Programming with jBC

TEMENOS EDUCATION CENTRE

TEMENOS
The Banking Software Company

Objectives

At the end of the session you will able to

- Describe and differentiate what a dynamic and a dimensioned array is
- Explain the delimiters FM, VM and SM
- Create programs using jBC commands such as
  - OPEN, READ, WRITE, READU
- Create programs using jBC constructs such as
  - IF THEN ELSE
  - CASE
  - LOOP REPEAT
  - FOR NEXT
- Compile and catalog programs
- Understand scope of variables
- Understand transaction blocks

Understanding Variable Declaration

- How do we declare variables generally?
  - int score

    Example : 45
  - Char name

    Example : A
  - boolean result

    TRUE or FALSE

Understanding Arrays

- How do we store more than 1 character or a number in one variable?
  - Use arrays

- How do we tell the array how many characters or numbers it should store
  - Char Name[10]

    Array ‘name’ can store up to a maximum of 10 characters

    Example : Temenos

T	E	M	E	N	O	S
0	1	2	3	4	5	6

Name[0] = T
Name[1] = E
Name[2] = M
Name[3] = E
Name[4] = N
Name[5] = O
Name[6] = S

Points To Remember

- For the array ‘Name’
  - 10 bytes of continuous blocks of memory will get allocated
  - 10 bytes will remain blocked for ‘name’ irrespective of whether there are 10 characters of data or not
  - ‘name’ can only contain characters as it is of type char

1. If we declare an character array, can it only contain characters?

2. If we declare a numeric array, can it only contain numbers?

Break The Rules

- You don’t want your array to have a fixed length
- You don’t want your array to be bound to a data type

Welcome to Dynamic Arrays In jBC

Break The Rules

- You wish to store the string Temenos in a variable called ARR1

  ARR1 = “Temenos”

- You wish to store today’s date in a variable called ARR1

  ARR1 = “160108”

- You wish to store the number 134.67 in a variable called ARR1

  ARR1 = 134.67

Can the variable ARR1 store all types of data and data of any length?

Dynamic Arrays

~~char~~ ARR ~~[10]~~

Dynamic Arrays And T24

- How are we able to have variable length records?
  - Using dynamic arrays

- How do we store values of multiple fields in one dynamic array
  - Using delimiters such as FM, VM and SM

ASCII Decimal	**Description**
254	Field Marker
253	Value Marker
252	Sub-Value Marker

Filed1**FM**Field2**FM** Value1**VM**Value2**VM**Value3**VM**Value4**FM**Field4**FM**SubValue1**SM**SubValue2**FM**Field5

Understanding FM, VM And SM

1 Name	TemenosTrg
2.1 Address	India
2.2 Address	UK
2.3 Address	Geneva
3.1 Course Category	Technical
4.1.1 Course Name	jBASE
4.1.2 Course Name	T24
3.2 Course Category	Functional
4.2.1 Course Name	Lending
4.2.2 Course Name	Financials
5 Free Text	
6 Inputter	TRAINER.1

→ Single value field

→ Multi value field

→ Sub value field

Dynamic Array – Data Storage

TEMENOS

TemenosTrg**FM**India**VM**UK**VM**Geneva**FM**Technical**VM**Functional**FM**

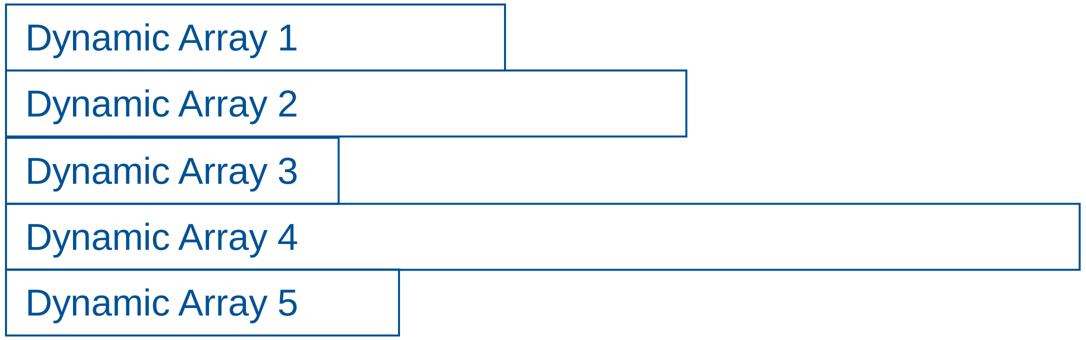
jBASE**SM**T24**VM**Lending**SM**Financials**FMFM**Trainer.1

When a field does not contain any value, there will still be a FM to delimit and say that there is field that does not contain a value now.

Dimensioned Arrays

- A dimensioned array is a group of dynamic arrays
- How many dynamic arrays form a dimensioned array?
    - You can configure that

      DIM ARR(5)



Initialing And Declaring – Is There A Difference

- Generally, a variable is declared, initialized and then used

  int score → Declaration

  score = 0 → Initialization

  if score > 100 then …… → Usage of the variable

- In jBASE
  - Variables don't have to be declared
  - It is a good practice to initialize the variable before using it

  score = 0 →Initialization

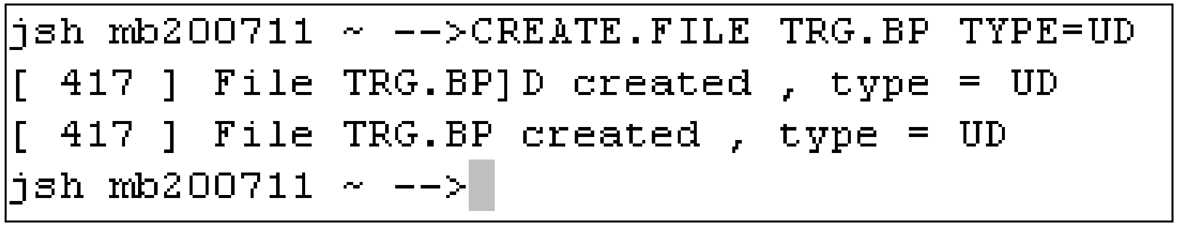
  If score > 100 then ……. → Usage of the variable

Question That You Might Have In Your Mind

Dynamic arrays do not have a data type. That being the case, how do you decide what to initialize the variable to?

