AN INTEGRATED BUSINESS SYSTEM
FOR A SMALL TUTORING SERVICE

By Jerry Parlee

Allen Bush
Herbert Haynes
David Thomas
11/21/94
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>BACKGROUND AND RATIONAL</td>
<td>2</td>
</tr>
<tr>
<td>NARRATIVE</td>
<td>4</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>6</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>8</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>22</td>
</tr>
<tr>
<td>APPENDIX A - PRIMARY TABLES</td>
<td>23</td>
</tr>
<tr>
<td>APPENDIX A.I - AUXILIARY TABLES</td>
<td>26</td>
</tr>
<tr>
<td>APPENDIX B - METHODS</td>
<td>30</td>
</tr>
<tr>
<td>APPENDIX C - E-R CHARTS</td>
<td>54</td>
</tr>
<tr>
<td>APPENDIX D - OBJECT TREES</td>
<td>57</td>
</tr>
</tbody>
</table>
ABSTRACT:
The project is an integrated business system (JIBS) that manages the information needed for a small tutoring business, Jane’s Tutoring. JIBS is a stand-alone system located in the owner’s home that creates and maintains a database of customers and tutors used to schedule appointments, bill customers, pay tutors, determine the tutors, and maintain the business' financial records.
BACKGROUND and RATIONAL:

A central problem for any small business is good record keeping. Often a harried owner puts off the detail work of writing down every aspect of running a small business, hoping she will remember as needed. This will work for a while, but sooner or later (at least for a successful business) the owner will be overwhelmed.

Jane's Tutoring Service has been in business for about three years. She has been successful in recruiting students through social contacts and has hired a few tutors to work for her. The majority of her clients need tutoring for the GED, ACT, or SAT. Recently she realized that her records were totally disorganized. They were kept in a freeform notebook - when she remembered to use it. Even though she believed that she had a thriving business, she often lost track of tutors and customers, their schedules, bills, and payroll. In addition she had no accounting system to give her an overview of the success (or failure) of her business. There were also concerns about the payment of appropriate taxes.

Jane had a number of options. She could simply discipline herself to keep better records. She could purchase generic business software that she might adapt to her needs. Or she could have a system built to her specifications. Merely keeping better records would not work for Jane: she has three children, a house to maintain, was going to night school at TAMU-K and teaching part time at Del Mar Community College. There simply wasn't time. Generic software was also problematic - she had neither the ability nor time to adapt it for her business. In addition it would not be cheap. She would need software for a scheduler, a database, and a spread sheet. It was unlikely she could
program it to the point where it would appear as an integrated system. Her third choice, having a system built, would also be moderately expensive but she would get what she wanted and, best of all, it would take the least of her time.
NARRATIVE:

In general JIBS does two things for Jane’s Tutoring Service: it prompts for needed information, and returns that information in a useful manner. JIBS accepts and maintains all pertinent information regarding students, tutors, sessions, services, schedules, and expenses. On request, it displays or provides hardcopy for student’s bills, tutor’s pay, and overall student and tutor reports. JIBS displays both today’s schedule and all scheduled sessions. JIBS also displays information about individual tutor and student schedules. In addition JIBS does monthly, quarterly and yearly accounting totals. JIBS uses Paradox for Window’s Graphical User Interface.

JIBS opens its execution by displaying today’s schedule automatically. When the schedule screen is closed the control panel is displayed. It shows an area for viewing and entering data, an area for viewing and printing output, a scheduling area, and an accounting area. The "VIEW & ENTER" area has four buttons: "STUDENT & SESSIONS", "SERVICES", "TUTORS", and "EXPENSES". Each button opens a screen that permits entering new data and viewing all old data in the relevant area. In addition each screen has searching and editing capabilities as well as an large "CLOSE" button. Each screen has a similar layout, but with noticeably different colors to prevent confusion.

The "VIEW & PRINT" area has "PAY TUTOR", "BILL STUDENTS", "TUTOR REPORT", and "STUDENT REPORT" buttons and their associated "PRINT" button. Bills, Payroll, the complete tutor report, and the complete student report are viewed or printed by pushing the appropriate button. Each report displays or prints one tutor’s or student’s current accounts or total accounts on a separate page.
The "SCHEDULES" area displays scheduled tutoring sessions. As previously stated, today's schedule is automatically displayed when JIBS is opened. In addition there is a "SCHEDULED SESSIONS" button for displaying all scheduled sessions.

The "ACCOUNTING" area prompts for a time period to display and print accounting information.

JIBS itself is loaded as a run time version of Borland's Paradox for Windows and is closed as all Windows programs are - either clicking the left mouse button on File | Exit, or double clicking on the upper left horizontal button.
ENVIRONMENT:

This Integrated Business System for Jane’s Tutoring Service uses the following hardware and software:

I. Organization: Jane’s Tutoring Service.

II. Hardware

A. Stand-alone computer
   1. IBM PC compatible: 486 DX33.
   2. 250M hard drive.

B. Epson dot matrix printer.

III. Software

A. Microsoft Windows 3.1.

B. Borland Paradox 5.0 for windows.

C. Borland ObjectPal database language.

ObjectPal is an object based, event driven, containership modeled language. It is a complete language with virtually all of the functions of more familiar programming languages such as C++ or Pascal. Paradox itself is based on objects - everything in the system is an object that has certain built in capabilities. All tables, forms, reports, queries, scripts, libraries, and design objects (boxes, circles, graphs, buttons, menus, etc) are objects. ObjectPal’s components (methods, procedures, etc.) and the messages (events) passed between objects are also objects. There are a large number of built-in method objects that have predefined actions. Each method is attached graphically to (contained by) objects in Paradox. An event is any action that takes place within Paradox...
(keystrokes, mouse movements, mouse button actions, arriving on an object, changing a value, etc.). When an event happens the root object (generally a form or report) either passes the event to it's known object for internal events or bubbles the event through the object tree for external events.
PROCEDURE:

A group of 5 core tables (STUDENT, SESSION, TUTOR, SERVICES, and EXPENSES - appendix A, I - V) hold the data for all data views, calculations, and scheduling. Another 12 tables (appendix A, I - X) are used for display purposes. The STUDENT table contains student information and has the following fields: STU_NO, STU_LAST_NAME, STU_FIRST_NAME, STREET, CITY, STATE, ZIP, PHONE, SUBJECT_1, SUBJECT_2, SUBJECT_3, and RATE. The primary key is STU_NO. STU_LAST_NAME and STU_FIRST_NAME are secondary keys used for sorting views of the table in any form or report. SUBJECT_1, SUBJECT_2, and SUBJECT_3 have a common look-up table - SUBJECT, that insures that only approved subjects are tutored and that the correct name of the subject is used.

The SESSION table contains all session information and has the following fields: STU_NO, DATE, START, TUTOR, TUTOR_FIRST_NAME, SUBJECT, END, HOURS, DISCOUNT, BILL, S_PAID_IN, BIL_BAL, TUT_FEE, TUT_ADJ, TUT_PAY, T_PAID_OUT, PAY_BAL, LOCATION, NOTES, TEST_DATE, and RESULTS. STU_NO, DATE, and START the primary keys. STU_NO and DATE are obvious keys and START (the time the session started) is better than other possible keys, such as TUTOR or SUBJECT, as the student might be tutored twice in one day by a specific tutor on a given subject. TUTOR and TUTOR_FIRST_NAME are secondary keys used to link the SESSION table to the TUTOR table and has a look-up table - TUTOR. Both primary and secondary keys have strict referential integrity. The primary key depends on there existing an equivalent key in the STUDENT table and the secondary key depends on there existing an
equivalent key in the TUTOR table. SUBJECT has a look-up table - SUBJECT, that insures that only approved subjects are tutored and that the correct name of the subject is used.

