INFO3105 Week 1 Class 2

Review

- IBM Developer for z/OS software installation
- Lab 1 Program creation Exercise
- Chapter 1
    - 4 divisions
    - Margins
    - Picture and Value clauses
    - Perform, move, compute, if

Overview of Mainframe Terminology

The architecture for the IBM mainframe technology (Z Series class processors and the ZOS operating system) dates back to the mid 1960's. At that time a large Data Center system would run you about 5.5 million dollars to buy or a monthly charge of $115K to rent. Comparably the average house sold for about 6K. Fast forward to today, the cost of the Watson Computer that IBM used on Jeopardy was about 3 Million dollars and delivered about 20,000 times the computing power.

Despite the predominance of mainframes in the business world, these machines are largely invisible to the general public, the academic community, and indeed many experienced IT professionals. Instead, other forms of computing attract more attention, at least in terms of visibility and public awareness. That this is so is perhaps not surprising. After all, who among us needs direct access to a mainframe? And, if we did, where would we find one to access? In truth, we are *all* mainframe users, whether we realize it or not.

How big are these machines? The following graphics and descriptions are taken from IBM's Intro to ZOS slides, the first one demonstrates the size from some different perspectives:

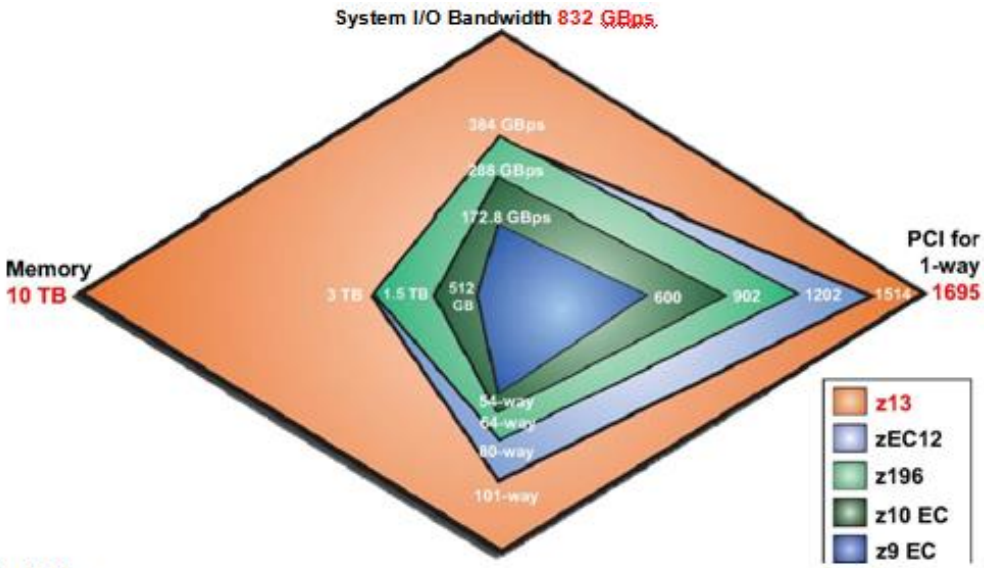


Figure 1-2 Platform design: The z13 versus its predecessors

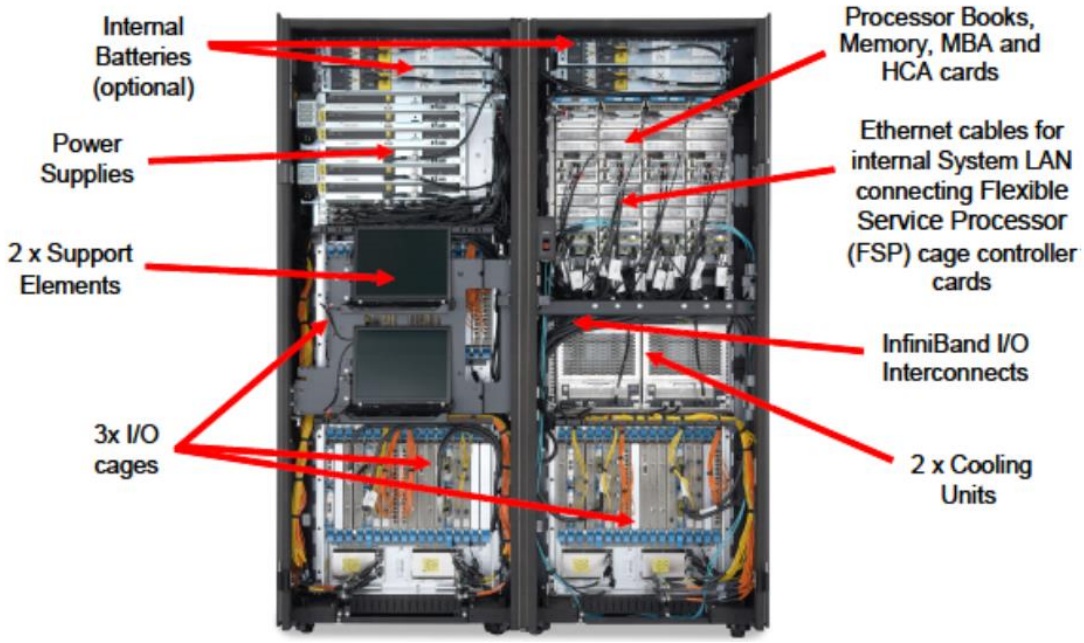
The physical boxes are not that large (relative to what they use to be), and are typically the size of large fridges now:

System z Business Class and Enterprise Class

From IBM Literature "The System z Business Class (BC) with a focus on small to midrange enterprise computing, delivers an entry point with very granular scalability and an unprecedented range of capacity settings to grow with the workload. It delivers unparalleled qualities of service to help manage growth and reduce cost and risk.

The BC server further extends System z leadership by enriching its flexibility with enhancements to the just-in-time capacity deployment functions in a single frame housing. The BC provides for a maximum of up to 10 configurable CPs (Central Processors).
The BC shares many of the characteristics and processing traits of its bigger brother the Enterprise Class (EC). This model also delivers granular scalability and capacity settings on a much larger scale targeted to very high end processing needs. It has a larger frame to house the extensive capacity to support greater processing requirements. The EC offers up to 64 configurable CPs and is considered IBM's flagship platform." The new z13 model offers 141 configurable CPs (configurable as LPARs - see description below).

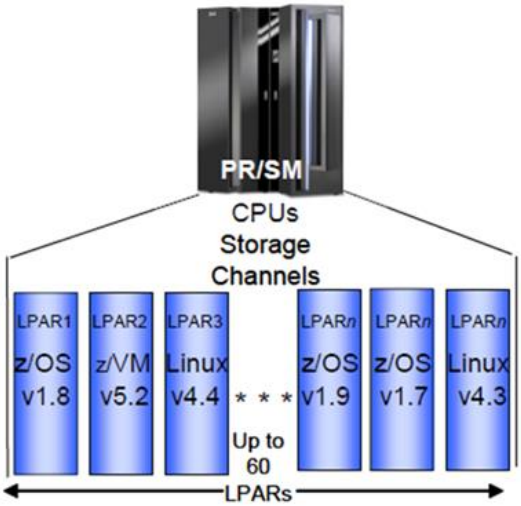
Inside the Box:



Notice there are no disk drives, this is basically a box that houses the RAM and processors.

Logical Partitions (LPARs) or Servers

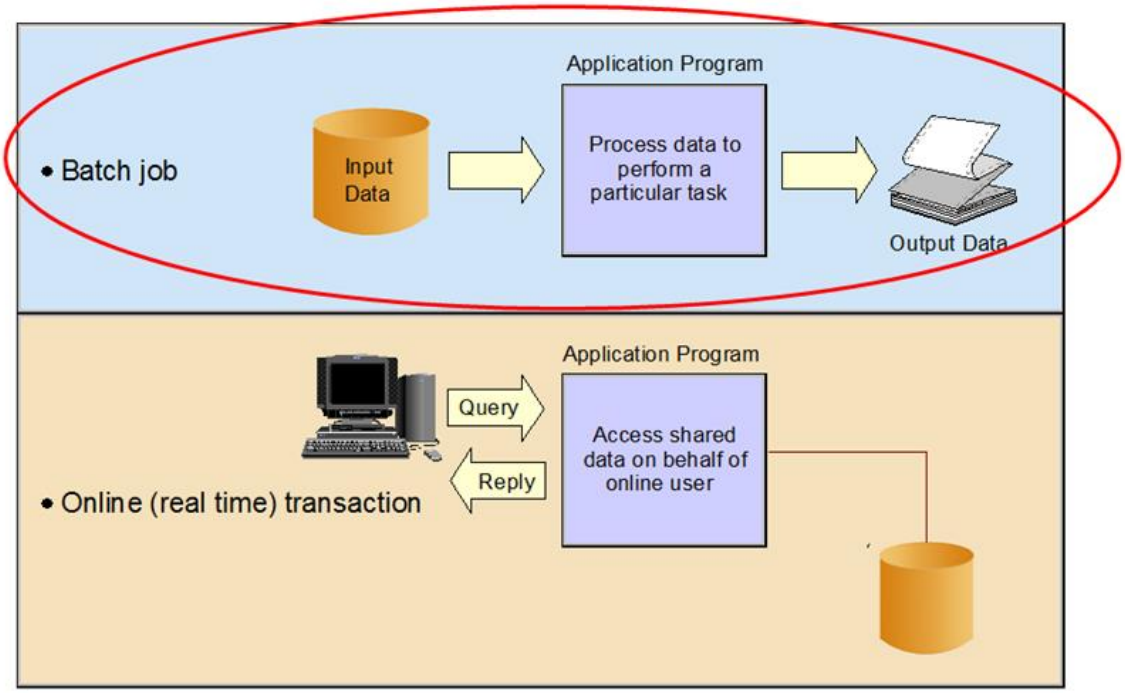
- A system programmer can assign different operating environments to each partition with isolation
- An LPAR can be assigned a number of dedicated or shared processors.
- Each LPAR can have different storage (CSTOR) assigned depending on workload requirements.
- The I/O channels (CHPIDs) are assigned either statically or dynamically as needed by server workload.
- Provides an opportunity to consolidate distributed environments to a centralized location



