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

- How do we declare variables generally?
  - int score

    Example : 45
  - Char name

    Example : A
  - boolean result

    TRUE or FALSE

- 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

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

- 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?

- 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

- 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?

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

- 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 FM	Field Marker	^
253 VM	Value Marker	]
252 SM	Sub-Value Marker	/

  - SAMFIELD =

Field1**FM**Field2**FM** Value1**VM**Value2**VM**Value3**VM**Value4**FM**Field4**FM**SubValue1**SM**SubValue2**FM**Field6

  - REC<EB.CUS.LANGUAGE> = '2'
  - SAMFIELD<1> = Field1
  - SAMFIELD<1,1,1> = Field1
  - SAMFIELD<3,2> = Value2
  - SAMFIELD<5,1,2> = SubValue2

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

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

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

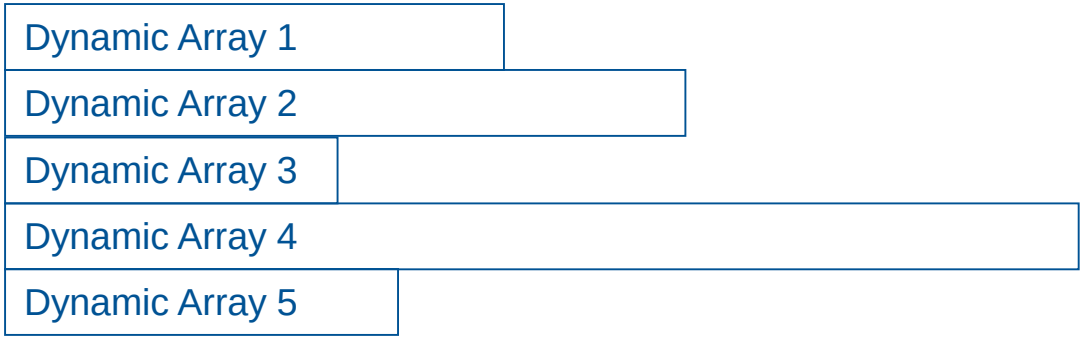
TemenosTrg**FM**India**VM**UK**VM**Geneva**FM**Technical**VM**Functional**FM**

jBASE**SM**T24**VM**Lending**SM**Financials**FM****FM**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.

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

    DIM ARR(5)



- Program skeleton

```
*Comments

*Author

*Date of creation

*Amendment date and amendment details

PROGRAM <Program Name>

-----------

----------

END
```

- 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
```

- FOR NEXT

```
FOR <variable>=<initval> TO <maxval>
<statements>
NEXT <variablename>


FOR YI = 1 TO 100
   CRT YI
NEXT YI
```

- Open LOOP

```
LOOP
  WHILE <Condition>
  <statements>
  if <condition> THEN BREAK
  <statements>
REPEAT


LOOP
         WHILE ID > 100
  ID ++
  CRT ID
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“

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

- 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

- 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

- 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"

END ELSE

        CRT "Unable to open file"

END
```

- 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
```

- 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

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

```
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
```

```
PROGRAM TEST
GOSUB INIT
GOSUB PROCESS
RETURN
INIT:
        CRT 'INIT'
RETURN
PRCESS:
        CRT 'PROCESS'
RETURN

END
```

- 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
```

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

Contents of R.TRAINER

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

Nick

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

NickFMTechnical

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

NickFMTechnicalFMIndia

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

```
Y.COUNT = DCOUNT(Y.COURSE.DELIVERED,',')
* 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
        NEXT Y.COURSE.COUNT
```

Counts the number of values delimited by delimiter comma

Append values using VM as delimiter

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

```
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
```

- 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
```

```
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>
```

- 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

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

- 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
```

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.

```
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 THEN
        CRT 'RECORD FOUND'
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,'_',1,3) ; *Extract the field number
    Y.FIELD.VALUE = FIELD(Y.FN.FV,'_',2,1) ; *Extract the field value
```

```
    *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
```

- 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
```

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

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

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

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

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

```
File TRAINER]D , Record 'DESIGNATION'
Command->
0001 D
0002 S
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”
```

- 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
```

- 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
```

- 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
```

- 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)
```

- 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
```

- 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
```