The TUTOR table contains tutor information and has the following fields: TUTOR, TUTOR_FIRST_NAM, STREET, CITY, STATE, ZIP, PHONE, SUBJECT_1, SUBJECT_2, SUBJECT_3, SUBJECT_4, T_RATE and WITHHOLDING. The primary key is TUTOR and TUTOR_FIRST_NAM. There are no secondary keys. SUBJECT_1, SUBJECT_2, SUBJECT_3, and SUBJECT_4 have a common look-up table - SUBJECT, that insures that only approved subjects are tutored and that the correct name of the subject is used.

The SERVICES table contains information on services and has the following fields: STU_NO, DATE, SERVICE, SER_COST, SER_BILL, SER_PAID, and SER_BAL. The primary key is STU_NO, DATE, and SERVICE. The primary key has strict referential integrity and depends on there existing an equivalent key in the STUDENT table. There are no secondary or look-up tables.

The EXPENSES table contains information on expenses and has the following fields: DATE, ITEM, DESCRIPTION, and COST. The primary key is DATE, ITEM, and DESCRIPTION. There are no secondary or look-up tables.

From these tables VIEWING AND ENTERING screens are generated that do the following:

1. Display current data and/or permit data entry for students and their associated sessions (STUDENTS & SESSIONS).

2. Display current data and/or permit data entry for tutors and their sessions
3. Display current data and/or permit data entry for services rendered to students (SERVICES).

4. Display current data and/or permit data entry for company expenses (EXPENSES).

The following VIEWING AND PRINTING screens are also available:

1. Display and provide hardcopy billing statements for students who currently owe for sessions received (BILL STUDENTS).

2. Display and provide hardcopy payroll statements for tutors whose sessions have been billed and paid for (PAY TUTORS).

3. Display and provide hardcopy for each student's complete record (STUDENT REPORT).

4. Display and provide hardcopy for each tutor's complete record (TUTOR REPORT).

The following SCHEDULES are available:

1. Today's schedule (automatic pop-up).

2. All future schedules (SCHEDULED SESSIONS).

Company ACCOUNTING information is also available. The user selects the date period.

On opening JIBS to the CONTROL PANEL form the ObjectPal "open" method (see appendix B, II. A) is executed. This method consists of two procedures, GetSced() and ShowSched(). GetSced() performs a query on the joined tables SESSION, STUDENT
and TUTOR. It queries the SESSIONS table for any sessions scheduled for today. If there are any, it places them in a table called SCHEDULE. If the SCHEDULE table is not empty ShowSched() opens the SCHEDULE form that displays that table. If the SCHEDULE table is empty ShowSched() displays the message "SORRY, NO APPOINTMENTS TODAY".

The CONTROL PANEL is a form with "push button" methods for opening each of the forms or reports listed above. There are 15 push buttons with their individual methods as shown in appendix B, II, B through P. Eight buttons (STUDENT'S SESSION form, STUDENT'S SERVICES form, TUTOR'S SESSION form, EXPENSES form, TUTOR report, print TUTOR report, STUDENT report, and print STUDENT report - appendix B, II, B through I) simply open the relevant forms or reports and do no processing other than maximizing the form or report object, and printing if appropriate. Seven buttons (TUTOR PAYROLL report, print TUTOR PAYROLL report, STUDENT BILL report, print STUDENT BILL report, SCHEDULE report, EXPENSES report, and print EXPENSES report - appendix B, II, I through P) query for specific conditions as explained in each form or report description below. In addition there are global variables, DATE and DATE1, used in the accounting form.

The STUDENTS & SESSIONS form displays and permits entering and editing the following data. For each student; last name, first name, address, phone number, up to three different subjects, and the rate to be billed. When a new student is entered a unique student number is generated. Student deletions are denied if the student has any sessions. If a deletion is successful the next new student receives the deleted student's
old number. For each session for that student it permits entering and editing tutor last name, tutor first name, subject, date, starting time, ending time, any discounts given, how much of the bill has been paid, notes on the session, location of the session, test date, test result. In addition, it calculates hours for this session, gross bill, net bill, bill balance, tutor fee, tutor adjustment, and tutor's pay, taking into account any discounts. There are push buttons for entering a new student, finding a student, start editing, stop editing, and closing the form. Entering new tutors and subjects is facilitated by a look-up table. There are three underlying tables for this form - STUDENT, SESSION, and, TUTOR. The joining key between STUDENT and SESSION is student number. The joining key between SESSION and TUTOR is tutor's last and first name. The CONTROL PANEL "open" method simply opens the form as no pre-selection of data is needed.

In the student section of the form there are two buttons (ADD NEW STUDENT and FIND A STUDENT) and three identical "setFocus" methods (to look up subjects). The ADD NEW STUDENT "push button" method (appendix B, III. G) first places the STUDENT table into edit mode then goes to the end of the table and appends a blank record. It then opens a "tCursor" (an ObjectPal device to move through a table) on the STUDENT table and looks for a gap in the STU_NO field. If it finds one it assigns that number to the new student STU_NO, if not it assigns the next unused number to it. The STU_NO field is attached to the form but is not visible at run time. The FIND A STUDENT "pushButton" method (appendix B, III. H) calls an action method called dataSearch that opens a SEARCH DIALOG BOX. If it fails to find the item it shows the message "Oops! either you cancelled the dialog box or I couldn't find the name you
entered" In reality the SEARCH DIALOG BOX can be set by the user to search for any field in the STUDENT table. The "setFocus" methods (setFocus()) occur when the cursor enters any SUBJECT field, appendix B, III. D, E, and F) are attached to each student subject record object. They check to see if the form is in edit mode, if it is then it key fakes the key press "control/space bar" which opens a look-up table for subjects. The user can then select a subject and automatically insert it into the field.

In the sessions section of the form there are two buttons (ENTER OR EDIT ALL and END EDIT) two "setFocus" methods (one to look up tutors the other to look up subjects) two "removeFocus" methods (one to calculate billing balances from an entry into PAID, another to calculate GROSS, BILL, TUT_FEE, TUT_ADJ, TUT_PAY, BILL_BAL, and PAY_BAL from an entry into END ) and one "canArrive" method (to prevent trying to enter a session ending time to a session that is scheduled for sometime in the future). Both ENTER OR EDIT ALL and END EDIT buttons are simple "pushButton" methods (appendix B, III. B and C) that call the methods "edit" and "endEdit" respectively. The "setFocus" methods (appendix B, III. I) check to see if the form is in edit mode, if it is then it key fakes the key press "control/space bar" which opens a lookup table for tutors and subjects. Note that the code for both tutors and subjects is identical, the fact that they are attached to either TUTOR or SUBJECT record objects determine which lookup table they call. The END "removeFocus" method (appendix B, III. I) first checks to see if the form is in edit mode, then if any time has been entered into the END field. If both conditions are met it uses the procedure DURATION() to calculate the decimal time of the session. That time is stored in the field HOURS and is used to calculate GROSS
(HOURS times RATE), student bill (HOURS times RATE less DISCOUNT), tutor fee (HOURS times tutor rate), tutor adjustment (tutor rate divided by student rate times
DISCOUNT), tutor pay (tutor fee less tutor adjustment), bill balance (student bill less student paid, if any), and tutor pay balance (tutor pay less tutor paid, if any). The tutor fields are attached to the form but are not visible at run time. The PAID "removeFocus" method (removeFocus()) occurs when the cursor leaves a field, appendix B, III. J) first checks to see if the form is in edit mode, then if any amount has been entered into the paid record. If both conditions are met it calculates the student session bill balance by converting the start and ending times into decimal numbers, calculating the difference, multiplying by the student rate, and subtracting the student discount. From that total the amount the student has paid is subtracted giving the session balance. The "canArrive" method (canArrive()) determines if conditions permit the cursor to enter a field, appendix B, III. K) for the END field checks to see if the session date is in the future. If it is, it shows the message "ERROR, THIS SESSION HAS NOT HAPPENED YET." and sets an error code "canNotArrive" to prevent the user from entering the END field. There is also a CLOSE button (appendix B, III. A) that performs an "endEdit" method then a "close" method to close the form.