Today's machines can be configured with up to 140 (previous Z10 allowed 60 LPARS) Logical Partitions or servers within a single box. Practical limitations of memory size, I/O availability, and available processing power usually limit the number of LPARs to less than these maximums. Each LPAR is considered an isolated and distinct server that supports an instance of an operating system (OS). The operating system can be any version or release supported by the hardware. In essence, a single mainframe can support the operation of several different OS environments. System administrators assign portions of memory to each LPAR; memory also known as central storage (CSTOR) cannot be shared among LPARs. CSTOR in past literature may also be referred to as main storage, provides the system with directly addressable, fast-access electronic storage of data. Both data and
programs must be loaded into central storage (from input devices) before they can be processed by the CPU.

- **LPARs are the equivalent of a separate mainframe for most practical purposes**
- **Each LPAR runs its own operating system**
- **Devices can be shared across several LPARs**
- **Processors can be dedicated or shared**
- **When shared each LPAR is assigned a number of logical processors (up to the maximum number of physical processors)**
- **Each LPAR is independent**

Typical mainframe workloads



A ***batch job*** is submitted on the computer, reads and processes data in bulk, and produces output. A batch job can last for hours. While batch processing is possible on distributed systems, it is not as commonplace as on mainframes because distributed systems often lack:

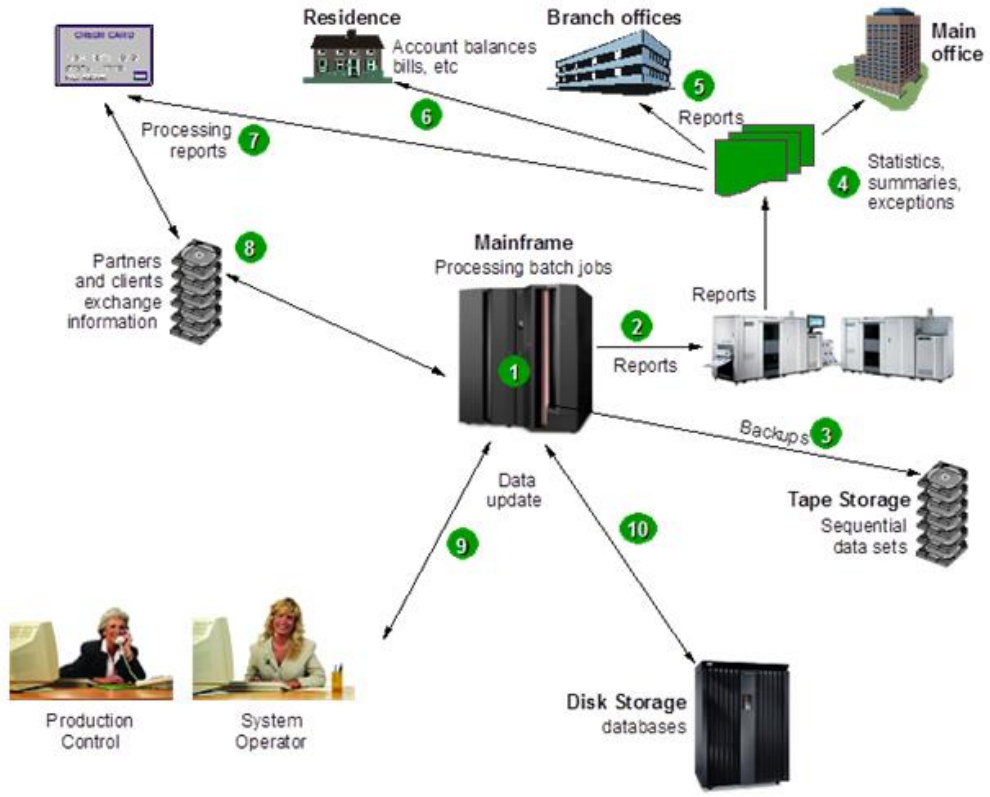
- Sufficient data storage
- Available processor capacity or *cycles*
- Management of system resources and job scheduling.
- We'll focus on batch type programs in this course as this is what you'll most likely be doing for intro and co-op jobs

Mainframes serve a vast number of *online transaction processing* (OLTP) systems. These are often mission-critical applications that businesses depend on for their core functions.

Some industry uses of online systems:

- Banks – (TD in London/Mississauga, Scotiabank in Stratford,etc.) , ATMs, banking/teller systems for customer service
- Insurance – Agent systems for policy management and claims processing
- Travel and transport – Airline reservation systems
- Manufacturing – Inventory control, production scheduling
- Government – Tax processing, license issuance and management.
- Large Retail- eg. Walmart, etc.

Typical batch use



- **z/OS, IBM's premier zSeries operating system, is a highly secure, scalable, high-performance enterprise operating system on which to build and deploy traditional and Java-enabled applications, providing a comprehensive and diverse application execution environment.**

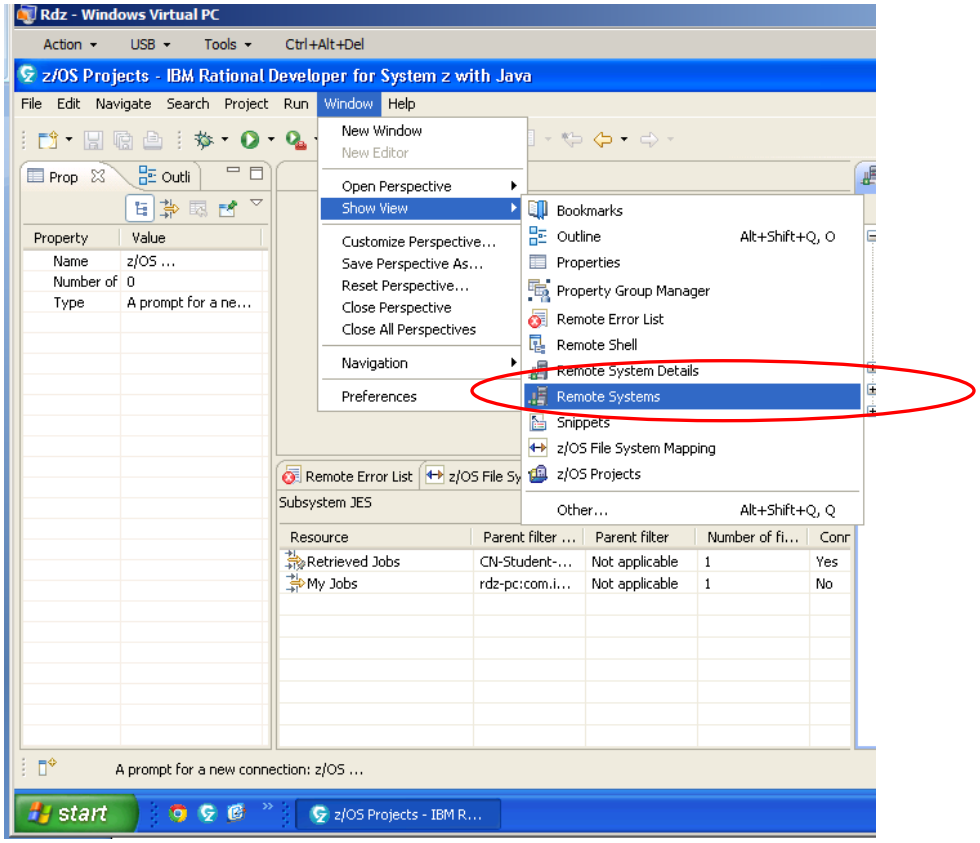
Timesharing Option – TSO

You will be given an account to sign on to the Marist Host shortly. Your first task will be to logon onto **TSO**. TSO allows users to create an interactive session with the z/OS® system. TSO provides a single-user logon capability and a basic command prompt interface to z/OS. Most users work with TSO through its menu-driven interface, Interactive System Productivity Facility (**ISPF**). This collection of menus and panels offers a wide range of functions to assist users in working with data files on the system. ISPF users include system programmers, application programmers, administrators, and others who access z/OS. In general, TSO and ISPF make it easier for people with varying levels of experience to interact with the z/OS system.