Where Do We Store The Program

- Before we actually start writing code, we need to have a directory to store the code (program) we write



```
jsh mb200711 ~ -->CREATE.FILE TRG.BP TYPE=UD
[ 417 ] File TRG.BP]D created , type = UD
[ 417 ] File TRG.BP created , type = UD
jsh mb200711 ~ -->
```

Write Your First jBC Program

TEMENOS

- Program skeleton

```
*Comments

*Author

*Date of creation

*Amendment date and amendment details

PROGRAM <Program Name>

-----------

----------

END
```

Write Your First jBC Program

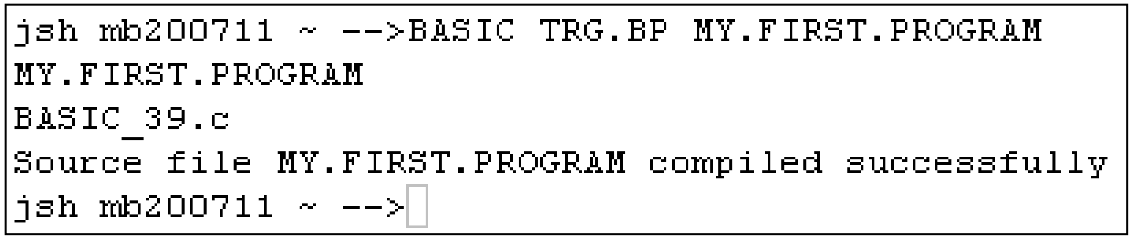
- Program to display a welcome note

```
jsh mb200711 ~ -->JED TRG.BP MY.FIRST.PROGRAM
```

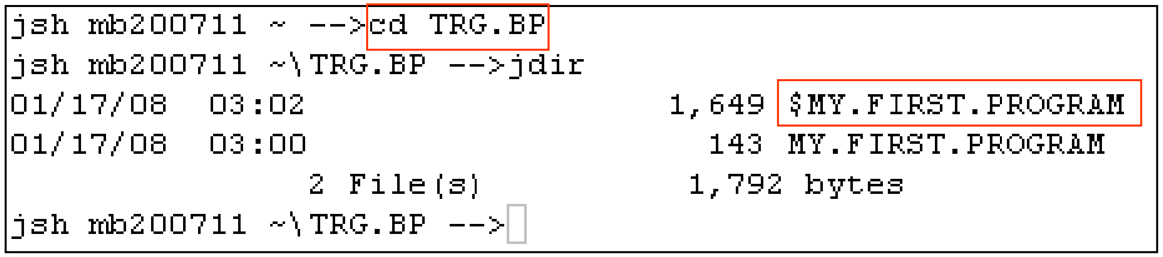
```
*Program to display a welcome note

*Temenos Training

*Jan 2008

PROGRAM MY.FIRST.PROGRAM

CRT “Welcome to Temenos Training”

END
```

Compile The Program

- Compiling the program
  - Checks for syntax errors
  - If no errors, an executable is produced and placed under the directory where the source file is



```
jsh mb200711 ~ -->BASIC TRG.BP MY.FIRST.PROGRAM
MY.FIRST.PROGRAM
BASIC_39.c
Source file MY.FIRST.PROGRAM compiled successfully
jsh mb200711 ~ -->
```

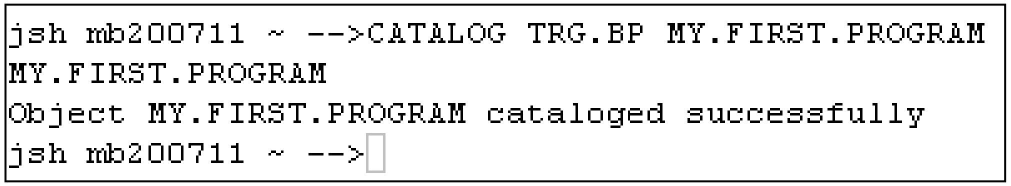


```
jsh mb200711 ~ -->cd TRG.BP
jsh mb200711 ~\TRG.BP -->jdir
01/17/08   03:02                   1,649 $MY.FIRST.PROGRAM
01/17/08   03:00                     143 MY.FIRST.PROGRAM
                 2 File(s)           1,792 bytes
jsh mb200711 ~\TRG.BP -->
```

Catalog The Program

- Process of placing the executable in a
- Environment variable that controls where the executable will be stored on cataloging : JBCDEV_BIN
- Environment variable that holds the search path of executables : PATH

