Distributed Gaming using XML

CS 297 Project Report

Padmini Paladugu

paddusyam@yahoo.com

Advisor: Dr. Chris Pollett

May 21, 2003

Abstract

Advancements in wireless technology have led to the emergence of a wide variety of wireless devices like Personal Digital Assistants (PDAs) and cell phones. A number of applications like word processors have been developed to run on these devices and among the most popular are gaming applications. The eXtensible Markup Language (XML) is an emerging technology for describing and interchanging data among various systems and databases. In this project, we will develop a Pokeman-style game played on wireless devices that uses a stripped down XML language that we create to communicate with a centralized Oracle Database. Various properties of our language and game set-up will be tested for efficiency and playability.

In this project, we have a number of wireless users on the one hand and an Oracle database on the other. Each player starts a game on a PDA and chooses an option about what kind of player he will be. After that, his choice is recorded into the centralized database. Depending on the type of player, the centralized database will assign an initial task to each player, and when one device gets into the range of another device then the devices will beep as an indication. If two people decide to play each other, they can point their devices at each other and send some token for the verification of transaction occurrence between them. In addition, if one player agrees that the other player's goal was completed, an additional token will be given to players. Also, players can synchronize with central database to update their scores, and the central database will assign new tasks and new powers to players. The Central database has a high score list which can be synchronized with the palm devices. The central database keeps track of all XML based messages that have passed between the devices and chooses next goals based on this information. Communication between players can be achieved using beaming and Bluetooth technologies.

TABLE OF CONTENTS

1. INTRODUCTION

2. XML and SAX PARSER
2.1 XML
2.2 SAX Parser
2.3 Deliverable 1

3. ORACLE XML DATABASE
3.1 Key Features of Oracle XML Database
3.2 Benefits of Oracle XML Database
3.3 Storing and Retrieving XML data
3.3.1 Storing XML data using XMLType datatype
3.3.2 Retrieving XML data using XMLType datatype
3.3.3 Updating XML Documents
3.4 Deliverable 2: Playing Tic-Tac-Toe Game and Storing data in XMLType Table

4. PALM OS CONCEPTS
4.1 Creating Palm OS applications
4.2 Deliverable 3: Storing and Displaying Person's Info

5. CONCLUSION

6. REFERENCES

APPENDIX

1. Introduction

Advances in wireless technology have led to the emergence of a wide variety of wireless devices like Personal Digital Assistants (PDA) and cell phones. Several customized applications have been developed to run on these devices, e.g. Word Processing applications. Among these, gaming applications have gained a lot of significance. Today single player games and double player games on PDAs are available on the market. The invention of distributed games has opened up the new avenues of entertainment for users. To facilitate distributed gaming, technologies like XML, Oracle, and Bluetooth technologies are employed.

Extensible Markup Language (XML) is a model for describing the structure of information, which makes it easier to transfer ordered information from one place to another place, or from one program to another program. XML performs a similar role for exchanging both complex data and simple data. It provides the syntax for defining the structure of information using XML Schemas or Document Type Definitions (DTDs). The XPath and XQuery features of an XML can be used to store and retrieve XML messages from the relational database system like Oracle9i.

Oracle9i provides support for handling XML data and documents. Oracle XML Database introduces a new datatype called XMLType while defining tables, columns in tables, and views. It can be used to store and manage XML documents in the same way it stores and manages other datatypes. With this, the power of relational database is enhanced in the context of XML. Oracle9i allows XPath expressions to navigate through an XMLType instance, and to search across multiple instances of XMLType.

The general goals of gaming on PDAs are providing communication between players, maintaining consistent state between players and central database, and managing all players' data in the database simultaneously.

In this project, Bluetooth technology is used for providing communication between wireless devices; devices communicate with database using XML messages. A centralized database is used to maintain consistency. Oracle based technology is used to manage details of all the players simultaneously.

This report is organized as follows. In Section 2, we briefly discuss about XML and SAX Parser. We present Oracle XML details in Section 3. In Section 4, we discuss about Palm OS concepts. Section 5 concludes our research project.

2. XML and SAX Parser

This section introduces the concepts of XML and SAX Parser. First we discuss the different concepts in XML and its syntax. Then we introduce the SAX Parser, which is used to retrieve XML data using JAVA. Finally, we illustrate these concepts in the context of deliverable that we have implemented.