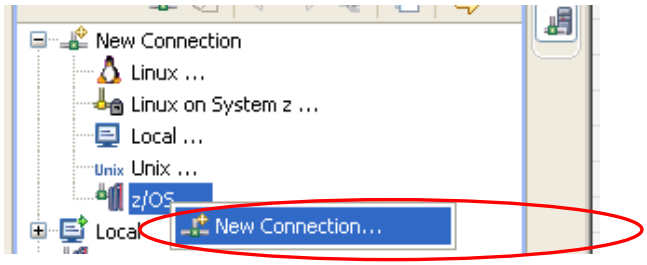
In a z/OS system, each user is granted a user ID and a password authorized for TSO logon. Logging on to TSO requires a 3270 display device or, more commonly, **a TN3270 emulator** running on a PC.

Your task for today is to obtain the account and password and setup your COBOL source program on the mainframe. You will create a source member in a data construct known as **PDS** (Partioned Data Set). You can think of a PDS as a folder on your machine, and then within that folder you will create a member (.cbl file on the PC).

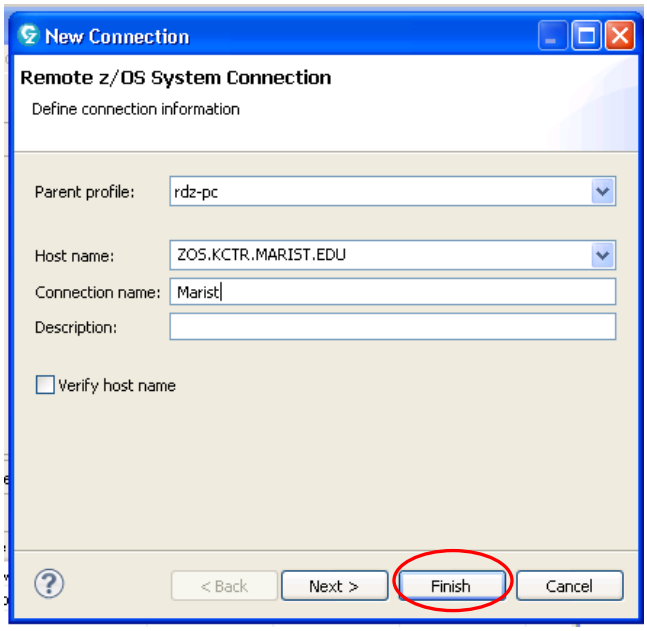
The first thing we need to do/confirm is establish a connection with a remote host. To do this we'll need to get into the remote systems view (Window→Show View→Remote Systems):



Next we'll establish a new **z/OS** connection:



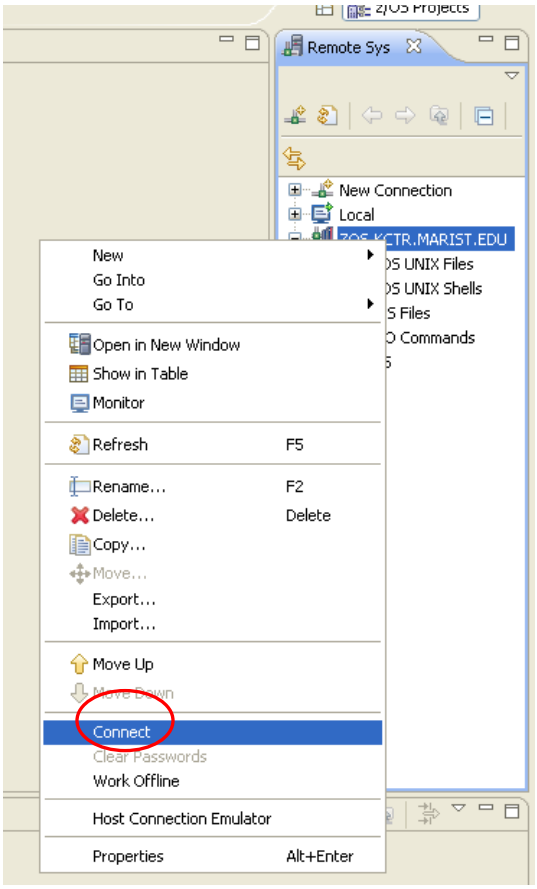
Add the following credentials:



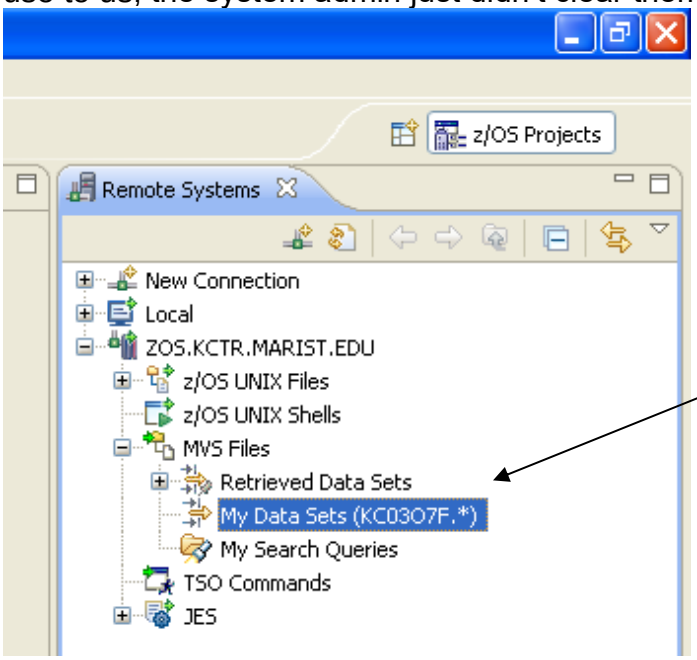
Then we will connect to the system … which you should already have done as part of a previous class.

Most old time developers used a text based mainframe development environment (called a “green screen” because the text/monitors were often green in colour). These environments were easy to use with just a keyboard, so before mice and GUI’s and IDE’s were ever created. This IBM Developer IDE provides an option to use an environment like this using a connecting option called Host Connection Emulator, which we WILL be using later in the course. See the screens/how to connect/use/login to this environment at the bottom of these course slides.

Now that we have a connection to the Mainframe TSO system setup, and you should have your KC03xxx user id setup and password that expired reset, we can go and create our PDS and COBOL source files with the IDZ client. **Right click the Marist connection** once more and choose the **Connect** option and proceed to login in with your account and new password:

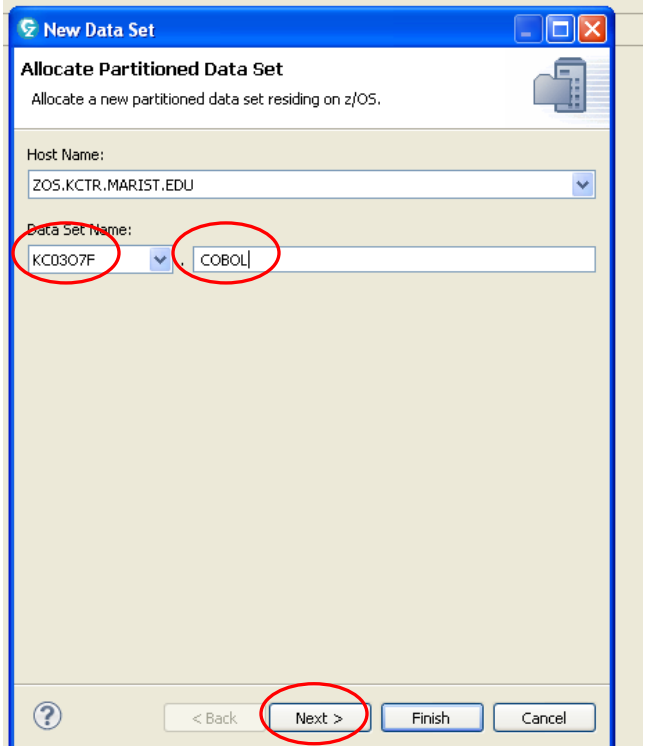


Once connected, you may see a pile of datasets, you can delete these as they were left by a previous user of this account (these were from another school so they are of no use to us, the system admin just didn't clear them out).

