IBC233 Chapter 8

**Chapter objectives: Create your own personalized CL command to display the system level**

**Write an RPGLE program to process a database file and a display file.**

**Lab Requirements:**

**Demonstrate your DSPSYSLVL command which shows the current system level and includes your name on various screens**

**Bonus mark available if you have the time and inclination to get the \*PRINTER option working**

**(DSPSYSLVL in QGPL demonstrates the \*PRINTER feature for the bonus mark)**

**Hint in Tug.ca magazines archives Seneca Update January 2010, Volume 25, Number 3**

**Demonstrate your interactive program STUDENTRPG**

Start an RDi and a Client Access Session.

# CL Command Review

**PGM** starts a CL program and defines parameter list

DCL Declares variables: data types can be   
Decimal (\*DEC), Character (\*CHAR), Logical (\*LGL) Signed Integer(\*INT) and Unsigned Integer(\*UINT)

DCLF Declare a file - physical, logical or display attribute

RCVF receives a record from a display, physical or logical file

**SNDRCVF** - sends a record format to user's terminal with current states of indicators and output fields,   
- waits for the user to input data and press Enter or a function key,   
- receives input fields and response indicators into the program.

**RTVOBJD** - Retrieve Object Description. Can find object owner, or the level of the system when the object was created.

CHGVAR Change the contents of a variable

GOTO Goes to another part of the program

MONMSG Monitor Message - checks for operating system messages

DOWHILE Executes a loop until a condition is met. The condition is tested at the beginning of the loop.

DOUNTIL Executes a loop until a condition is met. The condition is tested at the end of the loop.

**SELECT**-

ENDSELECT: Tests multiple conditions

IF-ELSE Tests one condition.

DO Start of a logical block.

ENDDO Ends a logical block or a loop.

ENDPGM ends a CL program

# Part A

**Objectives:**

* **Create a CLLE command that shows the system level of the operating system by examining QCMD in QSYS**

# The DSPSYSLVL command can be run from the IBC233LIB library.

# The RTVOBJD command is not allowed interactively, but, we can investigate it by typing RTVOBJD and pressing F4

# We are going to look at an IBM program object called QCMD.

# 

Look for the object type, owner and syslvl parameters and put in the following special values and field names.

# OBJ QCMD OBJTYPE \*PGM OWNER &OWNER SYSLVL(&LEVEL)

The command didn’t work, but you can retrieve it with the F9 key. Do this.

==>RTVOBJD OBJ(QCMD) OBJTYPE(\*PGM) OWNER(&OWNER) SYSLVL(&LEVEL)

You may have not known that the owner of an object can be retrieved with this command. By pressing F4 you see this parameter option. Let us try one option that does not show in the sample that was discussed.

What parameter will tell us the user that created the object? CRTUSER.

What do the letters R, V and M mean with the SYSLVL parameter?

V = VERSION

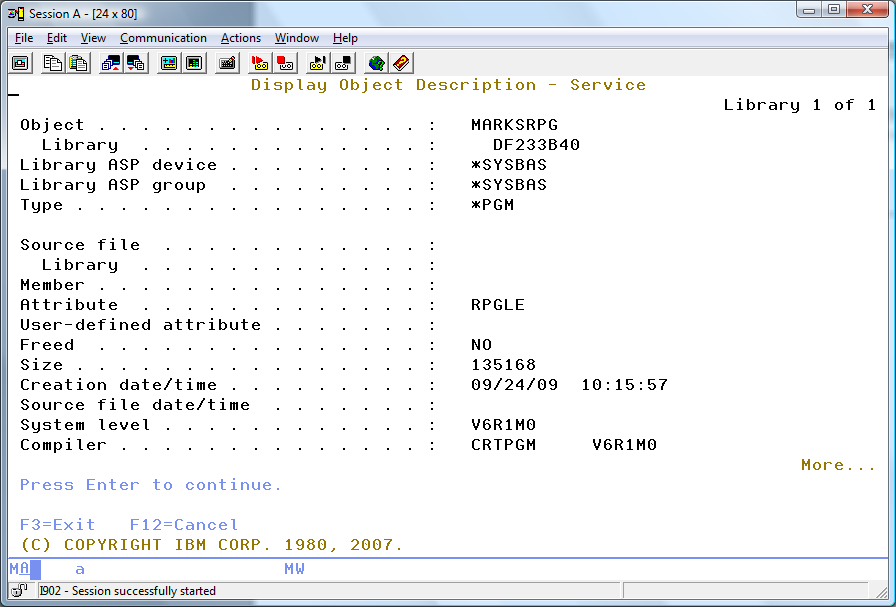
R = RELESE LEVEL

M = MODIFICATION LEVEL

(hint – use F1)

Check your program MARKSRPG for the system level. Type DSPOBJD and press F4. The required parameters for this are object name and object type.