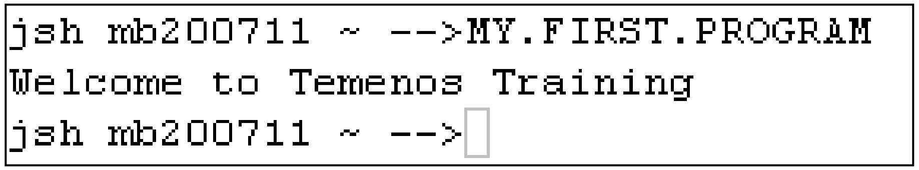
```
JBCDEV_BIN=$HOME/bin
PATH=$HOME/bin:$HOME/T24sbin:$PATH
```



```
jsh mb200711 ~ -->CATALOG TRG.BP MY.FIRST.PROGRAM
MY.FIRST.PROGRAM
Object MY.FIRST.PROGRAM cataloged successfully
jsh mb200711 ~ -->
```

Execute The Program

- Type the program name at the jsh prompt to execute the program



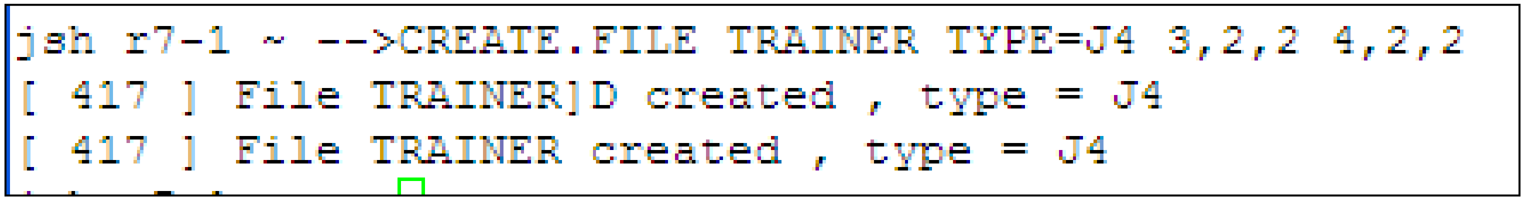
```
jsh mb200711 ~ -->MY.FIRST.PROGRAM
Welcome to Temenos Training
jsh mb200711 ~ -->
```

Task

- Create a hashed file with the below mentioned structure to store details of trainers

@ID	NAME	CLASSIFICATION	REGION	COURSES			
1	TOM	TECHNICAL	INDIA	T24	jBASE	Oracle	
2	DICK	BUSINESS	US	Retail	Treasury		
3	HARRY	TECHNICAL	UK	T24	jBASE	Oracle	DB2

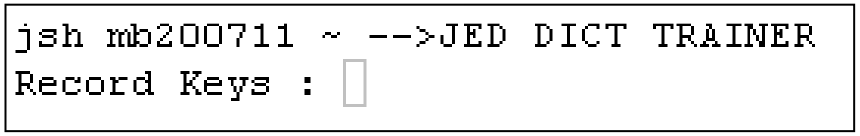
Solution – Create The File



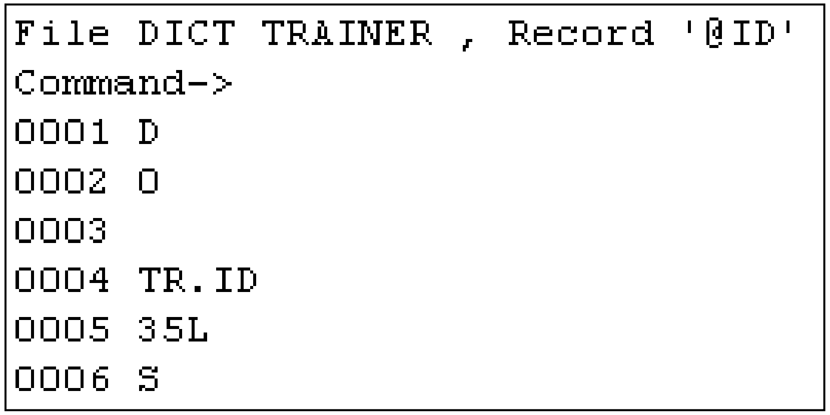
```
jsh r7-1 ~ -->CREATE.FILE TRAINER TYPE=J4 3,2,2 4,2,2
[ 417 ] File TRAINER]D created , type = J4
[ 417 ] File TRAINER created , type = J4
```

Solution – Add The Dictionary Items

- Dict definition for @ID (To hold the id if the trainer)

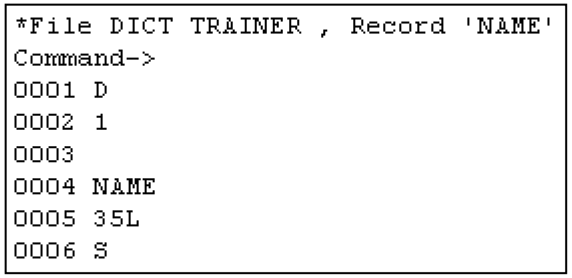


```
jsh mb200711 ~ -->JED DICT TRAINER
Record Keys :
```

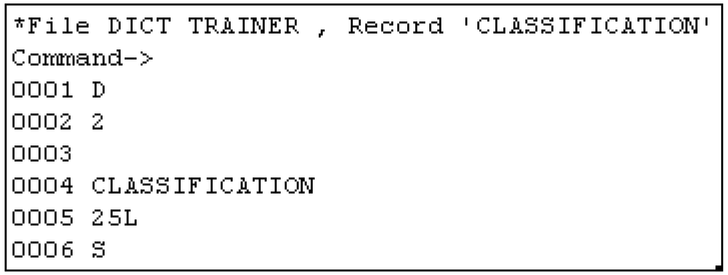


```
File DICT TRAINER , Record '@ID'
Command->
0001 D
0002 0
0003
0004 TR.ID
0005 35L
0006 S
```