The TUTORS form displays and permits entering and editing the following data. For each tutor; last name, first name, address, phone number, up to four different subjects to tutor, and the rate to be billed. For each session the following are displayed; name, date of session, subject, hours, fee, discount, pay, paid, and session notes. Paid is the only field in the sessions table that may be edited. There are push buttons for adding
new tutors, finding tutors, start editing, stop editing, and closing the form. There are three underlying tables for this form: TUTOR, SESSION, and, STUDENT. The joining key between TUTOR and SESSION is tutor's last and first name. The joining keys between SESSION and STUDENT is student number. The CONTROL PANEL "open" method simply opens the form as no pre-selection of data is needed.

In the tutor section of the form there are two buttons (ADD NEW TUTOR and FIND TUTOR) and four identical "setFocus" methods (to look up subjects). The ADD NEW TUTOR "push button" method (appendix B, IV. H) places the TUTOR table into edit mode then goes to the end of the table and appends a blank record. The FIND TUTOR "pushButton" method (appendix B, IV. G) calls an action method called dataSearch that opens a SEARCH DIALOG BOX. If it fails to find the item it shows the message "Oops! either you cancelled the dialog box or I couldn't find the name you entered" In reality the SEARCH DIALOG BOX can be set by the user to search for any field in the TUTOR table. The "setFocus" methods (appendix B, IV. C, D, E, and F) are attached to each student subject record object. They check to see if the form is in edit mode, if it is then it key fakes the key press "control/space bar" which opens a lookup table for subjects. The user can then select a subject and automatically insert it into the field.

In the sessions section of the form there are two buttons (EDIT ALL and END EDIT), one "removeFocus" method (to calculate session payroll balances from an entry into PAID), and nine "canArrive" methods (to prevent entering any session information except PAID). Both EDIT ALL and END EDIT buttons are simple "pushButton" methods (appendix B, IV. I and J) that call the methods "edit" and "endEdit" respectively. The
PAID "removeFocus" method (appendix B, IV. L) first checks to see if the form is in edit mode, then if any amount has been entered into the paid record. If both conditions are met it calculates the student session bill balance by converting the start and ending times into decimal numbers, calculating the difference, multiplying by the tutor rate, and subtracting the tutor adjustment. From that total the amount that has been paid to the tutor is subtracted giving the session balance. The session fields START, END, DISCOUNT, and RATE are used in this calculation and are therefore in the form but not visible at run time. The "canArrive" methods (appendix B, VI. K) for all session fields except PAID refuse entry and therefore editing of those fields. Those fields are either calculated fields or should be edited in the STUDENT/SESSIONS form. There is also a CLOSE button (appendix B, IV. A) that performs an "endEdit" method then a "close" method to close the form.

The SERVICES form displays and permits entering and editing the following data. For each service for a student: date, services or items purchased, cost of services to the tutoring service, charges billed to the student, and how much of the bill has been paid. For each student: last name, first name, address, and phone number are shown. There are push buttons for starting editing, stopping editing, and closing the form. There are two underlying tables for this form: STUDENT and SERVICES. The joining keys is student number. The CONTROL PANEL "open" method simply opens the form as no pre-selection of data is needed.

In the SERVICES form there are three buttons (CLOSE, EDIT, and END EDIT), and seven "canArrive" methods (to prevent entering any student information). CLOSE, EDIT,
and END EDIT buttons are simple "pushButton" methods (appendix B, V. A, B and C) that call the methods "close", "edit", and "endEdit" respectively. The "canArrive" methods (appendix B, V. D) for all student fields refuses entry and therefore editing of those fields. Those fields should be edited in the STUDENT/SESSIONS form.

The EXPENSES form displays and permits entering and editing the following data: date of incurrence, item type, description, and cost. There is only one underlying table - EXPENSES. In the EXPENSES form there are three buttons (CLOSE, EDIT, and END EDIT). CLOSE, EDIT, and END EDIT buttons are simple "pushButton" methods (appendix B, VI. A, B and C) that call the methods "close", "edit", and "endEdit" respectively. The CONTROL PANEL "open" method simply opens the form as no pre-selection of data is needed.

The STUDENT BILL report displays all current open bills and allows printing those bills. It shows the following data: Student name, student address, phone, subjects and billing rate. For each session for that student it displays: date, tutor last name, tutor first name, subject, hours, gross, discounts, actual bill, amount paid, balance. For session totals: total hours, total gross, total discount, total billed, total paid, total balance. For each service to that student it shows: date, name of service, bill, amount paid, balance due, and totals. For service totals: total bill, total paid, and total balance. It also displays the sum of session balance and service balance as a "please pay" amount. There are three underlying tables: Bil_SES, Bil_SER, and Stu2. When this form is being opened from the control panel its push button method runs the following local procedures: BilSes(), BilSer(), and Stu2(appendix B, II. L). Then it opens the report. The procedures
BiSes() and BiSer() run queries for sessions or services that have an outstanding balance. They create the tables BIL_SES and BIL_SER and index them on the fields STU_NO and DATE. Stu2() creates a table (STU2) of student information for only those students who have a balance in either sessions or services. It then indexes the table on the fields STU_NO and DATE. All calculating except certain summations were done in the STUDENT/SESSIONS form so no recalculation needed to be done in the queries. The field summations in the report use the sum() method and are contained by the respective totals fields. The field TOTAL BALANCE is a sum of all session and service balances and is a calculation using summation methods() contained by the SUM_TOTAL_VIEW object. The PRINT STUDENT BILL button is identical to the STUDENT BILL button except that it has a "print" method.

The report STUDENT REPORT is similar to BILL STUDENTS except that it does not exclude any sessions, services or students and therefore does not need to run the excluding queries. It uses three tables: STUDENT, SESSION, and SERVICES. Its method merely opens the report. It shows the following data: Student name, student address, phone, subjects and billing rate. For each session for that student it displays: date, tutor last name, tutor first name, subject, hours, gross, discounts, actual bill, amount paid, balance. For session totals: total hours, total gross, total discount, total billed, total paid, total balance. For each service to that student it shows: date, name of service, bill, amount paid, balance due, and totals. For service totals: total bill, total paid, and total balance. For overall: total billed, total paid, and total balance. The field summations in the report use the "sum" method and are contained by the respective totals fields. The
overall fields are sums of all session and service balances and are calculations using summation methods contained by the BILLED_VIEW, PAID_VIEW, and BALANCE VIEW objects. The PRINT STUDENT REPORT button is identical to the STUDENT BILL button except that it has a "print" method.

The PAY TUTOR report displays current payroll. It shows the following data: tutor name, address, phone, subjects, and pay rate. For each session for that tutor it shows: student name, date, subject, hours, fee, adjustment, net pay, paid, and balance. It also displays a calculated field, total balance. The underlying tables are PAY_TUT and TUT2. When this form is being opened from the control panel its push button method calls the procedures PAY_Q() and TUT2(), then it opens the report (appendix B, II, J). PAY_Q() includes tutors with outstanding pay, writes them to the table PAY_TUT and indexes that table on TUTOR, TUTOR_FIRST_NAM, and DATE. TUT2() selects only those tutors who have an outstanding balance and indexes that table on TUTOR, TUTOR_FIRST_NAM, and DATE. The TOTAL BALANCE field uses the "sum" method and is contained by it's object field. The PRINT PAY TUTOR button is identical to the PAY TUTOR button except that it has a "print" method (appendix B, II. K).