2.1 eXtensible Markup Language (XML)

eXtensible Markup Language (XML) has established a new epoch in the web technology by facilitating an efficient way of transmitting structured data. XML is a standardized text format for representing structured information on the web. XML is language-independent and platform-independent. It is used on Linux, Windows, Macintosh, and many other computer platforms. It has become a standard for newly designed document formats across almost all computer applications. The messages to be transmitted while playing games on wireless devices can be written in XML. XML provides interoperability between different platforms and different machines.

XML is a markup language similar to Hyper Text Markup Language (HTML). Unlike HTML, XML has user-defined tags. The standardization and simplicity of the XML syntax rules make it easier to learn and use. All XML elements must have a beginning and closing tags. XML tags are case sensitive and they must be properly nested to maintain correctness. All XML elements have a relationship between them and they are related as parents and children. To check for the correctness and integrity of rules, XML provides the well formedness check and validity check. XML with correct syntax rules is the well- formed XML document. Validating XML document against certain predefined rules provides the valid XML document.

Validation of XML documents is required to make sure that a given XML document follows the defined rules. The validation of each XML document is done against its corresponding Document Type Definition (DTD). DTDs are written in a proper syntax that explains where the elements and entities may appear in the document. It also explains the contents of element and attributes. The validating parser compares a document with its DTD and lists all details where the document differs from the constraints specified in the DTD. XML tools like XMLSPY and cooktop can be used to validate a document against its DTD. XML Schema enables a unified data model that can address both documents and structured data. XML Schema has emerged as a key innovation in managing document's content.

2.2 SAX Parser

Parsing is the method of reading an XML document and retrieving its content while checking for the document well formedness. The *Simple API for XML (SAX)* is an event-based API for reading XML documents. SAX will be used to create an XML-to-Java mapping with a minimum effort, even for relatively complex XML structures. To map XML data to a Java program, initially we should make sure that each character in the XML document has legal XML tokens such as start tags, attributes, end tags, and CDATA sections. Then, we should verify these tokens for well formedness. We need to confirm that all of the tags have matching opening and closing tags, and the attributes are appropriately arranged in the opening tag. If a DTD is also available, we have to ensure that the XML constructs found during parsing are valid in terms of the DTD, as well as well formedness.

SAX is mainly a collection of interfaces in the *org.xml.sax package*. SAX is an API, so the code is standard across all XML parsers. To run java applications that are using SAX, first we need to access an XML parser that supports SAXv2. We use Apache's *Xerces* parser. The primary task of SAX API is creating a class that implements the *ContentHandler* interface, a callback interface used by XML parsers to inform our program of SAX events as they are found in the XML document. The SAX API also provides a *DefaultHandler* implementation class for the *ContentHandler* interface. Once we have implemented the *ContentHandler* or extended the *DefaultHandler*, we need to direct the XML parser to parse a given document. We implemented a deliverable using XML and SAX API technologies.

2.3 Deliverable 1: TicTacToe Board display using XML and Java

Deliverable: This deliverable was supposed to describe the state of *TicTacToe board* using the XML Language, and to parse this XML document using SAX Parser. The result is then supposed to be displayed using Java Swing

The main purpose of this deliverable is to become familiar with writing XML documents and DTDs. This deliverable is also helpful to become proficient in using SAX Parser, which is used to retrieve XML data using JAVA. In this deliverable we did not perform the actual play of the game, rather we described the state of TicTacToe board using XML and displayed the board using JAVA.

TicTacToe Game Description: The TicTacToe game is played on a 3x3 board, and two players (X and O) will play the game. Each player occupies a single cell. Usually 'X' goes first, and there are nine different locations in which 'X' can go. After this cell is occupied there are eight different locations for 'O' to go, and they will continue to do it until one of the players has three cells aligned (either with ‘X’ or ‘O’) horizontally, vertically or diagonally.

In this deliverable we have implemented only the state of a TicTacToe board. Each cell in the board is occupied by one of ‘X’, ’O’ or ‘B’. First player uses character ’X’, Second player uses character ’O’, and ’B’ symbolizes the blank symbol. The XML document in Figure 1 characterizes the state of TicTacToe board.