If you select option 8 to display service attributes, you will see a screen similar to the one below.



# The MARKSRPG program was created with the system at V6R1M0. If the system is upgraded to V7, the MARKSRPG program will still show the same setting unless you compile it under the new upgraded operating system level.

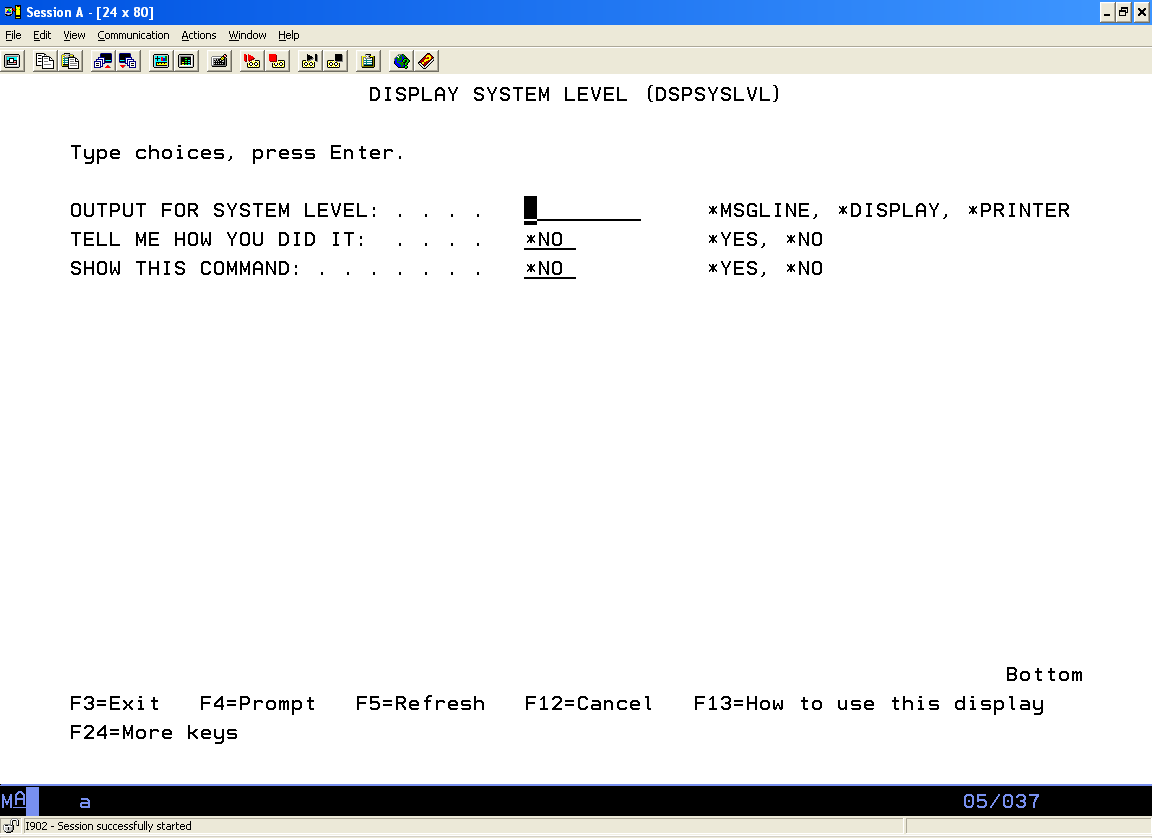
# Usually when the system is upgraded we can trust that the QCMD \*PGM object will reflect the new level right away.

# This will allow us to develop a new CLLE command that checks QCMD for the operating system level.

Our DSPSYSLVL command will have one required parameter and two parameters with default values

You are going to have your name show in a lot of areas to prove this is your DSPSYSLVL command.

DSPSYSLVL (F4)



Message line or \*MSGLINE was used here:

# 

# Solution:

# This solution was discussed in class. If you missed class, get notes from another student. A tutor may also be able to help you.