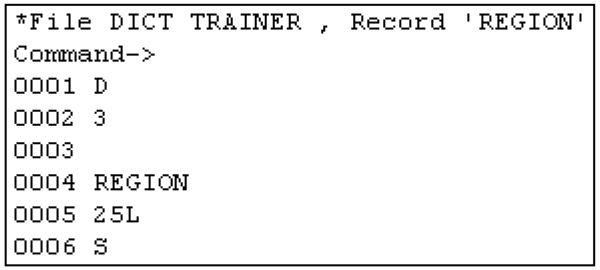
Solution – Add The Dictionary Items



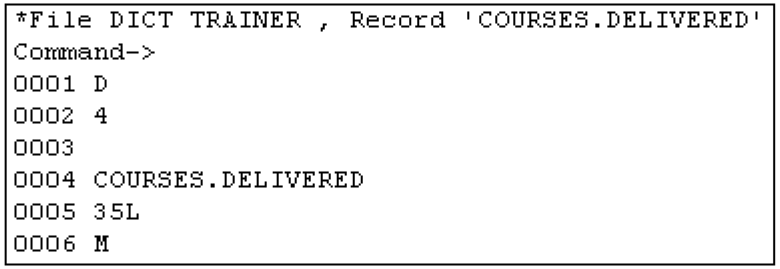
```
*File DICT TRAINER , Record 'NAME'
Command->
0001 D
0002 1
0003
0004 NAME
0005 35L
0006 S
```



```
*File DICT TRAINER , Record 'CLASSIFICATION'
Command->
0001 D
0002 2
0003
0004 CLASSIFICATION
0005 25L
0006 S
```



```
*File DICT TRAINER , Record 'REGION'
Command->
0001 D
0002 3
0003
0004 REGION
0005 25L
0006 S
```



```
*File DICT TRAINER , Record 'COURSES.DELIVERED'
Command->
0001 D
0002 4
0003
0004 COURSES.DELIVERED
0005 35L
0006 M
```

Workshop

- Create a hashed file with the below mentioned structure to store details of trainees

@ID	NAME	CLASSIFICATION	REGION	TESTS WRITTEN				SPECIALIZATION (Multimedia, programming, testing)	
1	TOM	TECHNICAL	INDIA	T24	jBASE	Oracle		Programming	
2	DICK	BUSINESS	US	Retail		Treasury		Programming	Multimedia
3	HARRY	TECHNICAL	UK	T24	jBASE	Oracle	DB2	Multimedia	

Control Structures Before You Proceed Further

TEMENOS

- IF THEN ELSE

```
IF <condition> THEN

        <statements>

END
```

```
IF <condition> THEN

        <statements>

END ELSE

        <statements>

END
```

- BEGIN CASE END CASE

```
BEGIN CASE

        CASE <variable> = <value>

                <statements>

        CASE <variable> = <value>

                <statements>

        CASE <variable> = <value>

                <statements>

        CASE 1

                <statements>

END CASE
```

Control Structures

TEMENOS

- FOR NEXT

```
FOR <variable>=<initval> TO <maxval>

<statements>

NEXT <variablename>
```

- Open LOOP

```
LOOP

  WHILE <Condition>

  <statements>

  if <condition> THEN BREAK

  <statements>

REPEAT
```

Appending Values In an Array

- **Appending Values using FM**
  - ArrayVar<-1> - Used to append values in an array using FM as delimiter
  - Example: Y.INFO<-1>="Test Info"
- **Appending Values using VM**
  - ArrayVar<FMPos,-1> - Used to append values to a position (Field) in an array using VM as delimiter
  - Example : R.CUS<EB.CUS.TEXT,-1>="THIS IS FROM TRAINIING"
- **Appending Values using SM**
  - ArrayVar<FMPos,VMPos,-1> - Used to append values to a certain multi value position (VMPos) of a field in an array using SM as delimiter
  - Note: FMPos should always contain a valid field number/position.
  - Example : R.CUS<6,2,-1>="No 50 Lake View Road, New York"

Task

Create a program that will display the following menu and perform the appropriate operations on the TRAINER file

Trainer File – Data Manipulation Menu

****

1. Insert records
2. Display records
3. Update records
4. Delete records
5. Exit

Algorithm

- Open the TRAINER file
- Create the menu
- Accept the choice from the user
- If choice is 1 (Insert)
  - Accept the @ID, name, classification, region and courses.delivered
  - Write the data on to the TRAINER file
- If choice is 2 (Display)
  - Accept the @ID
  - Check if the record exists. If it does

    Read the record from the file

    Display the record

Algorithm

- If choice is 3 (Update)
  - Accept the @ID
  - Check if the record exists. If it does

    Read and Lock the record from the file

    Accept the changes for the various fields

    Write the data on to the TRAINER file
- If choice is 4 (Delete)
  - Accept the @ID
  - Check if the record exists. If it does

    Delete the record from the file
- If choice is 5 (Exit)
  - Exit from the menu

Solution

- Open the TRAINER file

- Command to be used : OPEN
- Syntax

```
OPEN file-name TO file-variable {SETTING var} THEN|ELSE
   statements
```

- Example

```
OPEN “TRAINER” TO F.TRAINER THEN CRT “Open Successful” ELSE CRT “Unable to open
file”
```

Solution

- Create the menu and accept the choice from the user

- Commands to be used : CRT and INPUT
  - CRT : The CRT statement sends data directly to the terminal
  - INPUT : The INPUT statement accepts input from the user
- Syntax

```
CRT expression
INPUT variable
```