In Figure 1, line 1 represents XML comment. Parsers will not parse the comments. The line 2 defines the XML version and the character encoding used in the document. The external DTD is declared in line 3. The external DTD validates the tboard.xml document against the document tboard.dtd shown in Figure 2. Line 4 describes the root element of the document, and lines 5,10,13 describe the child elements of the root. Child elements must be correctly nested within their parent element. XML elements can also have attributes. The attribute values must be enclosed in quotes. These attributes provide the additional information about the element. Lines 6-8, 11-13,and 16-18 describe the elements with attribute values ‘X’, ’O’, and ‘B’. Figure 2 describes the rules for validating XML document. Lines 2-5 in Figure 2 describe the order of root element and

its sub elements. Lines 6-8 represent the contents of an element, and lines 9-11 explains about the possible attribute values for the element.

```
1.  <!-- tboard.xml- an XML document was designed to describe the state of TicTacToe
    board. This document describes a standard 3 X 3 TicTacToe Board. -->
2.  <? xml version="1.0" standalone="no"? >
3.  <! DOCTYPE board SYSTEM "tboard.dtd">
4.  <board>
5.  <row1>
6.          <column1 value="O"/>
7.          <column2 value="O"/>
8.          <column3 value="O"/>
9.  </row1>
10. <row2>
11.         <column1 value="X"/>
12.         <column2 value="X"/>
13.         <column3 value="X"/>
14. </row2>
15. <row3>
16.         <column1 value="B"/>
17.         <column2 value="B"/>
18.         <column3 value="B"/>
19. </row3>
20. </board>
```

Figure 1: XML document describing the state of a TicTacToe Board (tboard.xml).

```
<!-- tboard.dtd - an XML DTD was designed for  validating the tboard1.xml
document   -->
<! ELEMENT board (row1, row2, row3)>
<! ELEMENT row1 (column1, column2, column3)>
<! ELEMENT row2 (column1, column2, column3)>
<! ELEMENT row3 (column1, column2, column3)>
<! ELEMENT column1 EMPTY>
<! ELEMENT column2 EMPTY>
<! ELEMENT column3 EMPTY>
<! ATTLIST column1 value (O | X | B) #REQUIRED>
<! ATTLIST column2 value (O | X | B) #REQUIRED>
<! ATTLIST column3 value (O | X | B) #REQUIRED>
```

Figure 2: XML DTD for validating tboard.xml in *figure1* (tboard.dtd)

There are two classes TicTacToeParser.java and TicTacToeBoard.java available in *Appendix* for this deliverable. *TicTacToeParser* deals with parsing XML document using SAX API and java. This class is designed to read an XML document using SAXParser. It is responsible for fetching attribute values from the XML document and placing attribute

values into a *Vector*. The TicTacToeParser class will override the methods of the *DefaultHandler* class to gain notification of SAX Events. The code in Figure 3 receives the notification of the beginning of the XML document.

```
public void startDocument() throws SAXException
{
}
```

Figure 3: Begin document

The code in Figure 4 receives notification of the end of the XML document.

```
public void endDocument () throws SAXException
{
}
```

Figure 4: end document

The code in Figure 5 receives notification of the start of an element in the XML document and attribute values in an XML document.

```
public void startElement (String namespaceURI, String localName, String qName, Attributes attr )
throws SAXException
{
}
```

Figure 5: Start element and attributes

The code in Figure 6 receives notification of the end of an element in the XML document.

```
public void endElement (String namespaceURI, String localName, String qName)
throws SAXException
{
}
```

Figure 6: end of an element

The code in Figure 7 receives notification of the character data inside an element in the XML document.

```
public void characters (char[] ch, int start, int length) throws SAXException
{
}
```

Figure 7: Character data

The main method will read the XML document using FileReader and parse it using the parser.

The *TicTacToeBoard* class is designed to display TicTacToeBoard using JButtons. The result of this deliverable is shown in the Figure 8.