The report TUTOR REPORT is similar to PAY TUTOR except that it does not exclude any sessions or tutors and does not need to run excluding queries. It uses the tables: TUTOR, SESSION, and STUDENT. Its method (appendix B, II. F), merely opens the report. It shows the following data: tutor name, address, phone, subjects, and pay rate. For each session for that tutor it shows: STUDENT NAME, DATE, SUBJECT, HOURS, FEE, ADJUSTMENT, NET PAY, PAID, and BALANCE. It also displays the
TOTALS fields: HOURS, FEE, ADJUSTMENT, PAY, PAID, and BALANCE. The TOTALS fields use the "sum" method and are contained by their object fields. The PRINT PAY TUTOR button is identical to the PAY TUTOR button except that it has a "print" method.

The SCHEDULED SESSIONS form displays all future sessions. It shows the date of the session, student's name, student's phone, tutor name, tutor's phone, location, subject, start time, and notes. When this form is being opened from the control panel its push button method runs a query to select sessions that have future dates and runs the following local procedures: GetSched() and ShowSched (appendix B, II. N). It creates and uses the SCHEDULES table.

The TODAY'S SCHEDULE arrive method (appendix B, II. A) is identical to the SCHEDULED SESSIONS form except that it is called from the CONTROL PANEL's arrive method and checks for today's sessions rather than future ones.

The ACCOUNTING screen displays accounting information over any time period chosen. It shows service totals for: costs, billed to students, paid by students, billed less cost, billed less paid. Sessions totals are displayed for: billed to students, paid by students, owed to tutors, paid to tutors, billed less recieved, pay less paid out, billed less pay. An itemized list of all expenses is also displayed. Overall period totals: net expenses, net services, net sessions and net profit. When this form is being opened from the control panel its push button method (appendix B, II. O) first prompts the user for a beginning and an ending date, then runs the procedures Ses(), Exp(), and Ser() to select all services, sessions, and expenses in the appropriate time period. It uses tables SESDATE, SERDATE, and EXPDATE. The PRINT ACCOUNTING button is identical to
the ACCOUNTING button except that it has a "print" method.
BIBLIOGRAPHY:

IV. BORLAND PARADOX FOR WINDOWS USER'S GUIDE, 1992, BORLAND INTERNATIONAL Inc. 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001, USA.

V. BORLAND PARADOX FOR WINDOWS OBJECTPAL DEVELOPER'S GUIDE, 1992, BORLAND INTERNATIONAL Inc. 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001, USA.

VI. BORLAND PARADOX FOR WINDOWS OBJECTPAL REFERENCE, 1992, BORLAND INTERNATIONAL Inc. 1800 GREEN HILLS ROAD, P.O. BOX 660001, SCOTTS VALLEY, CA 95067-0001, USA.

VII. ATKINSON, LEE; HOVIS, TOM; MAGRUDER, RANDY, PARADOX FOR WINDOWS DEVELOPER'S GUIDE, CARMEL, IN, SAMS, 1993

VIII. KELLEN, VINCE; MEISNER, GAIL; RANDOLPH, DAVID; YOCK, WILLIAM; TODD, WILLIAM, PARADOX FOR WINDOWS POWER PROGRAMMING, CARMEL, IN, QUE, 1993

IX. PRATT, PHILIP, ADAMSKI, JOSEPH, DATABASE SYSTEMS MANAGEMENT AND DESIGN, BOSTON MA, BOYD & FRASER, 1991

X. PRATT, PHILIP, A GUIDE TO SQL, BOSTON MA, BOYD & FRASER, 1991

XI. SALDARINI, ROBERT, ANALYSIS AND DESIGN OF BUSINESS INFORMATION SYSTEMS, N.Y., N.Y. MACMILLIAN, 1989

XII. COURSES

A. CS 535, DATA BASE MANAGEMENT SYSTEMS, Dr. HERBERT HAYNES, FALL 1991.

B. CS 550, ADVANCED TOPICS IN DATA BASE MANAGEMENT SYSTEMS, Dr. HERBERT HAYNES, SPRING 1993.

C. CS 538, SYSTEMS ANALYSIS AND DESIGN, Mr. ALLEN BUSH, FALL 1992.
## I. STUDENT - TABLE

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
<th>DICTIONARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>STU_NO</td>
<td>N</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>student number</td>
</tr>
<tr>
<td>STU_LAST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
<td>student last name</td>
</tr>
<tr>
<td>STU_FIRST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
<td>student first name</td>
</tr>
<tr>
<td>STREET</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP</td>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_1</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutored subject</td>
</tr>
<tr>
<td>SUBJECT_2</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutored subject</td>
</tr>
<tr>
<td>SUBJECT_3</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutored subject</td>
</tr>
<tr>
<td>RATE</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>student billing rate</td>
</tr>
<tr>
<td>RECORD NAME</td>
<td>TYPE</td>
<td>SIZE</td>
<td>KEY</td>
<td>LOOKUP</td>
<td>DICTIONARY</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>------</td>
<td>------------</td>
<td>----------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>STU_NO</td>
<td>N</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>student number</td>
</tr>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>session date</td>
</tr>
<tr>
<td>START</td>
<td>T</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>session start time</td>
</tr>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>13</td>
<td>SECONDARY</td>
<td>TUTOR</td>
<td>session tutor last name</td>
</tr>
<tr>
<td>TUTOR_FIRST_NAM</td>
<td>A</td>
<td>12</td>
<td>SECONDARY</td>
<td>TUTOR</td>
<td>session tutor first name</td>
</tr>
<tr>
<td>SUBJECT</td>
<td>A</td>
<td>10</td>
<td></td>
<td>SUBJECT</td>
<td>session subject</td>
</tr>
<tr>
<td>END</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td>session end time</td>
</tr>
<tr>
<td>HOURS</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td>session length</td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>absolute session discount</td>
</tr>
<tr>
<td>BILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>session bill</td>
</tr>
<tr>
<td>S_PAID_IN</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>amount student has paid</td>
</tr>
<tr>
<td>BIL_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>session bill balance</td>
</tr>
<tr>
<td>TUT_FEE</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>session gross tutor fee</td>
</tr>
<tr>
<td>TUT_ADJ</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>absolute session adjustment</td>
</tr>
<tr>
<td>TUT_PAY</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>amount to pay tutor</td>
</tr>
<tr>
<td>T_PAID_OUT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>amount paid tutor</td>
</tr>
<tr>
<td>PAY_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>session pay balance</td>
</tr>
<tr>
<td>LOCATION</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
<td>where held</td>
</tr>
<tr>
<td>NOTES</td>
<td>M</td>
<td>80</td>
<td></td>
<td></td>
<td>comments</td>
</tr>
<tr>
<td>TEST_DATE</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td>date of test</td>
</tr>
<tr>
<td>RESULTS</td>
<td>A</td>
<td>6</td>
<td></td>
<td></td>
<td>student test score</td>
</tr>
</tbody>
</table>
### III. TUTOR - TABLE

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
<th>DICTIONARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
<td>tutor last name</td>
</tr>
<tr>
<td>TUTOR_FIRST_NAM</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
<td>tutor first name</td>
</tr>
<tr>
<td>STREET</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP</td>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_1</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutorable subject</td>
</tr>
<tr>
<td>SUBJECT_2</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutorable subject</td>
</tr>
<tr>
<td>SUBJECT_3</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutorable subject</td>
</tr>
<tr>
<td>SUBJECT_4</td>
<td>A</td>
<td>10</td>
<td>SUBJECT</td>
<td></td>
<td>tutorable subject</td>
</tr>
<tr>
<td>T_RATE</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>tutor pay rate</td>
</tr>
<tr>
<td>WITHHOLDING</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>withholding amount</td>
</tr>
</tbody>
</table>