- Example

```
CRT "Trainer File – Data Manipulation Menu "
CRT "*************************************"
CRT " 1. Insert "
CRT " 2. Display "
CRT " 3. Update"
CRT " 4. Delete”
CRT " 5. Exit”
INPUT Y.CHOICE
```

More on CRT

- CRT can also be used to display variable values

Example : CRT Y.CHOICE

- CRT can also be used to display variable values concatenated with string constants

Example : CRT “Choice input by the user is ”**:**Y.CHOICE

‘**:**’ is the concatenation operator

Solution

- Based on user input do appropriate processing
- Command to be used : BEGIN CASE….. END CASE

Solution

```
BEGIN CASE
CASE Y.CHOICE = 1
       GOSUB PERFORM.INSERT
CASE Y.CHOICE = 2
       GOSUB PERFORM.DISPLAY
CASE Y.CHOICE = 3
       GOSUB PERFORM.UPDATE
CASE Y.CHOICE = 4
       GOSUB PERFORM.DELETE
CASE Y.CHOICE = 5
       EXIT(1)
CASE 1
       CRT “Invalid option.”
       EXIT(1)
END CASE
```

Solution

- If choice is 1 (Insert)
  - Accept the @ID, name, classification, region and courses.delivered
  - Write the data on to the TRAINER file

- Commands to be used : CRT and INPUT
  - CRT and INPUT to display text and accept user input
  - WRITE to write the data to the TRAINER data file
- Syntax

```
WRITE array-variable TO file-variable,record-id {SETTING setvar} {ON ERROR
      statements}
END
```

Solution

```
CRT "Enter details to create a new record"
R.TRAINER = '`
CRT "EMP ID: "
INPUT Y.EMP.ID

CRT "NAME: "
INPUT Y.NAME
R.TRAINER<-1> = Y.NAME

CRT "CLASSIFICATION: "
INPUT Y.CLASSIFICATION
R.TRAINER<-1> = Y.CLASSIFICATION

CRT "REGION: "
INPUT Y.REGION
R.TRAINER<-1> = Y.REGION

CRT "COURSE DELIVERED: "
CRT "If the trainer has delivered multiple courses delimit values using comma ,”
INPUT Y.COURSE.DELIVERED
```

Contents of R.TRAINER

Nick

NickFMTechnical

NickFMTechnicalFMIndia

Solution

```
Y.COUNT = DCOUNT(Y.COURSE.DELIVERED,',') ⟶ Counts the number of values delimited by
                                           delimiter comma
* Change the commas to value markers
        FOR Y.COURSE.COUNT = 1 TO Y.COUNT
            Y.CD = FIELD(Y.COURSE.DELIVERED,',',Y.COURSE.COUNT,1)
            R.TRAINER<4,-1> = Y.CD ⟶ Append values using VM as delimiter
        NEXT Y.COURSE.COUNT
```

Content of R.TRAINER after the FOR loop has executed completely

Nick**FM**Technical**FM**India**FM**Oracle**VM**DB2

Write the record to the file

TEMENOS

```
WRITE R.TRAINER TO F.TRAINER,Y.EMP.ID SETTING V.ERR.VAR
ON ERROR
       CRT “Record could not written”
       CRT “Reason: “:V.ERR.VAR
END
```

Solution

- If choice is 2 (Display)
  - Accept the @ID
  - Check if the record exists. If it does read the record from the file and display the record
- Commands to be used : CRT and INPUT
  - CRT and INPUT to display text and accept user input
  - READ to read the file
- Syntax

```
READ array-variable FROM file-variable,record-id {SETTING setvar}
   {ON ERROR statements} THEN|ELSE statements
```

Solution

```
CRT "Which record do you want to display"
INPUT Y.EMP.ID

READ R.TRAINER FROM F.TRAINER,Y.EMP.ID SETTING Y.ERR.VAR ELSE
    CRT "Unable to read record”
    CRT “Reason: “:Y.ERR.VAR
    EXIT(1)
END

CRT "Name: ":R.TRAINER<1>
CRT "Classification: ":R.TRAINER<2>
CRT "Region: ":R.TRAINER<3>
CRT "Course Delivered: ":R.TRAINER<4>
```

Solution

- If choice is 3 (Update)
  - Accept the @ID
  - Check if the record exists. If it does

    Read and Lock the record from the file

    Accept the changes for the various fields

    Write the data on to the TRAINER file

- Commands to be used : CRT and INPUT
  - CRT and INPUT to display text and accept user input
  - READU to read and lock the file
  - WRITE to write data to the file

Insight into READ

READ R.TRAINER FROM F.TRAINER,1

@ID	NAME	CLASSIFICATION	REGION	COURSES			
1	TOM	TECHNICAL	INDIA	T24	jBASE	Oracle	
2	DICK	BUSINESS	US	Retail	Treasury		
3	HARRY	TECHNICAL	UK	T24	jBASE	Oracle	DB2

READ R.TRAINER FROM F.TRAINER,1

READU

- Use the READU statement when you wish to read and lock a record
- Syntax

```
READU array variable FROM file variable, record id {SETTING
   setvar} {ON ERROR statements} {LOCKED statements} THEN|ELSE
   statements
```

Insight into READU

User 1 - READU R.TRAINER FROM F.TRAINER,1

@ID	NAME	CLASSIFICATION	REGION	COURSES			
1	TOM	TECHNICAL	INDIA	T24	jBASE	Oracle	
2	DICK	BUSINESS	US	Retail	Treasury		
3	HARRY	TECHNICAL	UK	T24	jBASE	Oracle	DB 2

User 2 - READU R.TRAINER FROM F.TRAINER,1

Is record locked?