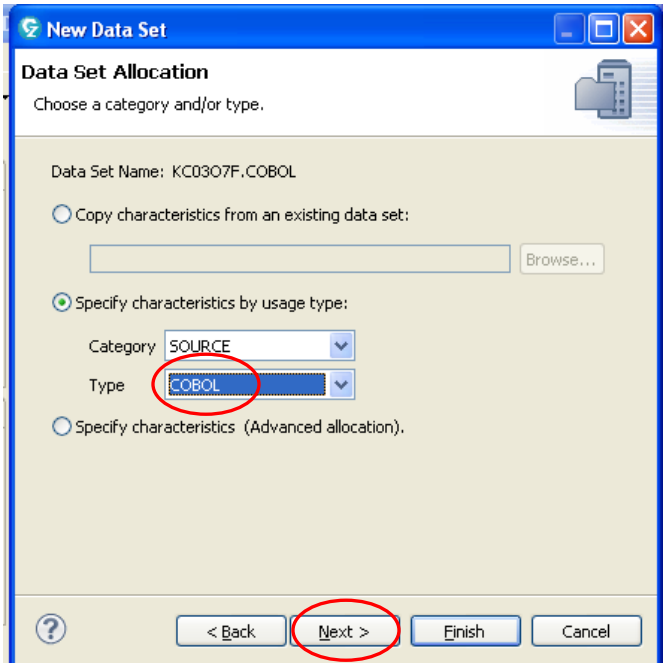


Initially clear any data sets out of here IF any extras (besides the KC03xxx.JCL) exist, otherwise skip this step

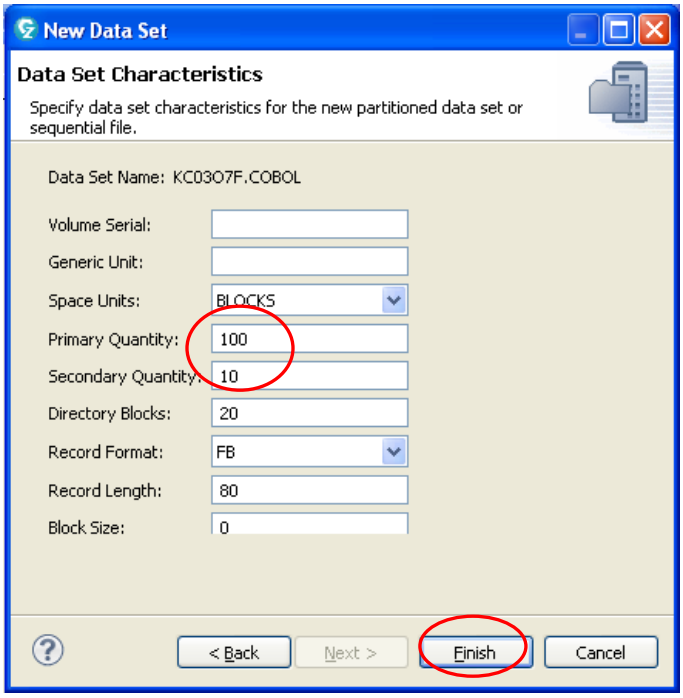
Once you have cleared all of the Data Sets, we can create a new PDS, **right click MVS files** and choose the New→Allocate **Partitioned** Data Set option with the Data Set Name of KC03xxx.COBOL (where xxx is your specific account):



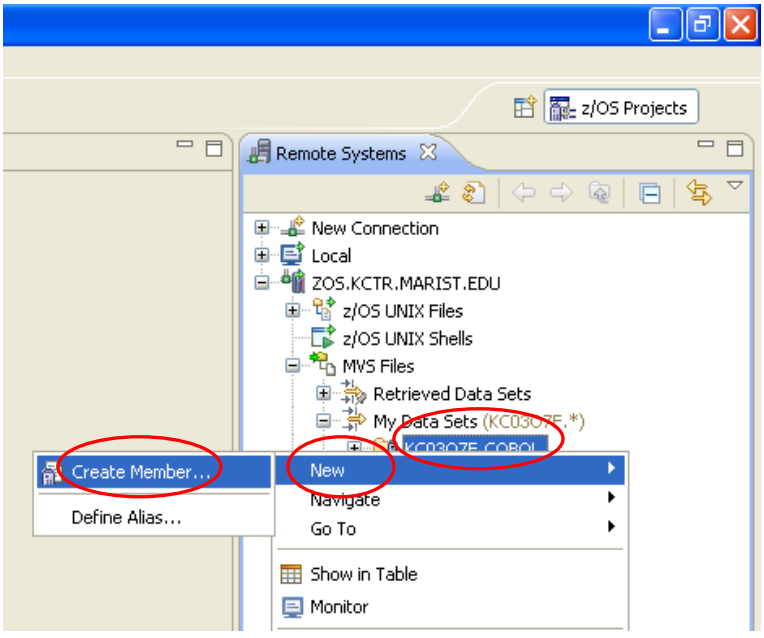
Click **Next** and change the type to **COBOL** on the next screen:



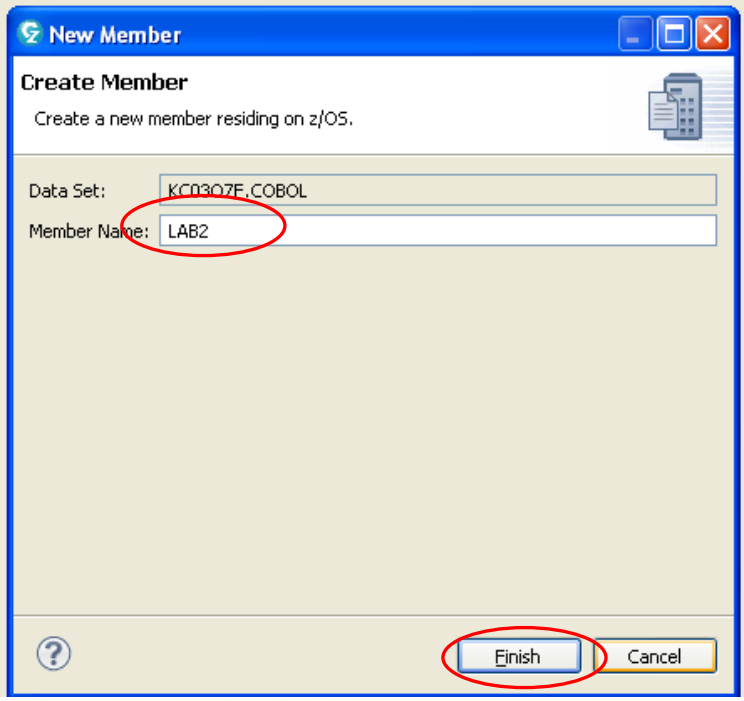
Shrink the default sizes down a bit and press Finish:



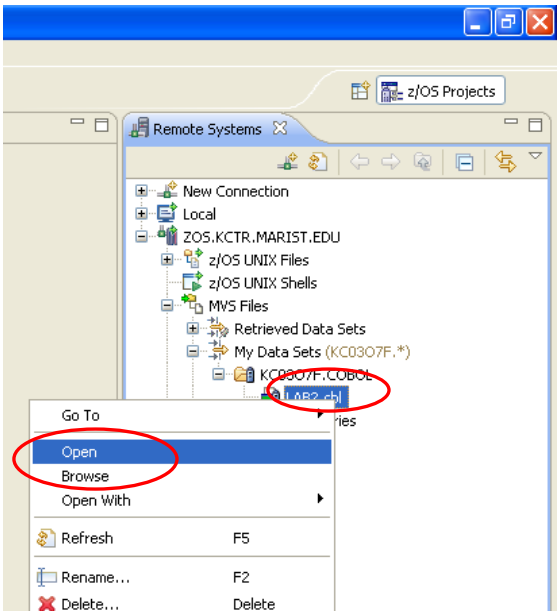
Expand the explorer out again, and right click the newly created PDS and choose the **New→Create Member** option:



We'll create a new source member called **Lab2**:

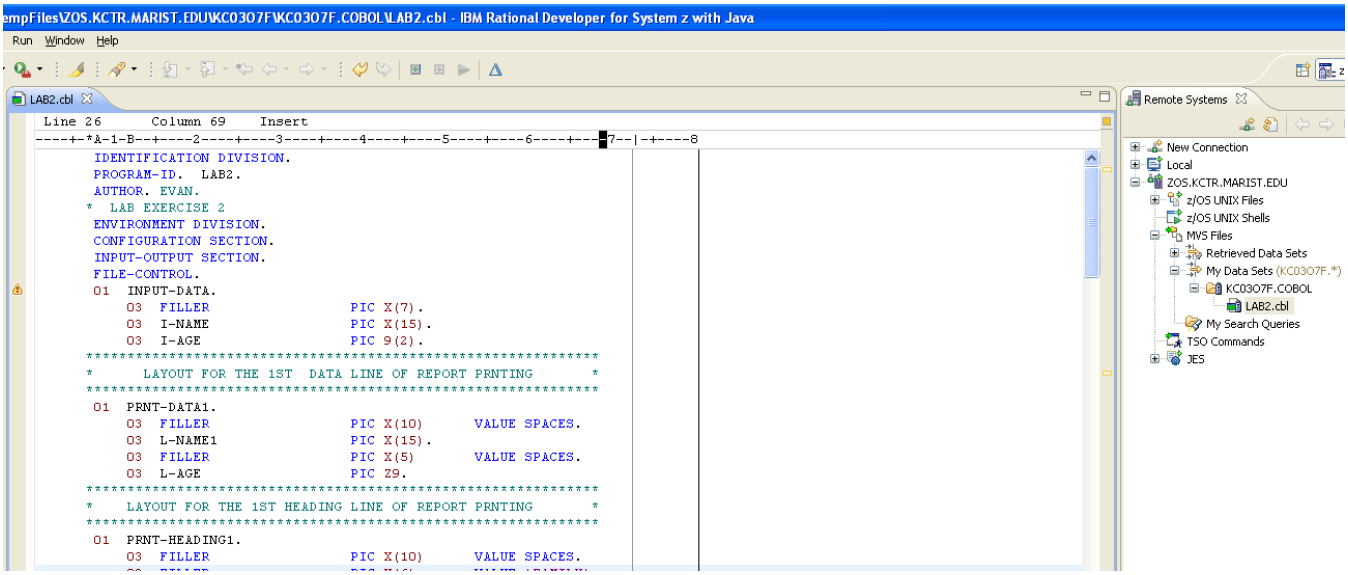


Open the newly created source member



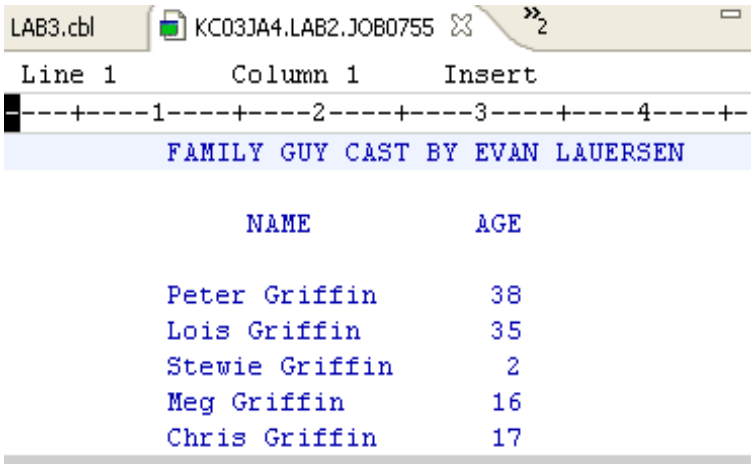
Or just double click the member.

Then copy the contents from your lab1 source file into the lab2 member on the host:



Before closing take note of the output I'd like you to eventually produce for Lab 2. It has the following modifications from the lab 1 code:

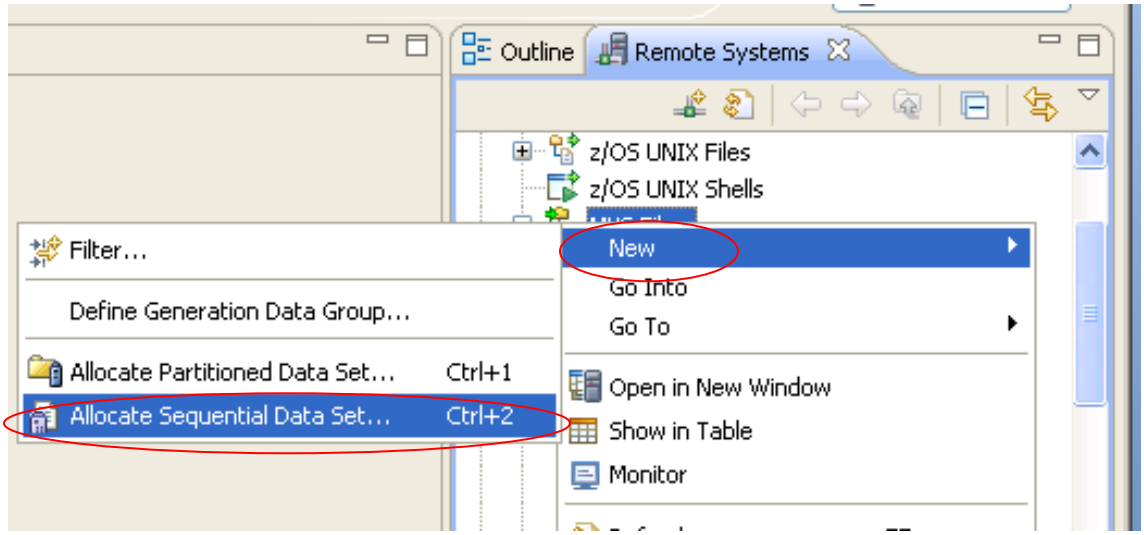
1. An extra heading line for NAME and AGE
2. An extra column in the detail lines (also, notice Stewie's age is zero suppressed)

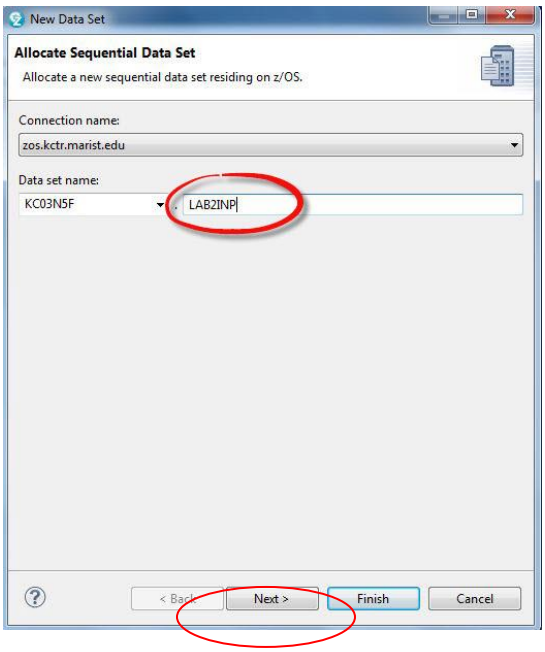


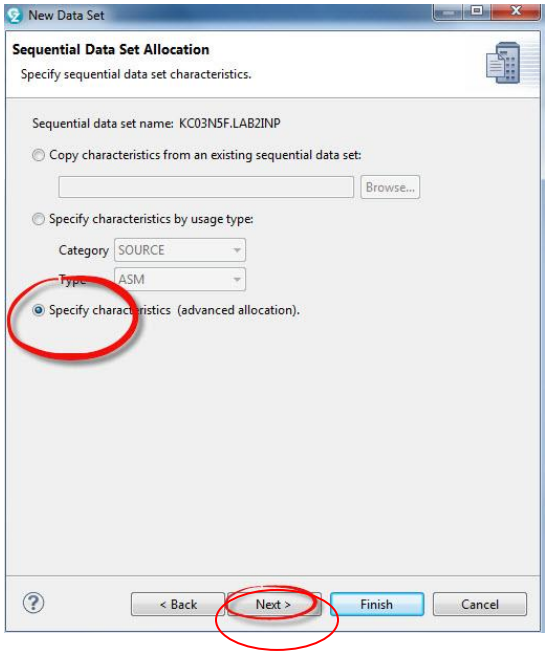
Note, you will need to **make modifications** to the program source to get this extra information to come out correctly. Study the current source and plan your changes. If you are not a Family Guy fan, feel free to change to your favourite cartoon/TV show (KimPossible, Harry Potter, Batman, etc...) , as long as the headings include your name, and the Name/Age fields & data. Note you can use the local project from last class to play around. We'll look at the actual mechanics next class.

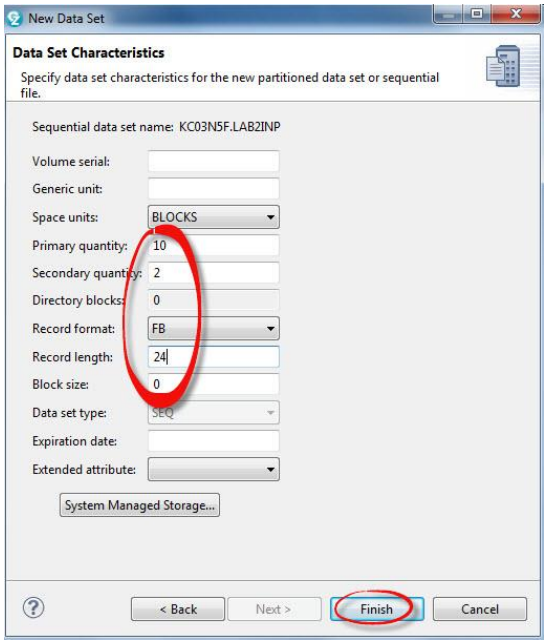
We need to add a new sequential data file to our set up called **LAB2INP.**
The mainframe has no concept of the file structure of a pc. For instance, there is no c: drive in the mainframe world. So we need to set up an input file for this program to use.