### IV. SERVICES - TABLE

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
<th>DICTIONARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>STU_NO</td>
<td>N</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>student number</td>
</tr>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>date service incurred</td>
</tr>
<tr>
<td>SERVICE</td>
<td>A</td>
<td>20</td>
<td>PRIMARY</td>
<td></td>
<td>service name</td>
</tr>
<tr>
<td>SER_COST</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>service cost to Jane</td>
</tr>
<tr>
<td>SER_BILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>service bill to student</td>
</tr>
<tr>
<td>SER_PAID</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>service bill paid by student</td>
</tr>
<tr>
<td>SER_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>service bill balance</td>
</tr>
</tbody>
</table>

### V. EXPENSES - TABLE

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
<th>DICTIONARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
<td>date expense incurred</td>
</tr>
<tr>
<td>ITEM</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
<td>name of expense</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>A</td>
<td>20</td>
<td>PRIMARY</td>
<td></td>
<td>description of expense</td>
</tr>
<tr>
<td>COST</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
<td>cost of expense</td>
</tr>
</tbody>
</table>
APPENDIX A.1
AUXILIARY TABLES TO JIBS

1. SCHEDULE - This table is used to hold either today's session or upcoming sessions depending on when it is called.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUTOR_FIRST_NAME</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOCATION</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST_DATE</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTES</td>
<td>M</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STU_LAST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STU_FIRST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE_1</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

II. BIL_SER - Part of the student billing system. Records are selected if there is a balance due.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>STU_NO</td>
<td>$</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>SERVICE</td>
<td>A</td>
<td>20</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>SER_COST</td>
<td>$</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>SERBILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SER_PAID</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SER_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
III. BIL_SES - Part of the student billing system. Records are selected if there is a balance due.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>STU_NO</td>
<td>N</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>SUBJECT</td>
<td>A</td>
<td>10</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>HOURS</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_PAID_IN</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BILL_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUTOR_FIRST_NAM</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IV. STU2 - TABLE - Part of the student billing system. Records are selected if there is a balance due in either services or sessions.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>STU_NO</td>
<td>N</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>STU_LAST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STU_FIRST_NAME</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STREET</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP</td>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_1</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_2</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_3</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RATE</td>
<td></td>
<td></td>
<td></td>
<td>$</td>
</tr>
</tbody>
</table>
V. PAY_TUT - Part of the payroll system. Records are selected if there is a balance due.

<table>
<thead>
<tr>
<th>RECORD_NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>13</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>TUTOR_FIRST_NAM</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>SUBJECT</td>
<td>A</td>
<td>10</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>HOURS</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUT_FEE</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUT_ADJ</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUT_PAY</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_PAID_OUT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAY_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STU_LAST_NAM</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STU_FIRST_NAM</td>
<td>A</td>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VI. TUT2 - TABLE - Part of the payroll system. Records are selected if there is a balance due in PAY_TUT.

<table>
<thead>
<tr>
<th>RECORD_NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>TUTOR</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>TUTOR_FIRST_NAM</td>
<td>A</td>
<td>12</td>
<td>PRIMARY</td>
<td></td>
</tr>
<tr>
<td>STREET</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CITY</td>
<td>A</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATE</td>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP</td>
<td>A</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHONE</td>
<td>A</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_1</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_2</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_3</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBJECT_4</td>
<td>A</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_RATE</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
VII. EXPDATE - Part of the accounting system. Records are selected if they fall in selected dates.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITEM</td>
<td>A</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>A</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COST</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VIII. SERDATE - Part of the accounting system. Records are selected if they fall in selected dates.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SER_COST</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SER_BILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SER_PAID</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SER_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IX. SESDATE - Part of the accounting system. Records are selected if they fall in selected dates.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BILL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_PAID_IN</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIL_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUT_PAY</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_PAID_OUT</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAY_BAL</td>
<td>$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FROM</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TO</td>
<td>D</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X. SUBJECT - Look-up table for subjects.

<table>
<thead>
<tr>
<th>RECORD NAME</th>
<th>TYPE</th>
<th>SIZE</th>
<th>KEY</th>
<th>LOOKUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT</td>
<td>A</td>
<td>10</td>
<td>PRIMARY</td>
<td></td>
</tr>
</tbody>
</table>
I. CONTROL PANEL - GLOBAL VARIABLES
   A. Used in multiple ACCOUNT REPORT queries
   Var
      DATE, DATE1 anytype
   endVar

II. CONTROL PANEL - LOCAL PROCEDURES
   A. Open today's schedule
   proc GetSced()
      ; Queries for today's schedule
      var
      QB query
      endvar
      QB =query
      SESSION.DB | STU_NO| DATE | TUTOR | TUTOR_LAST_NAME | START |
      | _EG01 | Check | TODAY | Check | Check | Check |
      SESSION.DB | SUBJECT | LOCATION | TEST_DATE | NOTES |
      | Check | Check | Check | Check |
      STUDENT.DB | STU_NO | STU_LAST_NAME | STU_FIRST_NAME | PHONE |
      | _EG01 | Check | Check | Check |
      TUTOR.DB | TUTOR | TUTOR_LAST_NAME | PHONE |
      | _EG03 | Check | Check |
      EndQuery
      if not executeQBE(QB, "SCHEDULE.db") then
      errorShow()
      endif
      endproc

   proc ShowSched()
      ; Opens today's SCHEDULE FORM
      var
      TBVR Table
      FRM Form
      endvar
      if isTable("SCHEDULE.db") then
      TBVR.attach("SCHEDULE.db")
      if not TBVR.isEmpty() then
FRM.OPEN("SCHEDULE.fsl")
    FRM.maximize()
    FRM.WAIT()
    else
        msgStop("SORRY", "NO APPOINTMENTS TODAY")
    endif
else
    msgStop("STOP", "CANT FIND TODAY'S SCHEDULE")
endif
endproc

;===========================================================================

method open(var eventInfo Event)
GetSced()
ShowSched()
endmethod

B. Opens STUDENT'S SESSION form

method pushButton(var eventInfo Event)
    var
        TU form
    endvar
    TU.open("STU_SESS.FSL")
    TU.maximize()
endmethod

C. Opens STUDENT'S SERVICES form

method pushButton(var eventInfo Event)
    var
        TU form
    endvar
    TU.open("STU_SER.FSL")
    TU.maximize()
endmethod

D. Opens TUTOR'S SESSION form

method pushButton(var eventInfo Event)
    var
E. Opens EXPENSES form

method pushButton(var eventInfo Event)
   var
      TU form
   endvar
   TU.open("EXPENSES.FSL")
   TU.maximize()
endmethod

F. Opens TUTOR report

method pushButton(var eventInfo Event)
   var
      RU report
   endvar
   RU.open("TOT_PAY.RSL")
   RU.maximize()
endmethod

G. Opens and prints TUTOR report

method pushButton(var eventInfo Event)
   var
      RU report
   endvar
   RU.open("TOT_PAY.RSL")
   RU.print()
   RU.close()
endmethod

H. Opens STUDENT report

method pushButton(var eventInfo Event)
   var
      RU report
   endvar
RU.open("TOT_BIL.RSL")
RU.maximize()
endmethod