# Here are some pieces for the solution:

# The program that processes the command

Program name: DSPSYSLVLP

Program type: CLLE

Parameters for program: Output

Solution

ShowCmd

The program can be tested at the command line with the following:

==>**CALL DSPSYSLVP Parm('\*MSGLINE' '\*NO' '\*NO')**

Parameters are passed to a CLLE program by specifying one or more variables on the PGM line at the start of the program.

# The following command can be used to show a message on the message line.

SNDPGMMSG MSG('System Level is' \*BCAT +

&SYSLVL \*BCAT +

'according to' \*BCAT +

'Russell Pangborn''s New CLLE Command')

MSGTYPE(\*COMP)

You would insert your name in the appropriate spot

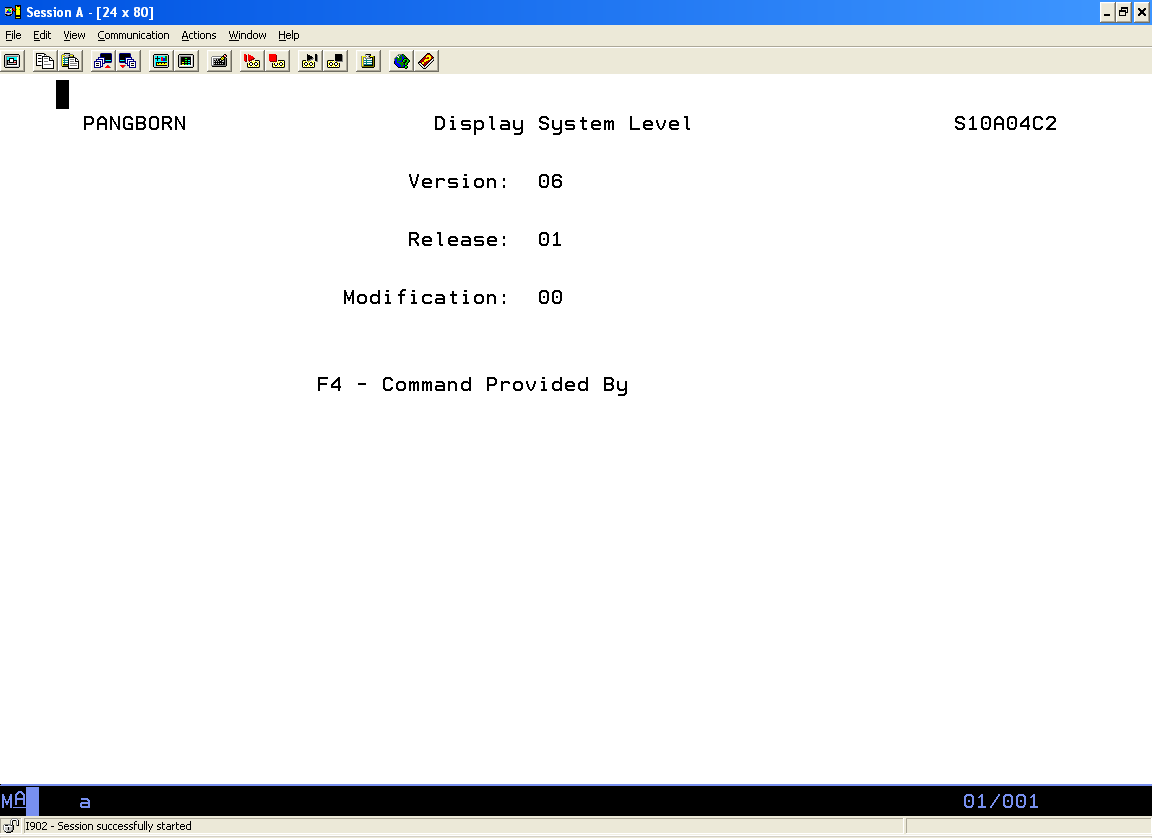
==>**CALL DSPSYSLVP Parm('\*DISPLAY' '\*NO' '\*NO')**

The following command will get the modification level of 00 from the variable &SYSLVL

&SYSLVL = 'V06R01M00'

CHGVAR &SYSLVLM %SST(&SYSLVL 8 2)

(Will return “00” by going to position eight for a length of two.)



In addition to program source code you need to enter source code for the actual command.