Lock information in memory

Record 1 in TRAINER locked by User 1

Yes

READU – When is the lock released

The lock taken by the READU statement will be released by any of the following events:

- The record is written to by the same program with WRITE, WRITEV or MATWRITE statements.
- The record is deleted by the same program with the DELETE statement.
- The record lock is released explicitly using the RELEASE statement.
- The program stops normally or abnormally.

More code

```
CRT "Which record do you wish to update?"
INPUT Y.EMP.ID
READU R.TRAINER FROM F.TRAINER, Y.EMP.ID LOCKED
CRT “Locked by Port “:SYSTEM(43):” -retrying”
END ELSE
CRT “Record does not exist”
END
Y.OPERATION = ‘Y’ ; *Get into the loop the first time
LOOP
WHILE Y.OPERATION EQ 'Y' DO
    CRT "Enter the field number and the new value delimited by _"
    CRT " 1.NAME 2.CLASSIFICATION 3.REGION 4.COURSE DELIVERED“
    CRT “Value for COURSES DELIVERED should be delimited with ‘,’ “
    INPUT Y.FN.FV  ; *Accept field number and field value
    Y.FIELD.NUM = FIELD(Y.FN.FV,'_',1,1) ; *Extract the field number
    Y.FIELD.VALUE = FIELD(Y.FN.FV,'_',2,1) ; *Extract the field value
```

Solution

```
     *Only field 4(Courses delivered) needs to be handled differently
     IF Y.FIELD.NUM NE 4 THEN
        R.TRAINER<Y.FIELD.NUM> = Y.FIELD.VALUE
     END
     ELSE
          R.TRAINER<4> = '' ; *Delete all values in field COURSES.DELIVERED
          Y.COUNT = DCOUNT(Y.FIELD.VALUE,',')
          FOR Y.COURSES.COUNT = 1 TO Y.COUNT
              Y.CD = FIELD(Y.FIELD.VALUE,',',Y.COURSES.COUNT,1)
              R.TRAINER<4,-1> = Y.CD
        NEXT Y.COURSES.COUNT
     END
     CRT "Do you wish to update another field"
     INPUT Y.OPERATION
REPEAT
*Write the new record to the file
WRITE R.TRAINER TO F.TRAINER,Y.EMP.ID SETTING V.ERR.VAR ON ERROR
       CRT "Record could not written"
       CRT "Reason: ":V.ERR.VAR
END
```

Solution

- If choice is 4 (Delete)
  - Accept the @ID
  - Check if the record exists. If it does

    Delete the record from the file

- Commands to be used : CRT and INPUT
  - CRT and INPUT to display text and accept user input
  - DELETE to delete the record from the file
- Syntax

```
DELETE file variable,record id  {SETTING setvar} {ON ERROR
   statements}
```

- Example

```
DELETE F.TRAINER,Y.EMP.ID SETTING Y.ERR.VAR ON ERROR

CRT "Unable to delete record"

CRT "Reason: ":Y.ERR.VAR

END
```

Workshop

Write a program to display the following menu and perform appropriate actions based on the option chosen. All operations should be based on the TRAINEE file.

Trainee File – Data Manipulation Menu

****

1. INSERT (To insert a new record)
2. UPDATE (To update an existing record)
3. DISPLAY (Display the details of a given record)
4. DELETE (To delete an existing record)
5. Exit

Once a record id is accepted, a check to see if the record already exists has to take place

If the record id exists

    Insert operation should not be permitted

    All other operations can be permitted

If the record id does not exist

    Insert operation should be permitted

    All other operations should not be permitted

Task

Add a new field called designation in the TRAINER table.

Write a program which will check the number of courses delivered by the technical trainers. If the number is more than 5, then update the field DESIGNATION to SENIOR else JUNIOR. Use dimensioned arrays while writing the program

Algorithm

Step 1: Edit the dict file TRAINER]D add the field ‘DESIGNATION’

Step 2: Open the TRAINER file

Step 3: Select all trainer ids whose designation is technical

Step 4: Remove one trainer id from the selected list

Step 5: Read the record

Step 6: Get the value for the field COURSE.DELIVERED

Algorithm

Step 7: Count the number of courses delivered

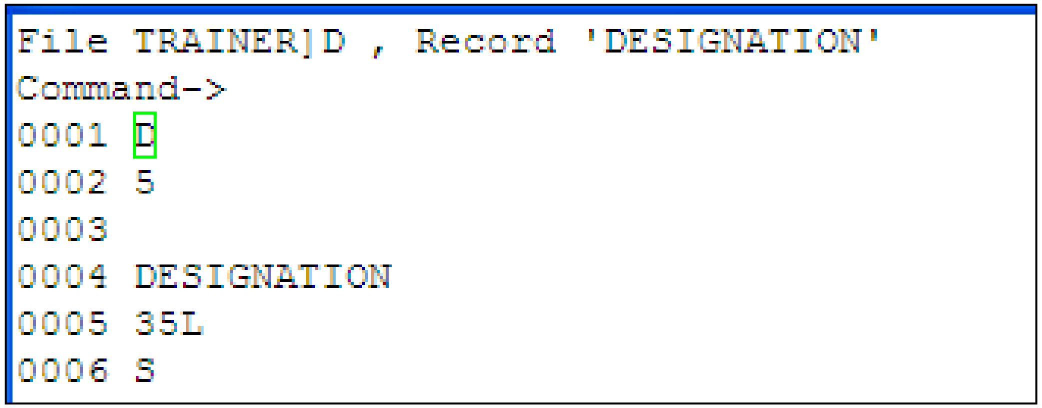
Step 8: If the count is greater than 5, set the field designation to senior else junior