I. Opens and prints STUDENT report

method pushButton(var eventinfo Event)
var
   RU report
endvar

RU.open("TOT_BIL.RSL")
RU.print()
RU.close()
endmethod

J. Open TUTOR PAYROLL report

proc PAY_Q()
;Query to select only those tutors who have a balance
;Index it
var
   QB query
       TBLVAR Table
endvar

QB = Query

TUTOR.DB | TUTOR | TUTOR_FIRST_NAME |
         | Check _join1 | Check _join2 |
SESSION.DB | STU_NO | DATE | TUTOR | TUTOR_FIRST_NAME | SUBJECT | HOURS |
           | _join3 | Check | _join1 | _join2 | Check | Check |
SESSION.DB | TUT_FEE | TUT_ADJ | TUT_PAY | T_PAID_OUT | PAY_BAL |
           | Check | Check | Check | Check | Check < -0.04 OR > 0.04, NOT BLANK |
STUDENT.DB | STU_NO | STU_LAST_NAME | STU_FIRST_NAME |
            | _join3 | Check | Check |

EndQuery

if not executeQBE(QB, "PAY_TUT.db") then
   errorShow()
endif

if isTable("PAY_TUT.db") then
   TBLVAR.attach("PAY_TUT.db")
   index TBLVAR
   primary
on "TUTOR", "TUTOR_FIRST_NAM", "DATE"
endIndex
else
    msgStop("STOP", "CAN'T FIND PAY TABLE.")
endif
endproc

;==============================================================================

proc TUT2()
; Select tutors with a pay balance.
var
    QB query
    TBLVAR Table
endvar
QB = Query
TUTOR.DB | TUTOR | TUTOR_FIRST_NAM | STREET | CITY  | STATE | ZIP |
        | Check | Check_join1 | Check_join2 | Check | Check | Check | Check |
TUTOR.DB | PHONE | SUBJECT_1 | SUBJECT_2 | SUBJECT_3 | SUBJECT_4 | T_RATE |
        | Check | Check | Check | Check | Check | Check | Check |
TUTOR.DB | WITHHOLDING |
        | Check |
PAY_TUT.DB | TUTOR | TUTOR_FIRST_NAM |
            | _join1 | _join2 |
EndQuery
if not executeQBE(QB, "TUT2.db") then
    errorShow()
endif
if isTable("TUT2.db") then
    TBLVAR.attach("TUT2.db")
    index TBLVAR
    primary
    on "TUTOR", "TUTOR_FIRST_NAM", "DATE"
endIndex
else
    msgStop("STOP", "CAN'T FIND PAY TABLE.")
endif
endproc
method pushButton(var eventInfo Event)
    var
        TU REPORT
    endvar
    PAY_Q()
    TUTZ()
    TU.OPEN("FI_PAY.RSL")
    TU.maximize()
else
    msgStop("STOP", "CAN'T FIND PAY TABLE.")
endif
endproc

;====================================================================================================

method pushButton(var eventInfo Event)
    var
        TU REPORT
    endvar

    PAY_Q()
    TU.OPEN("FI_PAY.RSL")
    TU.print()
    TU.close()
endmethod

L. Open STUDENT BILL report

proc BilSes()
    ;Query to select students with a session balance
    ;Index it it

    var
        QB query
        TBLVAR Table
    endvar

    QB = query

    SESSION.DB | STU_NO | DATE | TUTOR | TUTOR_FIRST_NAME | SUBJECT | HOURS |
    | Check | Check | _join1 | _join2 | Check | Check |

    SESSION.DB | GROSS | DISCOUNT | BILL | S_PAID_IN | BIL_BAL |
    | Check | Check | Check | Check | Check < 0.04 OR > 0.04, NOT BLANK |

    TUTOR.DB | TUTOR | TUTOR_FIRST_NAME |
    | Check _join1 | Check _join2 |

EndQuery

if not executeQBE(QB, "BIL_SES.db") then
    errorShow()
endif

if isTable("BIL_SES") then
    TBLVAR.attach("BIL_SES")

36
index TBLVAR
primary
on "STU_NO", "DATE"
endIndex
else
    msgStop("STOP", "CAN'T FIND BILLABLE SESSIONS.")
endif
endproc

;=================================================================

proc BilSer()  
;Query to select students with a service balance
;Index it

var
    QB query
    TBLVAR Table
endvar

QB = query

SERVICES.DB | STU_NO | DATE | SERVICE | SER_COST |
            |      |      |         | Check |
| SERVICES.DB | SER_BILL | SER_PAID | SER_BAL |
            |        |         | Check < -0.04 OR > 0.04, NOT BLANK |

EndQuery

if not executeQBE(QB, "BIL_SER.db") then
    errorShow()
endif

if isTable("BIL_SER") then
    TBLVAR.attach("BIL_SER")
    index TBLVAR
    primary
    on "STU_NO", "DATE"
endIndex
else
    msgStop("STOP", "CAN'T FIND BILLABLE SERVICES.")
endif
endproc

;=================================================================

proc Stu2()
var QB query TBLVAR Table endvar

QB = query

BIL_SER.DB | STU_NO |
           | _join1 |
STUDENT.DB | STU_NO   | STU_LAST_NAM | STU_FIRST_NAM | STREET | CITY |
           | Check _join1 | Check | Check | Check | Check |
           | Check _join3 | Check | Check | Check | Check |
STUDENT.DB | STATE    | ZIP | PHONE | SUBJECT_1 | SUBJECT_2 | SUBJECT_3 |
           | Check | Check | Check | Check | Check | Check |
           | Check | Check | Check | Check | Check | Check |
STUDENT.DB | RATE |
           | Check |
           | Check |
BIL SES.DB | STU_NO |
           | _join3 |

EndQuery

if not executeQBE(QB, "STU2.db") then
  errorShow()
endif

if isTable("STU2") then
  TBLVAR.attach("STU2")
  index TBLVAR primary on "STU_NO", "DATE"
  endIndex

  sort TBLVAR on "STU_LAST_NAM", "STU_FIRST_NAM"
  endSort
else
  msgStop("STOP", "CAN'T FIND BILLABLE STUDENTS.")
endif
endproc
method pushButton(var eventInfo Event)
var
   TU REPORT
endvar

BillSes()
BillSer()
Stu2()

TU.open("FL_BIL.RSL")
TU.maximize()
endmethod

M. Open and print STUDENT BILL report

proc BillSes()
; Query to select students with a session balance
; Index it

var
   QB query
   TBLVAR Table
endvar

QB = query

| SESSION_DB | STU_NO | DATE | TUTOR | TUTOR_FIRST_NAME | SUBJECT | HOURS |
|           | Check  | Check| _join1 | _join2          | Check   | Check |
| SESSION_DB | GROSS  | DISCOUNT | BILL | S_PAID_IN | BIL_BAL |
|           | Check  | Check  | Check | Check    | Check < -0.04 OR > 0.04, NOT BLANK |
| TUTOR_DB   | TUTOR | TUTOR_FIRST_NAME |
|           | Check _join1 | Check _join2 |
EndQuery

if not executeQBE(QB, "BIL_SES.db") then
   errorShow()
endif

if isTable("BIL_SES") then
   TBLVAR.attach("BIL_SES")
   index TBLVAR
   primary
   on "STU_NO", "DATE"
   endindex
else
   msgStop("STOP", "CAN'T FIND BILLABLE SESSIONS.")
endif

39
endproc

;===================================================================

proc BilSer()
;Query to select students with a service balance
;Index it

var
  QB query
tblvar Table
endvar

QB = query

SERVICES.DB | STU_NO | DATE  | SERVICE | SER_COST |
            | Check | Check | Check  | Check   |

SERVICES.DB | SER_BILL | SER_PAID | SER_BAL |
            | Check   | Check   | Check < -0.04 OR > 0.04, NOT BLANK |