Here is partial code for DSPSYSLVL.cmd:

CMD 'DISPLAY SYSTEM LEVEL'

PARM KWD(OUTPUT) +

TYPE(\*CHAR) +

LEN(9) +

RSTD(\*YES) +

VALUES(\*MSGLINE \*DISPLAY \*PRINTER) +

MIN(1) +

PROMPT('OUTPUT FOR SYSTEM LEVEL:')

(the following PARMS do not use the MIN keyword:

PARM KWD(SOLUTION) +

??

DFT( ) put what you want for the default value here for this

parameter that does not require an entry

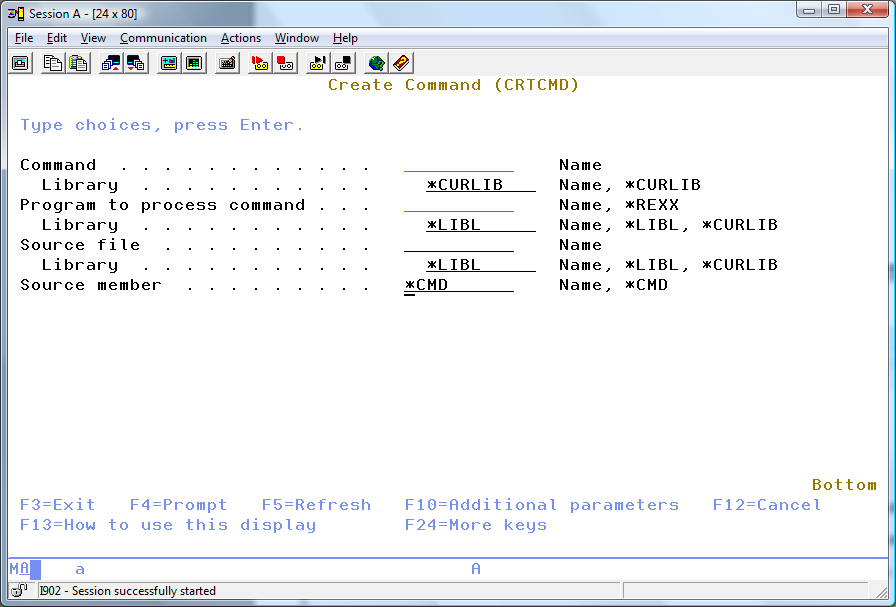
PARM KWD(SHOWCMD) +

?? +

DFT( ) put what you want for the default value here for this

parameter that does not require an entry

CREATING THE COMMAND

**Part B**

**Objectives:**

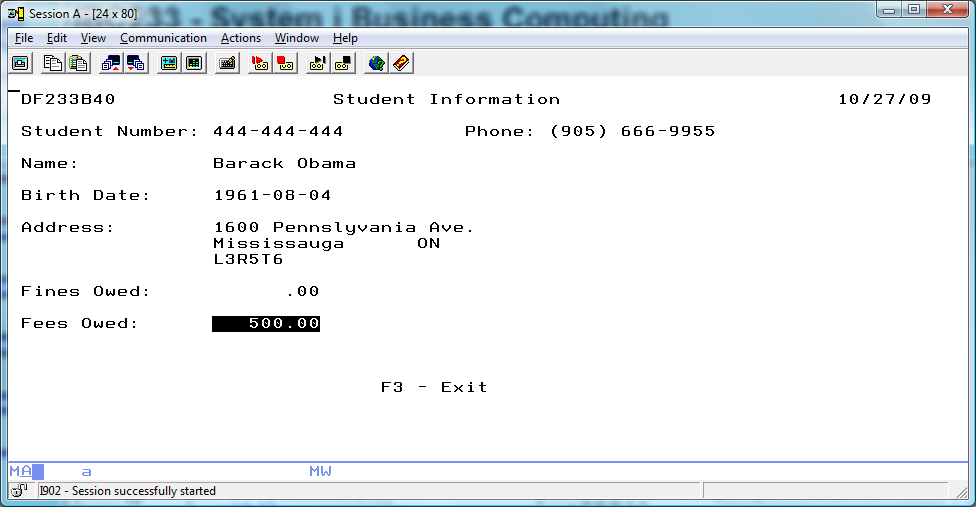
* **Use an externally described file to fill your display file interactive screen with appropriate fields**
* **Write a display file program, STUDENTRPG to display the student information on screen and send the messages to the message queue.**

**In your RDi Session:**

Using Screen Designer, design an interactive screen to show the contents of a student record after it has been read from the STUDENT file.