We just can't cut and paste the LAB1 data file because its record length was 22, we need this new file to have a record length of 24 to accommodate a new age field. Right click the **MVS** icon, select **New** and **Allocate <u>Sequential</u> Data Set** options.



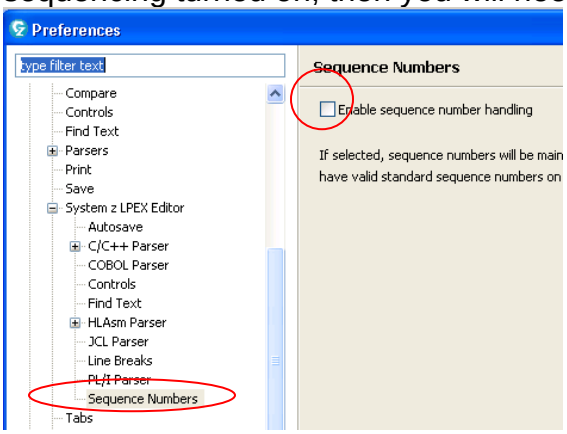




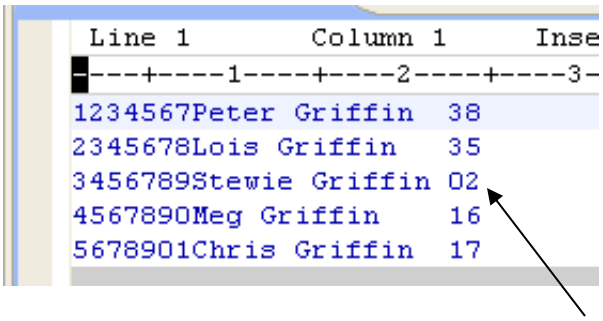


Primary=**10**, Secondary=**2**, Record Length=**24**

Locate the **LAB2INP** sequential file in the Remote Systems tab and double click on it to open it. **IF** columns 1-6 are unavailable it is probably because the editor has sequencing turned on, then you will need to go into **Preferences** and turn it off:



Then just select and copy and paste the contents of LAB2 file on FOL into the new LAB2INP sequential file:



Note the leading zero is important for Stewie Griffin, depending on how you declare and move the age field, it may cause your program to crash upon reading it.

Also, note you will need to modify the FILE CONTROL area of the code to accommodate the mainframe files.

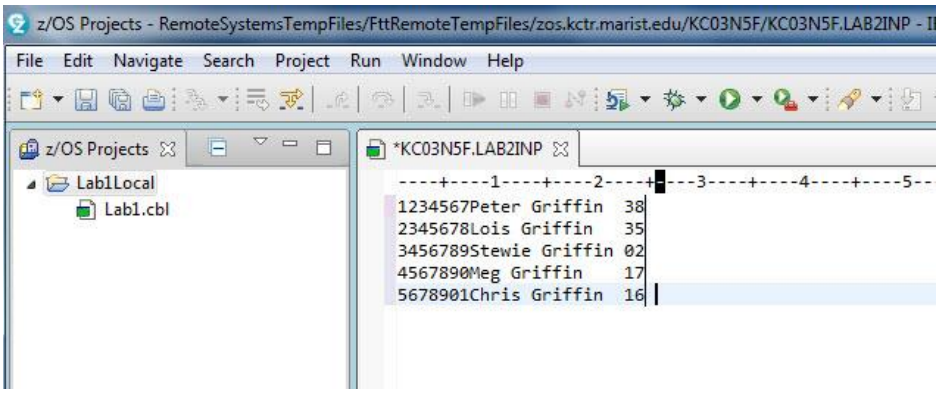
```
       AUTHOR. EVAN.
      *  LAB EXERCISE 2
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT INPUT-FILE   ASSIGN TO DA-S-INPUT
                   FILE STATUS IS IFCODE-I.
            SELECT PRNT-FILE    ASSIGN TO UR-S-PRNT.

       DATA DIVISION.
```

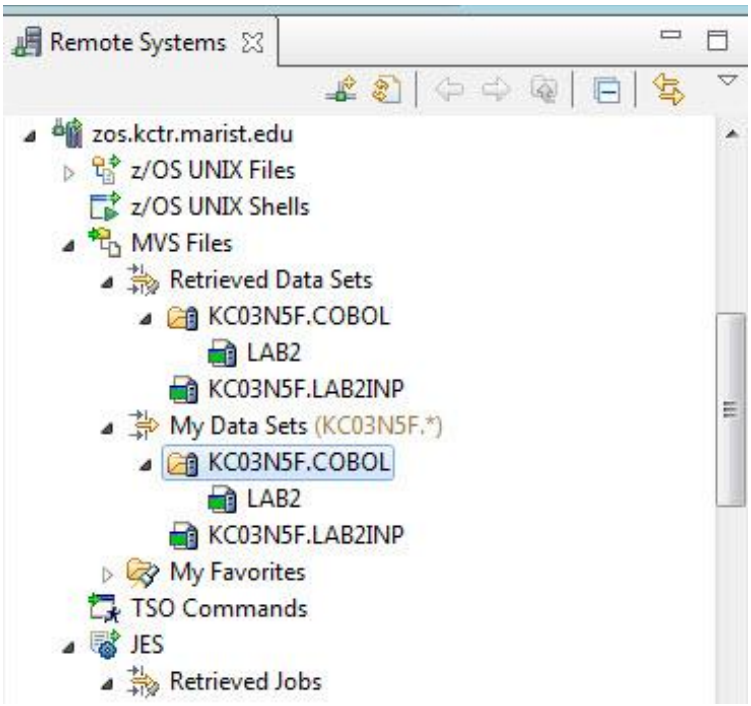
W1C2 Lab2 – 2%

- Review the mainframe overview
- Read chapter 17 of the Textbook
- Sign into Marist with Host Connection Emulator using the assigned account and change the password
- Using the IBM Developer for z/OS IDZ client
    - Clean out old data sets and files IF they still exist
    - Create a Partitioned Data Set called COBOL
    - Create a member in the COBOL PDS called LAB2 that contains COBOL code found in the local **lab1.cbl.**
        - Make modifications to the new lab2 member that will incorporate the new columns and heading.
        - Also, note below you will need to modify the FILE CONTROL area to accommodate the mainframe files.
    - Create a Sequential Data Set called Lab2INP that contains data found in the file lab2.dat on FOL
- Submit:
    - Cobol code in IDZ editor (make sure I can see the tab)
    - Data code in IDZ editor (make sure I can see the tab)
    - IDZ Remote Systems showing both files

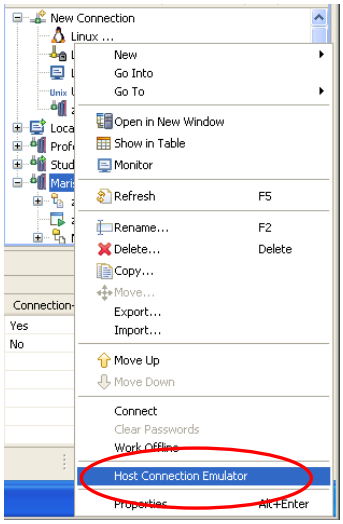
1. LAB2.cbl

```
Line 15      Column 34     Insert
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|
       IDENTIFICATION DIVISION.
       PROGRAM-ID.  LAB2
       AUTHOR. EVAN.
      *  LAB EXERCISE 2
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
            SELECT INPUT-FILE   ASSIGN TO DA-S-INPUT
                   FILE STATUS IS IFCODE-I.
            SELECT PRNT-FILE    ASSIGN TO UR-S-PRNT.

       DATA DIVISION.

       FILE SECTION.
       FD  INPUT-FILE.
       01  INPUT-REC                PIC X(22).

       FD  PRNT-FILE.
       01  PRNT-REC                 PIC X(125).
       WORKING-STORAGE SECTION.
      ****************************************************************
      *          LAYOUT FOR THE INPUT FILE                           *
      ****************************************************************
       01  INPUT-DATA.
           03  FILLER                PIC X(7).
```

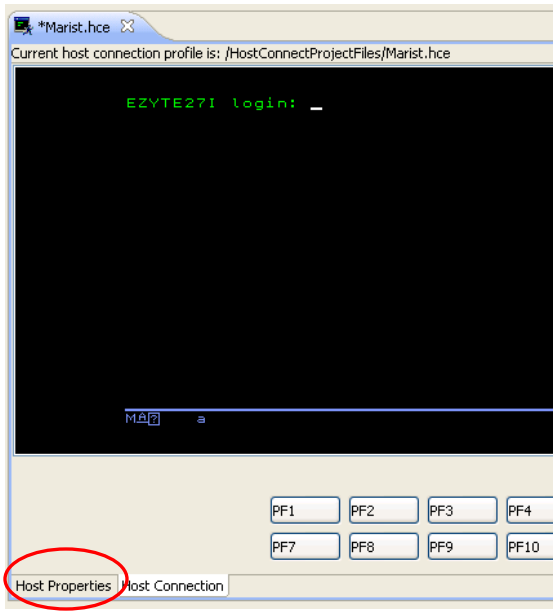
2. 