EndQuery

if not executeQBE(QB, "BIL_SER.db") then
  errorShow()
endif

if isTable("BIL_SER") then
  tblvar.attach("BIL_SER")
  index tblvar
  primary
  on "STU_NO", "DATE"
endIndex
else
  msgStop("STOP", "CAN'T FIND BILLABLE SERVICES.")
endif

endproc

;===================================================================

method pushButton(var eventInfo Event)

var
  TU REPORT
endvar

BilSes()
BilSer()

TU.open("FL_BIL.RSL")
N. Open SCHEDULE report

proc GetSched()
; Queries for sessions today or later

var
    QB query
endvar

QB = Query

<table>
<thead>
<tr>
<th>SESSION.DB</th>
<th>STU_NO</th>
<th>DATE</th>
<th>TUTOR</th>
<th>TUTOR_LAST_NAME</th>
<th>START</th>
</tr>
</thead>
<tbody>
<tr>
<td>_EG01</td>
<td>Check &gt;= TODAY</td>
<td>Check _EG03</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SESSION.DB</th>
<th>SUBJECT</th>
<th>LOCATION</th>
<th>TEST_DATE</th>
<th>NOTES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Check</td>
<td>Check</td>
<td>Check</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUDENT.DB</th>
<th>STU_NO</th>
<th>STU_LAST_NAME</th>
<th>STU_FIRST_NAME</th>
<th>PHONE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_EG01</td>
<td>Check</td>
<td>Check</td>
<td>Check</td>
<td>Check</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TUTOR.DB</th>
<th>TUTOR</th>
<th>TUTOR_LAST_NAME</th>
<th>PHONE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_EG03</td>
<td>Check</td>
<td>Check</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EndQuery

if not executeQBE(QB, "SCHEDULE.db") then
    errorShow()
endif
endproc

proc ShowSched()
; displays the schedule
var
    TBVR TABLE
    FRM Form
endvar

if isTable("SCHEDULE.db") then
    TBVR.attach("SCHEDULE.db")
    if not TBVR.isEmpty() then
        FRM.OPEN("SCHEDULE.fsl")
    endif
endif
FRM.maximize()

else
    msgStop("SORRY", "NO APPOINTMENTS")
endif

else
    msgStop("STOP", "CAN'T FIND THE SCHEDULE TABLE")
endif
endproc

:=----------------------------------------------------------------- method pushButton(var eventInfo Event)

GetSched()
ShowSched()
endmethod

O. Open EXPENSES report

proc Ses()
;query to select session data for these dates
var
    QB query
endvar

QB = Query

SESSION.DB | DATE   | BILL | S_PAID_IN | BIL_BAL | TUT_PAY | T_PAID_OUT |
            | >=~DATE, <=~DATE1, calc ~DATE as FROM, calc ~DATE1 as TO | Check | Check | Check |
Check       | Check |
SESSION.DB | PAY_BAL |
            | Check |

EndQuery

if not executeQBE(QB, "SESDATE.DB") then
    errorShow()
endif
endproc

:=----------------------------------------------------------------- proc Exp()
;Query to select expenses for chosen dates
var
QB query
endvar

QB = Query

| EXPENSE.DB | DATE | ITEM | DESCRIPTION | COST | check >=~DATE, <=~DATE1 | check | check | CheckPlus |

EndQuery

if not executeQBE(QB, "EXPDATE.DB") then
    errorShow()
endif

endproc


proc Ser()
    ; Select service items for time period
var
    QB query
endvar

QB = Query

| SERVICES.DB | DATE | SER_COST | SER_BILL | SER_PAID | SER_BAL | >=~DATE, <=~DATE1 | checkPlus | check | check | check |

EndQuery

if not executeQBE(QB, "SERDATE.DB") then
    errorShow()
endif

endproc

method pushButton(var eventInfo Event)

VAR
    RU REPORT
ENDVAR

DATE.VIEW("ENTER THE BEGINNING DATE")
DATE1.VIEW("ENTER THE ENDING DATE")

Ses()
Exp()
Ser()

RU.OPEN("ACCOUNT.RSL")
RU.maximize()
endmethod

P. Open and print EXPENSES report

proc Ses()
;query to select session data for these dates
var
QB query
endvar

QB = Query

SESSION.DB | STU_LAST_NAM | STU_FIRST_NAM | DATE | TUTOR | HOURS |
| _join1 | _join2 | >=~DATE, <=~DATE1, CALC ~DATE AS FROM, CALC ~DATE1 AS TO | _join3 |
CheckPlus _HRS |

SESSION.DB | DISCOUNT |
| Check _DIS |

SESSION.DB | S_PAID_IN |
| Check CALC _SRATE * _HRS AS SGROSS, CALC (_SRATE * _HRS) - _DIS AS SNET |

SESSION.DB | T_PAID_OUT |
| Check CALC _TRATE * _HRS AS TGROSS, CALC _TRATE * _HRS - (_TRATE / _SRATE) * _DIS AS TNET, CALC (_TRATE / _SRATE) * _DIS AS TDIS |

STUDENT.DB | STU_LAST_NAM | STU_FIRST_NAM | RATE |
| _join1 | _join2 | _SRATE |

TUTOR.DB | TUTOR | T_RATE |
| _join3 | _TRATE |

EndQuery

if not executeQBE(QB, "SESDATE.DB") then
    errorShow()
endif
endproc
proc Exp() 
: Query to select expenses for chosen dates 
var 
   QB query 
endvar 
QB = Query

| EXPENSE.DB | DATE   | ITEM | DESCRIPTION | COST | [check >=~DATE, <=~DATE1] | check | check | CheckPlus |
EndQuery

if not executeQBE(QB, "EXPDATADB") then 
   errorShow() 
endif 
endproc

:======================================================================

proc Ser() 
: 
var 
   QB query 
endvar 
QB = Query

| SERVICES.DB | DATE | SER_COST | SER_BILL | SER_PAID | [>=~DATE, <=~DATE1] | checkPlus | check | check |
EndQuery

if not executeQBE(QB, "SERDATE.DB") then 
   errorShow() 
endif 
endproc 

:======================================================================

method pushButton(var eventInfo Event) 
VAR 
   RU REPORT 
ENDVAR 

DATE VIEW("ENTER THE BEGINNING DATE")
DATE1. VIEW("ENTER THE ENDING DATE")

Ses()
Exp()
Ser()

RU. OPEN("ACCOUNT.RSL")
RU. print()
RU. close()
endmethod

III. STUDENT/SESSIONS - LOCAL PROCEDURES

A. Close button
method pushButton(var eventInfo Event)
VAR
   frm FORM
ENDVAR
STUDENT. endEdit()
frm. attach("Form : STU_SESS.fst")
frm. close()
endmethod

B. Edit button
method pushButton(var eventInfo Event)
STUDENT. Edit()
endmethod

C. End edit button
method pushButton(var eventInfo Event)
STUDENT. endEdit()
endmethod

D. Make the look-up table open (subject 1)
method setFocus(var eventInfo Event)
if STUDENT_SESSIONS. isEdit() then
   if not eventInfo. isPreFilter() then
      STUDENT_SESSIONS. keyPhysical(32, VK_SPACE, Control)
   endif
endif
endif
endmethod

E. Make the look-up table open (subject 2)
method setFocus(var eventInfo Event)
if STUDENT_SESSIONS. isEdit() then
   if not eventInfo. isPreFilter() then
      STUDENT_SESSIONS. keyPhysical(32, VK_SPACE, Control)
F. Make the look-up table open (subject 3)
method setFocus(var eventInfo Event)
if STUDENT_SESSIONS.isEdit() then
    if not eventInfo.isPreFilter() then
        STUDENT_SESSIONS.keyPhysical(32, VK_SPACE, Control)
    endif
endif
endmethod