Step 9: Flush data to disk

Step 10: Repeat steps 4 to 9 for all trainers using the loop and repeat statements

Solution

- Edit the dictionary file TRAINER]D and add the field 'DESIGNATION'



```
File TRAINER]D , Record 'DESIGNATION'
Command->
0001 D
0002 5
0003
0004 DESIGNATION
0005 35L
0006 S
```

- Open the TRAINER file

```
OPEN "TRAINER" TO F.TRAINER THEN CRT "Open Successful" ELSE CRT "Unable to open file"
```

Solution

TEMENOS

- Select all technical trainers

- Command to be used : EXECUTE
  - Assign the select statement to a variable
  - Execute the select statement stored in the variable

- Syntax

```
EXECUTE Selectstmt {RTNLIST return variable}
```

- Example

```
SELECT.STATEMENT = 'SELECT TRAINER WITH CLASSIFICATION EQ TECHNICAL'

EXECUTE SELECT.STATEMENT RTNLIST KEY.LIST
```

Solution

- Remove trainer ID from the selected list
- Command to be used : REMOVE
- Syntax

```
REMOVE variable FROM array SETTING setvar
```

- Example

```
REMOVE Y.TRAIN.ID FROM KEY.LIST SETTING POS
```

Solution

- Read the record
- Command to be used : MATREAD
- Syntax

```
MATREAD array variable FROM file variable, record id {SETTING
   setvar} {ON ERROR statements} {THEN|ELSE statements}
```

- Example

```
DIM Y.MAT.REC(10)

MATREAD Y.MAT.REC FROM F.TRAINER, Y.TRAIN.ID ELSE

    CRT "UNABLE TO READ RECORD"

END
```

Solution

- Get the value for the field courses delivered and count the number of courses delivered

```
Y.COURSES.DELIVERED = Y.MAT.REC(4)

Y.COUNT = DCOUNT(Y.COURSES.DELIVERED,VM)
```

Solution

- If the count is greater than 5, set the field designation to senior else junior

```
IF Y.COUNT GT 5 THEN

    Y.MAT.REC(5)<1> = 'SENIOR'

END

ELSE

    Y.MAT.REC(5)<1> = 'JUNIOR'

END
```

Solution

- Flush data to disk
- Command to be used : MATWRITE
- Syntax

```
MATWRITE array ON file variable, record id {SETTING setvar}
   {ON ERROR statements}
```

- Example

```
MATWRITE Y.MAT.REC ON F.TRAINER, Y.TRAIN.ID
```

Workshop

Add a new field called STATUS in the TRAINEE table.

Write a program which will check the number of tests passed by the trainee. If the number is more than 5, then update the field STATUS to PERMANENT else PROBATION. Use dimensioned arrays in the program. Use MATREADU to read. Refer the www.jbase.com website for details on the MATREADU statement.

Variables - Scope

- What is meant by variable scope?

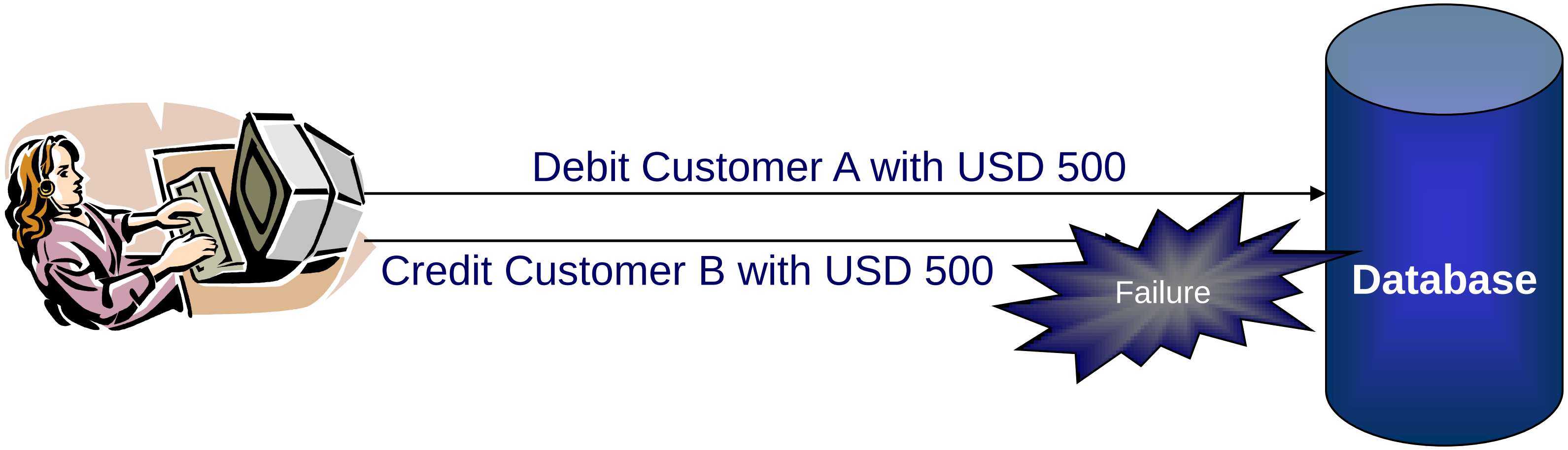
Transaction Management

What Is A Transaction Block?