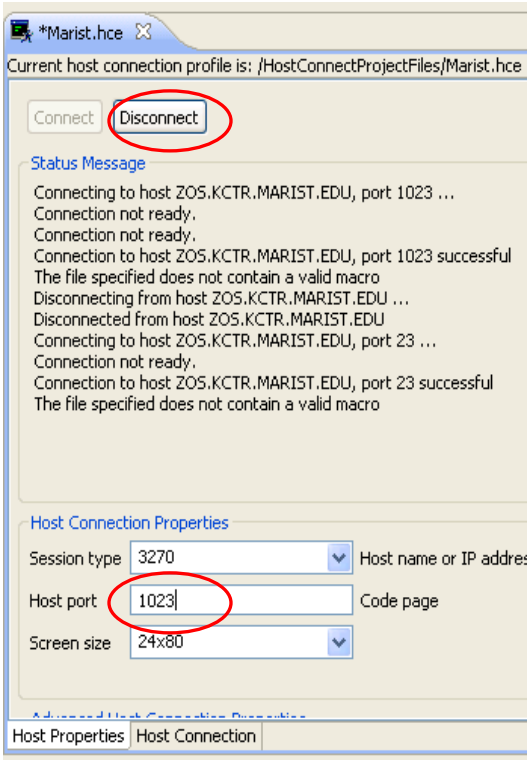
3. 

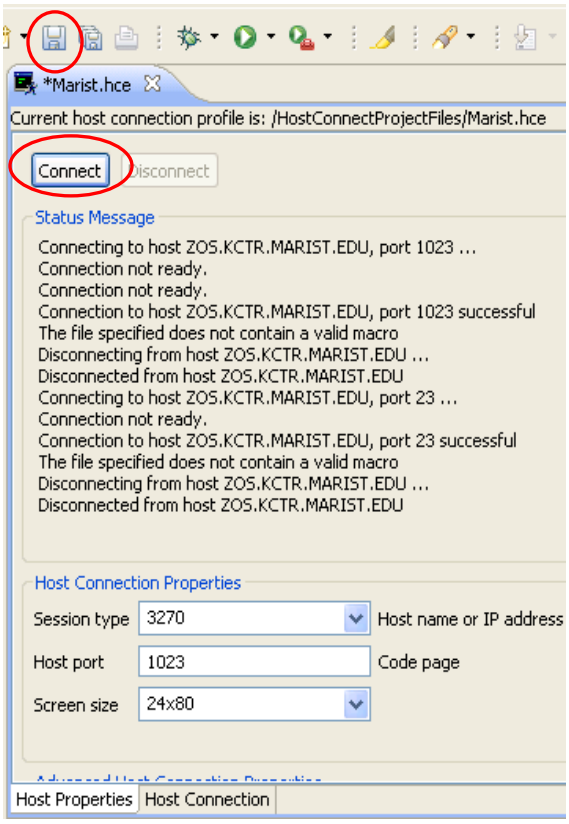
Optional / to learn how to setup/use the OLD Text based Green screen (pre mice, GUI's and IDE's) Host Connection Emulator environment, in IBM Developer, right click your Marist connection and then choose **Host Connection Emulator** option:



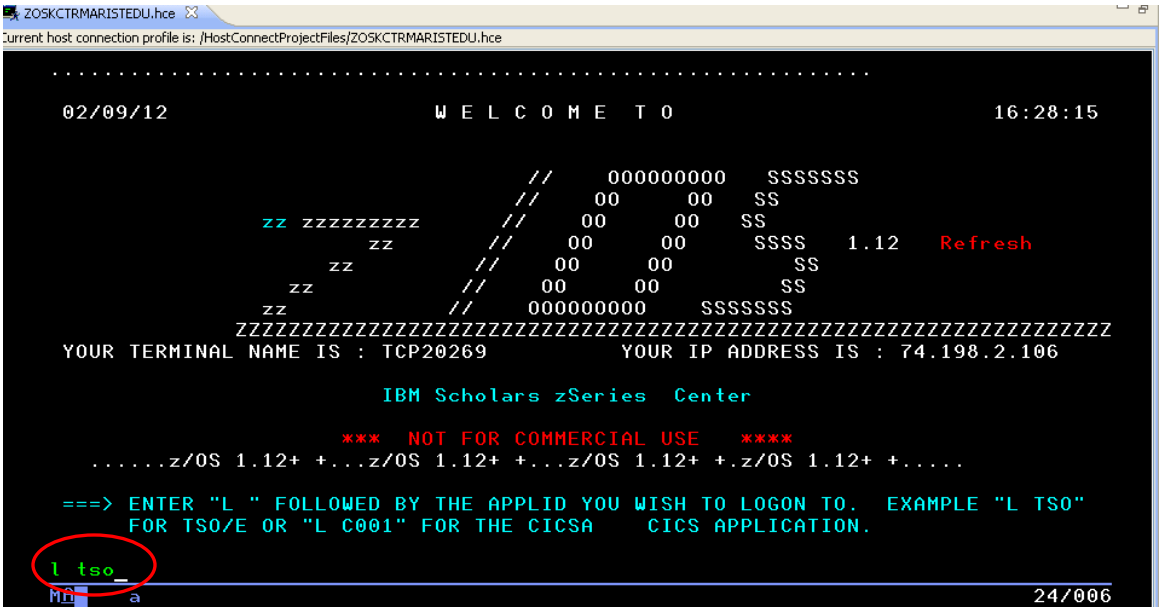
Note this will default to port 23, we actually want to use **port 1023**, so make the following changes:





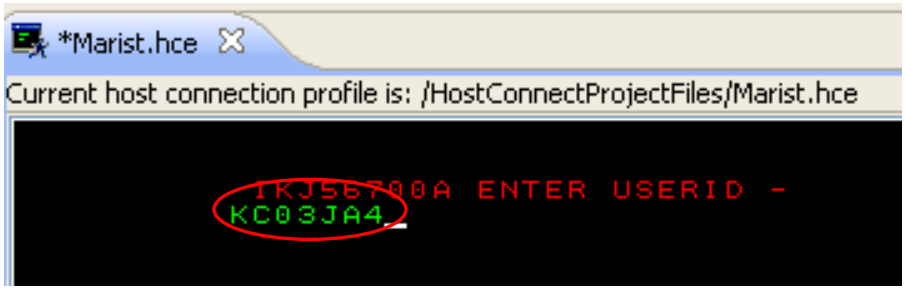


Save these settings with the save icon before connecting



Next we'll logon to **TSO** which is the ZOS interactive environment -use command **L tso**.

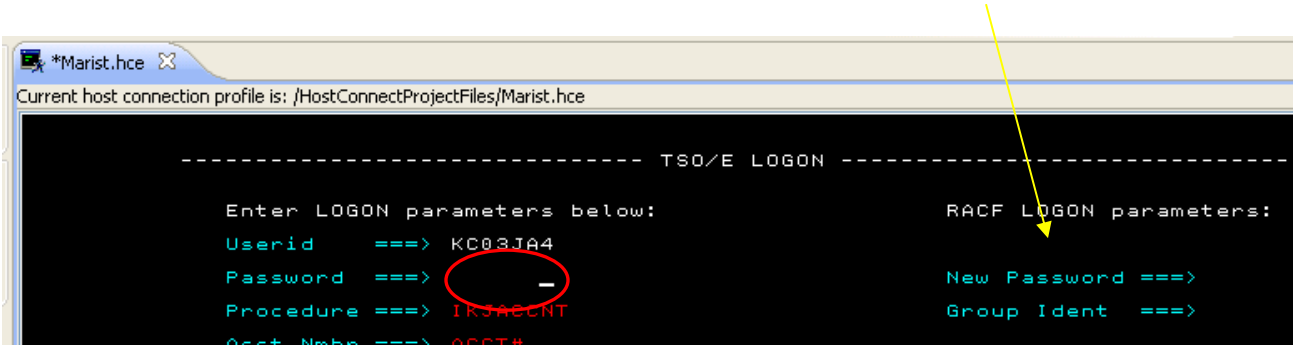
When prompted for a **userid** use the one that is **assigned to you** in class!
Note for demonstration purposes I am using a couple of accounts **KC03JA4, KC03N5F, and KC03O7F** in today's examples.