Create a member called STUDENTDSP in QDDSSRC and open it with Screen Designer

Your finished screen will look like the following:



For now, create a field on Record1 called FEESOWED that is to contain the Fees Owed, which is read from the STUDENTS file.

What is the length of this field? Is it a five digit with two decimal places field or is it a seven digit field? What is the length of the first name field? Is the Birth Date stored as a character or date field?

You can guess at these things and inevitably make a few mistakes that will need to be corrected or take advantage of the benefits of externally described files. An externally described file not only contains the data like telephone numbers and addresses – it contains information about the fields used in the file. Field types, field lengths and column headings used by query tools when showing field contents are stored with the file. Utilities like DFU or Screen Designer can tap into that information.

Delete your one field that you had put on RECORD1, We will now get accurate field names, types and lengths to use.

Use the show in table feature on the STUDENTS file and select Fields. (Make sure you are not trying to use this on the STUDENTS DDS code)

From the Field Table tab, click on the Student Number field and bring it up to your screen.

From this simple activity you save a lot of work and a lot of mistakes. The field has the correct length and type, it is already named and if you had used an Edit Word in the DDS to create this field it will be used on your screen field.

According to the screen shot above, a user name and a date show at the corners of you screen. Set this up.

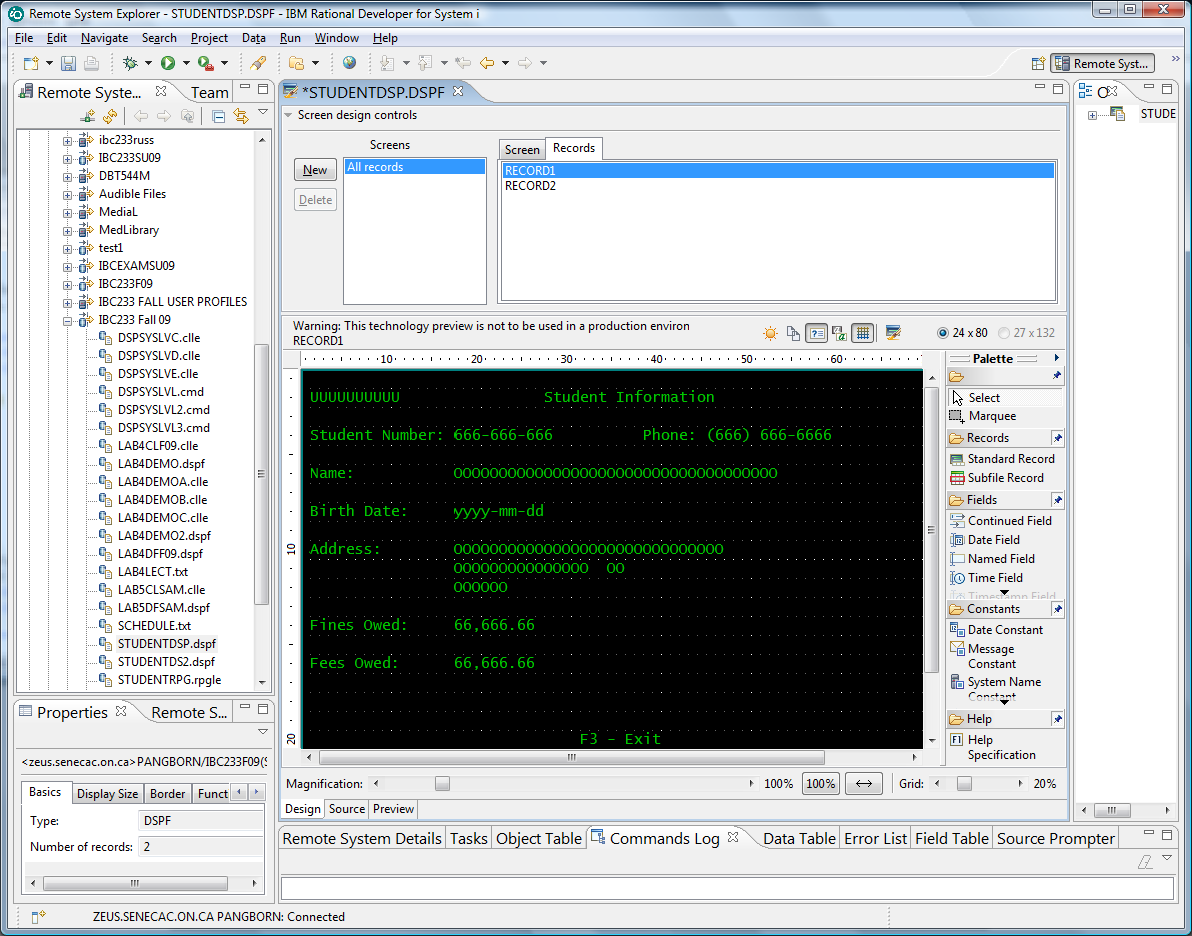
Create a text constant for the Student Number – “Student Number:”

Right click and copy the student number text and move it up to where the title “Student Information” is to appear.