G. Create unique student number. Use deleted number for next student.
method pushButton(var eventInfo Event)
var
    n, v number
    tb table
    tc tcursor
endvar
STUDENT.Edit()
STUDENT.insertRecord()
doDefault
    tc.open("STUDENT.DB")
    tc.home()
    n = 1
    tc.fieldValue("STU_NO", v)
    while v = n and not tc.eot()
        tc.nextRecord()
        n = n + 1
        tc.fieldValue("STU_NO", v)
    endwhile
STU_NO = n
action(DataPostRecord)
endmethod

H. This method attempts to locate a user-entered student name, using
the built-in Locate Value dialog box.
method pushButton(var eventInfo Event)
var
    retVal Logical
endvar
retVal = STUDENT.action(DataSearch) ; call dialog and perform search
if not retVal then ; If search failed...
    beep() ; beep and display a message
msgInfo("Oops!", "Either you canceled the dialog box or I couldn't find the name you entered.")
endif
endmethod

1. Make the look-up table open (tutor)
method setFocus(var eventInfo Event)
if STUDENT_SESSIONS.isEdit() then
  if not eventInfo.isPreFilter() then
    STUDENT_SESSIONS.keyPhysical(32, VK_SPACE, Control)
  endif
endif
endmethod

J. Calculate session balance after filling in student paid.
method removeFocus(var eventInfo Event)
var
  tb1, tb2, tb3 table
t time
endvar
if isEdit() then
  if not END.isBlank() then
    tb1.attach("STUDENT_DB")
    tb2.attach("SESSION_DB")

    BIL_BAL = (((END.HOUR() + (END.MINUTE()/60.00)) - (START.HOUR() + (START.MINUTE()/60.00)) * RATE) - DISC) - PAID_EDIT
  endif
  action(DataPostRecord)
endif
endmethod

K. Prevents the filling in of an end time for sessions that are scheduled for the future method canArrive(var eventInfo MoveEvent)
if DATE_EDIT.value > today() then
  MsgStop("ERROR", "THIS SESSION HAS NOT HAPPENED YET.")
  eventInfo.setErrorCode(CanNotArrive)
endif
endmethod

L. Calculates session hours, session gross, session bill tutor's fee, tutor's adjustment, tutor's pay, bill balance, and pay balance.

PROC DURATION(var C_hours number)
C_hours = (END.HOUR() + (END.MINUTE()/60.00)) - (START.HOUR() + (START.MINUTE()/60.00))
IV. TUTOR/SESSIONS LOCAL METHODS
   A. Close TUTOR/SESSIONS

   method pushButton(var eventInfo Event)
   VAR
      frm FORM
   ENDVAR
   TUTOR.endEdit()
   frm.attach("Form : TUT_SESS.fsl")
   frm.close()
   endmethod

   B. Edit subject table
method pushButton(var eventInfo Event)
var
    tb tableview
endvar

    tb.open("SUBJECT")
tb.wait()
tb.close()
endmethod

C. Open look-up table (subject 1)

method setFocus(var eventInfo Event)
if TUTOR.isEdit() then
    if not eventInfo.isPreFilter() then
        TUTOR.keyPhysical(32, VK_SPACE, Control)
    endif
endif
endmethod

D. Open look-up table (subject 2)

method setFocus(var eventInfo Event)
if TUTOR.isEdit() then
    if not eventInfo.isPreFilter() then
        TUTOR.keyPhysical(32, VK_SPACE, Control)
    endif
endif
endmethod

E. Open look-up table (subject 3)

method setFocus(var eventInfo Event)
if TUTOR.isEdit() then
    if not eventInfo.isPreFilter() then
        TUTOR.keyPhysical(32, VK_SPACE, Control)
    endif
endif
endmethod

F. Open look-up table (subject 4)

method setFocus(var eventInfo Event)
if TUTOR.isEdit() then
    if not eventInfo.isPreFilter() then
        TUTOR.keyPhysical(32, VK_SPACE, Control)
    endif
endif
endmethod
G. This method attempts to locate a user-entered tutor name, using the built-in Locate Value dialog box.

```plaintext
method pushButton(var eventInfo Event)
  var
    retVal Logical
  endvar
  TUTOR.moveTo(); move to TUTOR field
  retVal = TUTOR.action(DataSearch); call dialog and perform search
  if not retVal then  ; If search failed...
    beep();          ; beep and display a message
    msgInfo("Oops!", "Either you canceled the dialog box or I couldn't find the name you entered.")
  endif
endmethod

H. Add new tutor

method pushButton(var eventInfo Event)
  TUTOR.EDIT()
  TUTOR.insertRecord()
endmethod

I. Edit TUTOR/SESSIONS

method pushButton(var eventInfo Event)
  TUTOR.EDIT()
endmethod

J. Endedit TUTOR/SESSIONS

method pushButton(var eventInfo Event)
  TUTOR.endEdit()
endmethod

K. Can't edit these fields: STU_LAST_NAM, STU_FIRST_NAM, DATE_VIEW, SUBJECT_VIEW, HOURS, TUT_FEE, LOCATION_VIEW, TEST_DATE, RESULTS, NOTES_VIEW, TUT_ADJ, TUT_PAY, and PAY_BAL

method canArrive(var eventInfo MoveEvent)
  eventInfo.setErrorCode(CanNotArrive)
endmethod

L. Calculate PAY_BAL from TUT_PAID
method removeFocus(var eventInfo Event)
  var
    tb1, tb2 table
    t time
  endvar
  if isEdit() then
    if not END.isBlank() then
      tb1.attach("TUTOR.DB")
      tb2.attach("SESSION.DB")
      PAY_BAL = ((((END.HOUR() + (END.MINUTE() / 60.00)) - (START.HOUR() +
      (START.MINUTE() / 60.00))) * T_RATE) - (DISC * (T_RATE / RATE))) - TUT_PAID
    endif
    action(DataPostRecord)
  endif
endmethod

V. SERVICES LOCAL PROCEDURES

A. Close button
method pushButton(var eventInfo Event)
  VAR
    frm FORM
  ENDVAR
  SERVICES_EDIT.endEdit()
  frm.attach("Form : STU_SER.fsl")
  frm.close()
endmethod

B. End edit button
method pushButton(var eventInfo Event)
  SERVICES_EDIT.endEdit()
endmethod

c. Edit button
method pushButton(var eventInfo Event)
  SERVICES_EDIT.Edit()
endmethod

D. Can't edit LAST_NAME, FIRST_NAME, STREET, CITY, STATE, ZIP, and PHONE
method canArrive(var eventInfo MoveEvent)
  eventInfo.setErrorCode(NotArrive)
endmethod

E. Calculate services balance
method removeFocus(var eventInfo Event)
  var
tb1, tb2 table
endvar

if isEdit() then
    BAL_CALC = BILL_EDIT - PAID_EDIT
endif
endmethod

VI. EXPENSES LOCAL PROCEDURES
A. Close button
method pushButton(var eventInfo Event)
VAR
    frm FORM
ENDVAR
EXPENSE.EndEdit()
frm.attach("Form : EXPENSES.fsi")
frm.close()
endmethod

B. Edit button
method pushButton(var eventInfo Event)
EXPENSE/Edit() 
endmethod

C. End edit button
method pushButton(var eventInfo Event)
EXPENSE.EndEdit() 
endmethod

VII. SCHEDULE LOCAL PROCEDURES
A. Close button
method pushButton(var eventInfo Event)
VAR
    frm FORM
ENDVAR
frm.attach("Form : SCHEDULE.fsi")
frm.close()
endmethod