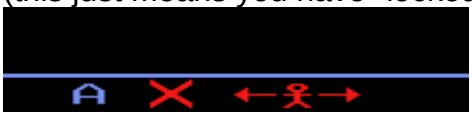
WARNING IF you have already Used / Updated your KC03xxx User ID Password ... you can just use that new password in the password field (note best not to use the mouse ... but to use the tab key ONLY if you need to change fields...

If you are using your KC03xxx User ID for the first time & changing your password here, you only will have a couple of tries at this next step and then the SYSTEM will revoke your password and lock you out of your account, so go slowly. Note the mainframe screens accept input only when you press Enter, so you need to type Your **userid** and **then press Enter** (which may be your ctrl key or the enter key on your numeric keypad) .

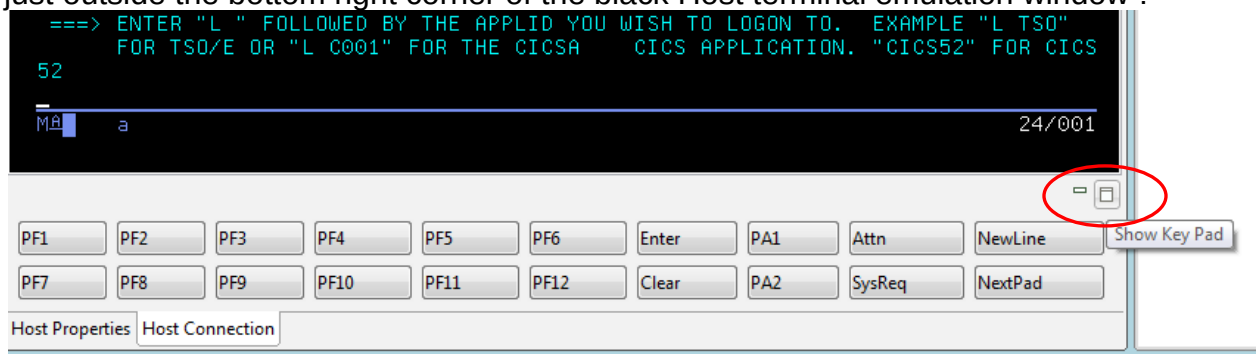
The system will prompt for your password (that I gave you).
The first time in you will be asked to change your password on the right **after** you have successfully entered the password. **Notice where the cursor is - and use TAB vs using a mouse click to get to the correct field.** Note password rules min 6, max 8 characters.



Note ==> You will need to use the TAB key to move from one input field to the next (not the mouse ...) . If you type in a place on the screen that is not an input field, you will see a capital X with a little stick person in the middle of arrows pointing left and right (this just means you have "locked up your screen / session ") .

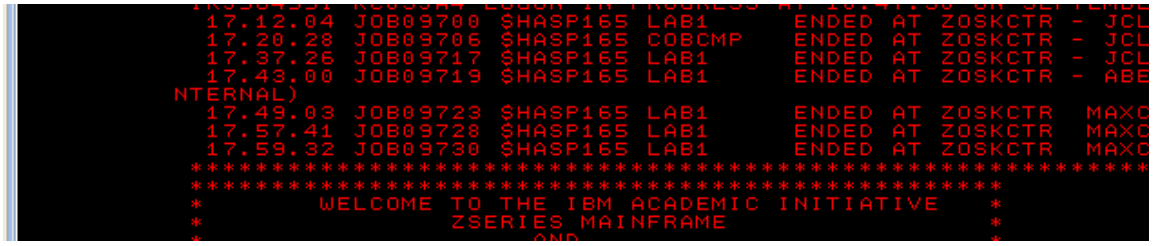


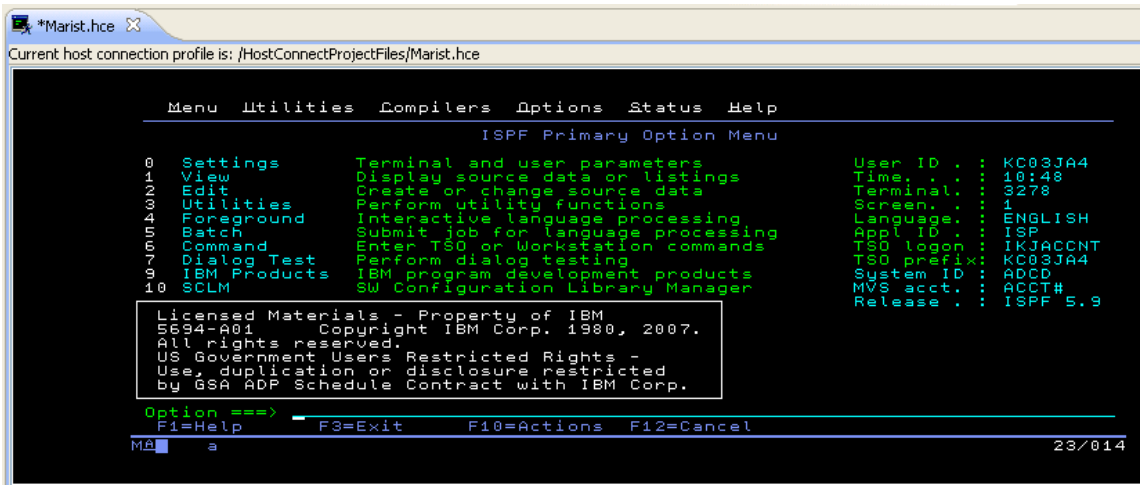
IF this happens, you may be able to clear it by pressing Esc, or you will need to click just outside the bottom right corner of the black Host terminal emulation window :



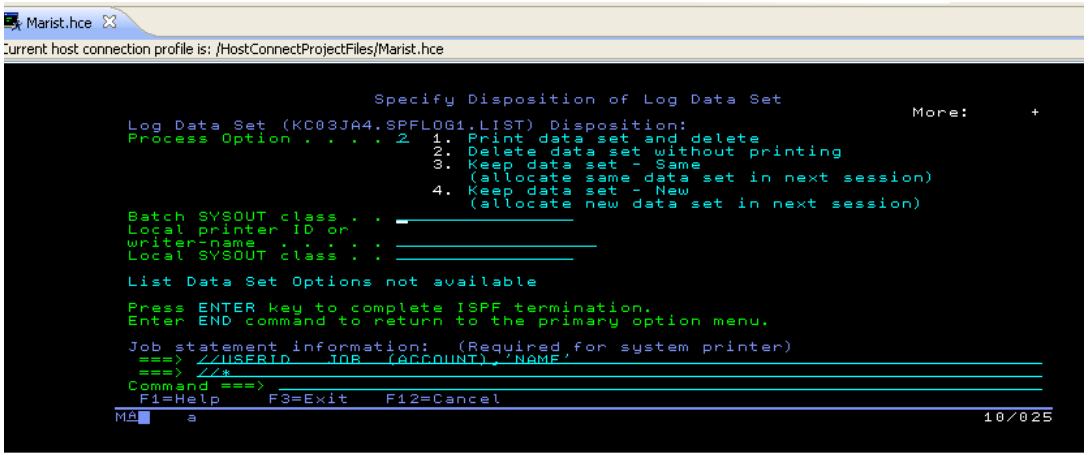
to Show Key Pad, then click on the Attn button, then Enter to allow the host to accept your keyboard input.

You will see a bunch of red screens appear, just keep hitting enter until you see the following:

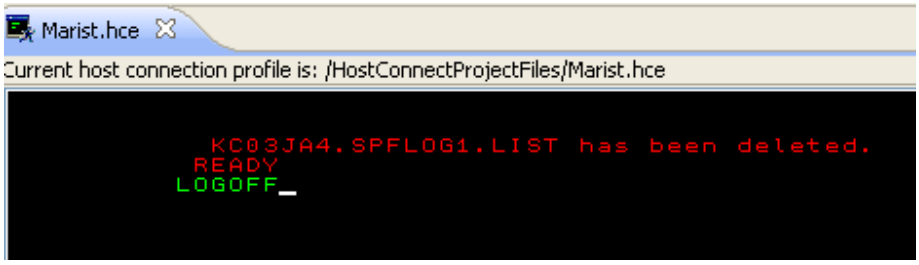




We are now logged on! Once you have your password changed press F3 (or enter an Exit command) once from the primary ISPF screen and a final screen will appear, enter a **2** to dispose of the log and leave TSO.



Then just enter LOGOFF to finally logout.



You can close the host connection emulator window safely now.