Now that you have copied this text, just keep pasting it in and moving it to where all the other field prompts show.

Grab the Properties tab in the lower right corner and move it up upper left corner. (as shown below)

When the focus is on “Student Number:” at the top of the screen, you can change the actual text in the Properties box.



Click on each “Student Number:” text and make a quick change in your properties box to the correct field text.

After the text has been changed to “First Name:” , “Birth Date:” etc , drag all the fields onto the screen beside the appropriate text.

Each field has been placed on the screen with the correct field type and length, but one thing was assumed. The fields are set up as input/output fields. Since we are only showing the Student File records, we do not want the user to have the illusion that anything they type into the field will be actually be applied as updates. So, put your focus on each field and use the conveniently placed property box to change all fields as “output only” fields.

Note the length of the FirstName and LastName fields. Add one to this and provide a non referenced FullName field in the appropriate location.

Show how a referenced field is coded by providing the code for Student Number

Name Name Ref Field Data Dec Use Loc Loc Functions

Type Length Type Pos Line Pos

\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ \_\_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It was easy to place the field on the screen. You did not have to write the above code. You should be able to see this type of code and identify what it means. What does the value you wrote under Loc Line mean?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You will be including additional features for this display file later.

Compile and exit from Screen Designer.

Code an RPGLE program called STUDENTRPG that will read and display all the records in the STUDENT file.

A starter program is provided below:

FileName File File File Device

Type Designation Format

F STUDENTDF C F E WORKSTN

F STUDENTS I F E DISK

/FREE

READ STUDENTS;

DOW (NOT %EOF);

EXFMT RECORD1;

READ STUDENTS;

ENDDO;

\*INLR = \*ON;

RETURN;

/END-FREE

After you get your program working add the following features:

First and Last Names placed appropriately in a FullName Field

F3 exits the program when not at end of file

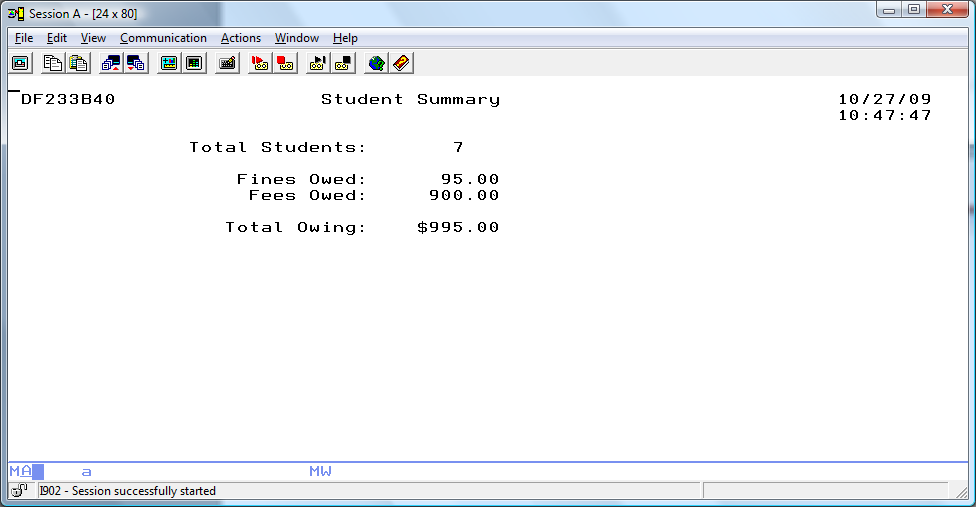
Numeric field editing (look at Telephone Number)

Finesowed in reverse image if there is a value other than 0 in it.

Feesowed in reverse image if there is a value other than 0 in

A student Summary Screen shows after end of file or pressing the F3 Key

Your name shows on the Summary Screen



#### To view a sample program add IBC233LIB to your library list and call STUDENTRPG