierele fizice): prin intermediul CL (securitate 40, 50)
- Limbajul CL (*interfata* CL) este definita in termeni abstracti → nu este permis accesul la modul de implementare al obiectelor
- Exemplu: fie obiectul AS400: DTAARA ("data area object"); nu exista nici un mecanism care sa permita accesul la date in obiectul fizic → singura metoda: comenzi CL: CRTDTAARA si CHGDTAARA

Obiecte AS400 (3)

- Recapitulare:
ansamblul metodelor
(i.e. operatiile definite
in raport cu un obiect)
formeaza interfata
obiectului
- Cazul DTAARA:
metode (CRTDTAARA,
CHGDTAARA,
DSPDTAARA)



Obiectul DTAARA

Modelul Operator - Operand

- Modelul obiectual AS400 implementeaza ***modelul operator operand (MOO)***: operandul este o entitate pasiva si reactioneaza in raport cu operatorul activ.
- Exemplu: fie comanda CHGDTAARA; se cere modificarea unei DTAARA date:

CHGDTAARA DTAARA (D1) NEWVAL ('valoarenoua')

Mesaje

- Pentru a actiona asupra unui obiect, trebuie invocata una din metodele sale, i.e. se trimite «un mesaj» unui obiect, iar ca raspuns, acesta apeleaza o procedura.
- Un mesaj AS400 consta in:
 - Numele metodei (operatia ce trebuie sa fie executata)
 - Parametrii ce urmeaza a fi transmisi

Modelul Mesaj – Obiect

- Utilizarea CHGDTAARA cu un obiect: un mesaj este adresat obiectului (mesajul cere acestuia sa invoce metoda CHGDTAARA in vederea schimbarii datelor / actualizarea obiectului cu un set de date → **Modelul Mesaj – Obiect (MMO)**
- Diferente MOO / MMO:
 - ➔ In MMO diferitele tipuri de dependente sunt izolate in cadrul obj; in MOO diferitele tipuri de dependente sunt "imprastiate" de-alungul sistemului, in fiecare operator (operatorul selecteaza operatia ce se va executa in raport cu tipul operandului)
 - ➔ Apelul unui operator presupune apelul unei aceleiasi rutine(i.e. mesajul este un apel indirect: mesajul specifica operatia ce trebuie executata, iar obj. selecteaza metoda in raport cu ce contine mesajul) → obiectele pot raspunde diferit aceluasi mesaj

Obiecte noi in sistem (1)

- Avantajul POO consta si in adaugarea / modificarea codului existent al diverselor obj.
- Structura obj. AS400 nu este "impachetata": comenzile CL sunt separate de programele SO pe care le apeleaza (si care, la randul lor, sunt separate de obiectele din sistem)
- Exemplu: utilizand MOO se considera cda DSPFD:

```
If object is dspf then  
  Call dsp_dspf(object)  
Else if object is savf then  
  Call dsp_savf(object)
```

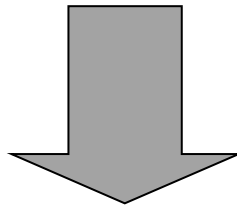
(se tine cont de faptul ca rutina poate fi apelata de toate tipurile de fisiere)

Obiecte noi in sistem (2)

- In cazul POO, codul se va gasi intr'un singur loc (impreuna cu obiectul respectiv)
- In cazul AS400, adaugarea unui nou obiect implica modificarea rutinei de lucru (care trebuie sa faca fata acestei «schimbari»). In acest caz, schimbarile sunt «imprastiate» in tot sistemul
- In locul apelului unei proceduri ce va trebui sa decida ce sa faca cu un obiect de un anumit tip, obiectul decide ce sa faca in raport cu un mesaj transmis:
 - If message is dsp then
 - Call_dsp_this_obj
 - Else if message is chg then
 - Call_chg_this_obj

Programarea cu obiecte (1)

- *Intrebare*: cum poate fi identificat un obiect a.i. sa'i fie transmis un mesaj?
- *Raspuns*: obiectele sunt variabile, iar mesajul este transmis obiectului utilizand un nume de variabila
- **Idee**:
 - Subrutina acceseaza doar datele corespunzatoare ei
 - Programul principal actioneaza direct asupra datelor (le poate modifica) → se produce o «dependenta de reprezentare»



Nu se recomanda

Programarea cu obiecte (2)

- In POO se realizeaza incapsularea (in jurul datelor / rutinelor) → accesul la date se face doar prin interfata specifica
- Daca se schimba formatul unor date, aceasta se poate face cunoscand doar operatiile afectate de aceste modificari – si nu tot programul
- In limbajul RPG (de ex.), incapsularea trebuie facuta de programator (nu exista la nivelul compilatorului posibilitatea de blocare a accesului la date)

Intrebari

1. Daca un obiect = date + cod asociat, pentru 3 DTAARA, exista cod duplicat pentru fiecare obiect?
2. Daca codul corespunzator aferent pt doua tipuri de obiecte (DSPFD pentru DSPF si SAVF) este quasi-identific, trebuie el sa fie duplicat pentru fiecare obiect?
3. Ce crestere de performanta se poate observa prin implementarea OO pe AS400 ?
4. Care este legatura intre programarea structurata si cea OO pe masini AS400 ?
5. Cum difera MMO de un apel clasic de procedura?