A cheque to

- Debit Customer A with USD 500
- Credit Customer B with USD 500

Reason : House Rent



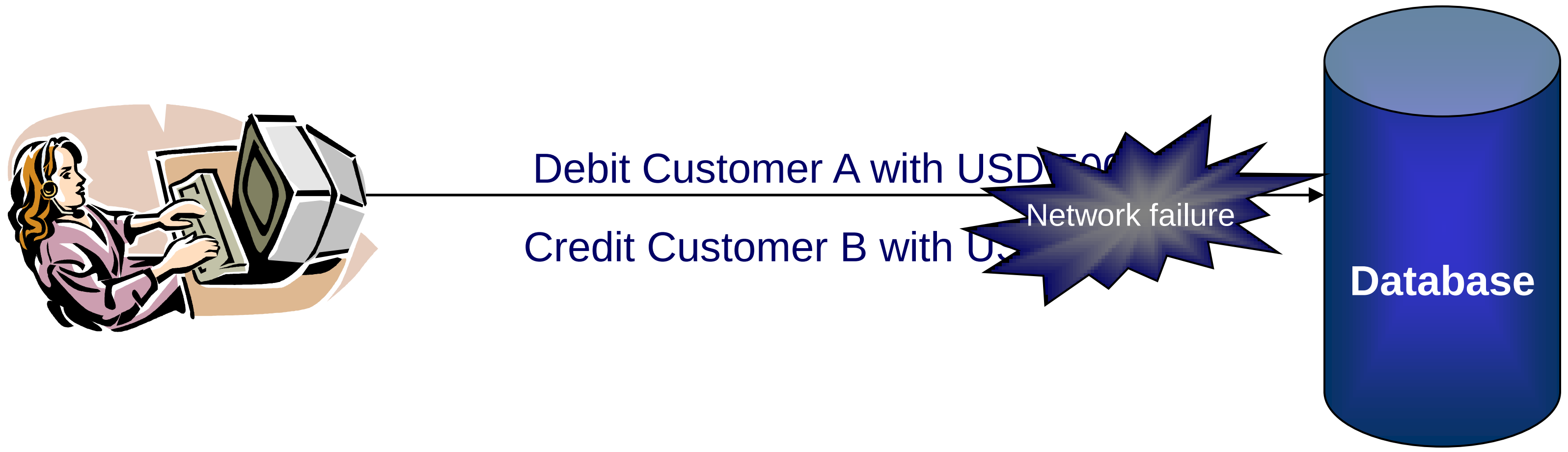
Lead to inconsistent Data

What Is A Transaction Block? (Cont.)

A cheque to

- Debit Customer A with USD 500
- Credit Customer B with USD 500

Reason : House Rent



Either both are saved or none are saved

Transaction block in jBASE

```
TRANSTART

<statements>

<statements>

TRANSEND
```

Transaction Processing

Workshop

You have already written code for

“Write a program which will check the number of tests passed by the trainee.

If the number is more than 5, then update the field STATUS to PERMANENT else PROBATION. Use dimensioned arrays in the program.

Use MATREADU to read. Refer the www.jbase.com website for details on the MATREADU statement.”

Amend this program and incorporate transaction management so that all records selected are updated or none are updated

Useful jBASE commands

- INDEX
  - Returns position of character in a string
- FIND
  - Returns the position of a string in a dynamic array
- LOCATE
  - Finds the position of an element within a specified dimension in a dynamic array

Quiz

1. Dynamic arrays are
    1. Variables that can store only character data
    2. Variables that can store any amount of data
    3. Variables that need to be declared before being used

2. Dimensioned arrays are
    1. A collection of dynamic arrays
    2. Declared using the MAT statement
    3. Variables that can hold character and numeric data alone

3. The system delimiters are (Choose all appropriate)
    1. FM
    2. VM
    3. SM
    4. DM
    5. AM

4. The command to compile a jBC program is
    1. COMPILE
    2. BASIC
    3. CATALOG
    4. RUN

5. READ statement reads and locks a record (TRUE/FALSE)

6. When the WRITE statement is executed within a transaction block, write always happens in the buffer until TRANSEND is encountered (TRUE/FALSE)

Quiz

7. OPEN command actually opens a file in jBC (TRUE/FALSE)

8 . A transaction block is

   1. Set of statements within a TRANSTART and TRANSEND
   2. Set of statements within a program
   3. Set of statements executed from the jsh prompt

9. Common variables can be accessed from

   1. Any program that has the insert file included. This file should contains the definition of the common variables
   2. Any program
   3. Any program running in the same session

10. Some commonly used looping structures in jBC are (Choose all that are appropriate)

    1. IF THEN ELSE
    2. CASE
    3. FOR NEXT
    4. LOOP REPEAT

Summary

You will now be able to

- Describe and differentiate what a dynamic and a dimensioned array is
- Explain the delimiters FM, VM and SM
- Create programs using jBC commands such as
  - OPEN
  - READ
  - WRITE
  - READU
- Create programs using jBC constructs such as
  - IF THEN ELSE
  - CASE
  - LOOP REPEAT
  - FOR NEXT
- Compile and catalog programs

Summary (Cont.)

You will now be able to

- Explain scope of variables
- Describe transaction block
- Use transaction management related commands in jBC programs

Authoring Contributions:

Alagammai(First Edition, 2001) – Temenos India Private Ltd.

Alagammai(Second Edition, 2004) – Temenos India Private Ltd.

Alagammai(ThirdEdition, 2008) – Temenos India Private Ltd.

Kalaiselvi (Fourth Edition, 2010) - Temenos India Private Ltd

Thankful Acknowledgements:

Temenos Corporate Training Team