Raspunsuri (1)

1. Fizic, codul nu este duplicat; logic – el este. Toate obiectele apartin unei clase → aceste obiecte isi «share»-uiesc codul clasei respective. Fizic, codul nu va fi duplicat, dar logic se poate considera ca fiecare obiect posedă propria copie a codului.
2. Raspunsul se bazează pe proprietatea de mostenire
3. Se poate asocia un cost pentru fiecare apel de subrutina. In general, exista un numar mai mare de subrutine apelate in POO, decat in cea programarea structurata → POO s'a dezvoltat in ultimii ani odata cu dezvoltarea *hard*-ului (POO este si un compromis intre productivitatea programatorului si performanta programului)

Raspunsuri (2)

4. POO largeste aplicabilitatea programelor structurate: aspectele procedurale ale acesteia sunt folosite in scrierea codului intern al rutinelor obiectelor
5. A trimite un mesaj catre un obiect (pentru apel de subrutina) reprezinta un apel indirect (nu se specifica numele subrutinei) → acelasi mesaj adresat diferitelor obiecte poate conduce la apelarea unor subrutine diferite

Implementarea unor asemenea tehnici in limbaje ne-obiectuale poate conduce la confuzii (se foloseste doar tehnica apelului de subrutina)

Clase AS400 (1)

- Pentru sistemele AS400, **clasa** = **template** care descrie unul / mai multe obiecte similare
- Clasa descrie atat interfata vizibila, cat si elementele invizibile
- Toate obiectele ce apartin unei aceleiasi clase au o aceeaasi interfata si structura – dar fiecare obiect va avea un “depozit” de date asociat

Clase AS400 (2)

- Conceptul de clasa AS400 este consistent in raport cu modelul obiectual (i.e. toate obiectele de acelasi tip au un acelasi set de comenzi CL)
- Exemplu: fie clasa DTAARA ce defineste interfata si structurile pentru 3 obiectefizice DTAARA. Clasa defineste metodele valide in raport cu obiectele (i.e. CRTDTAARA, CHGDTAARA, etc). Cele 3 obiecte vor fi identice – mai putin datele stocate in interiorul lor.
- O clasa AS400 va contine o descriere a datelor aferente obiectelor din clasa respectiva.
- Obiectele dintr'o clasa *share*-uiesc codul clasei respective
- Clasele nu memoreaza date.

Mosteniri AS400 (1)

- **Mostenire** = un mecanism de *cod-sharing* prin care o clasa noua se defineste in raport cu o clasa deja existenta, *fara copierea fizica a codului*.
- Toate metodele valabile in clasa *veche*, vor fi valabile si in clasa *noua*.
- Cand o clasa mosteneste o alta clasa, o metoda definita la nivelul superclasei, poate fi re-definita in cadrul subclasei.
- Exemplu 1: clasa FILE mosteneste metode de la clasa DSPOBJD; dar la nivelul FILE vor fi adaugate 2 noi metode: DLTF si DSPFD (care refera la informatia de la nivelul clasei FILE)

Mosteniri AS400 (2)

- Exemplu 2: DSPFD afiseaza diverse categorii de informatie ce depind de tipul fisierului → aplicarea metodei se schimba la nivelul subclasei
- Clase abstracte: clase fara instante – sunt necesare pentru structurarea mai multor subclase in cadrul unei clase comune
- Mostenirea este utila mai ales in cazul schimbarilor (propagare superclasa → subclasa). Este necesara *recompilarea* codului.

Polimorfism

- **Tipuri de polimorfism:**

- ➔ **Overloading:** utilizarea aceluiasi operator pentru mai multe tipuri de operanzi. Ex.: CLEAR (RPG)

- ➔ **Binding:** cand un mesaj este trimis, compilatorul trebuie sa determine codul in raport cu clasa obiectului respectiv (aceasta decizie se ia la compilare sau la rulare)

- **Static binding:** decizia se ia la compilare (d.e. apelul de subrutina)

- **Dynamic binding:** decizia se ia in timpul rularii

- **Avantaj:** se pot introduce obiecte de tip nou fara a fi necesara recompilarea programului.

Comparatie: modelul AS400 & abordarea OO

- Implementarea obiectuala este comparabila (suporta incapsularea si abstractizarea); difera in special prin modul de utilizare a obiectelor:
 - ➔ CHGDTAARA DTAARA(D1) NEWVAL ('AUTO') – modelul operator – operand
- Modelul AS400 nu este polimorfic (exista doar o rutina fizica corespunzatoare unei comenzi CL)
- Un obiect AS400 este un obiect *pasiv*

Concluzii

- Sistemele AS400 au o implementare OO (includ conceptele: obiect, clasa, mostenire).
- Polimorfismul este integrat intr'o forma specifica (combina *overloading* cu *binding*)