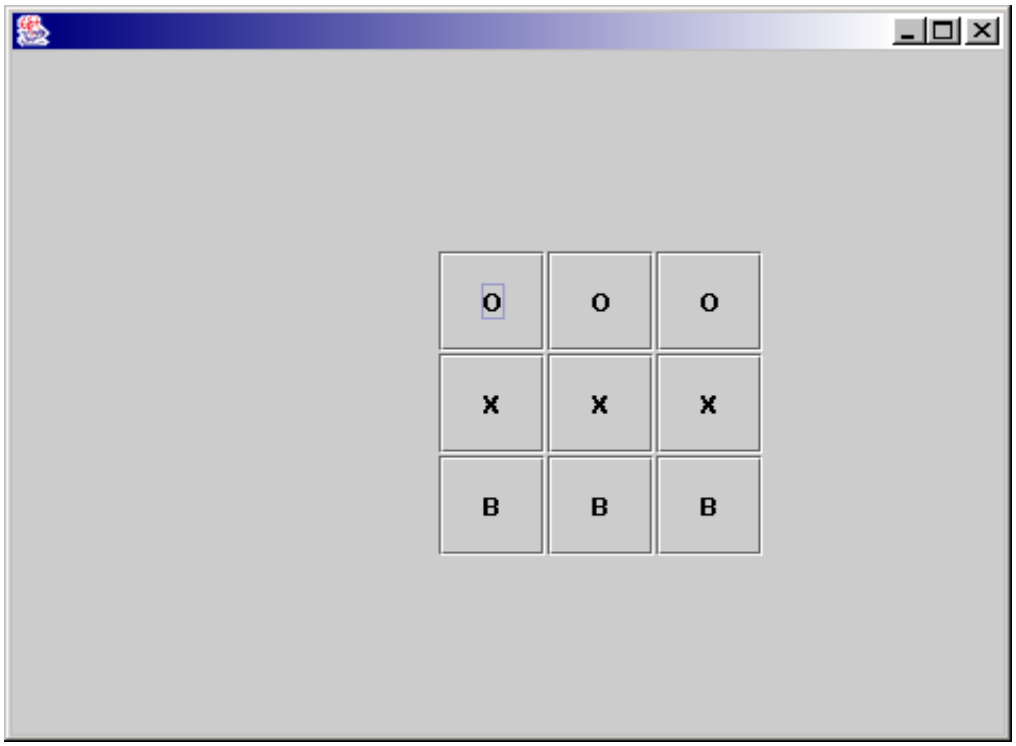

Figure 8: TicTacToe Board

3.Oracle XML Database

The XML features of the Oracle9i Database Management System (Oracle XML) provide tools for building XML applications. Oracle XML can be used to store, query, update, transform, and process XML, while providing SQL access to the XML data.

3.1 Key Features of Oracle XML Database [ORA9i]

Key Features in Oracle XML include XML Types, DOM fidelity, Document fidelity, XML schema, XML Schema storage with DOM fidelity, XML Schema validation, XML piecewise update, XPath search, XML indexes, SQLX operators, XSL transformations for XMLType, PL/SQL and OCI interfaces, Schema caching, XML generation, Oracle XML Database Repository, SQL Repository search, WebDav, HTTP, and FTP access, and Versioning.

XMLType datatype can be used to store and retrieve XML data to and from the database. Using XMLType datatype we can perform following SQL operations:

- Querying and invoking OLAP functions on XML data.
- Searching XPath and transformation of XSL on SQL data.

XML Schema validation in Oracle XML allows validation of XML documents stored in database against their schema. Using *XML piecewise update*, XPath expressions specify individual elements and attributes of an XML document during updates without rewriting the entire document. This is more efficient for large XML documents.

XPath search uses XPath syntax to query XML content in the database. *XML indexes* are indices created for XPath searches using XPath expressions. Indices enable fast access to XML documents.

3.2 Benefits of Oracle XML Database [ORA9i]

Applications usually manage structured data as tables and unstructured data as files or Large Objects (LOBs). This enables developers to execute on different kinds of data.

Unstructured Data: Document access is transparent and table access is complex with unstructured data.
Structured Data: Document access is complex and table access is transparent with structured data.

Oracle XML provides the following benefits:

- Oracle XML has the ability to store and manage both structured and unstructured data.
- Oracle XML provides the complete transparency and interchangeability between the XML and SQL data views.
- Oracle XML offers a better management of unstructured XML data by supporting
    - Piecewise updates
    - XML indexing
    - Integrated XML text search with Oracle Text
    - Multiple views on the data including relational views for SQL access
- Oracle XML provides high performance and scalability for XML operations by providing XML-specific data caching and memory management, query optimization on XML, and so on.
- Oracle XML has the capacity to access data and documents from disparate systems and combine it into a standard data model. This integration reduces the complexity of developing applications that deal with data from different sources.

3.4 Storing and Retrieving XML data

3.4.1 Storing XML data using XMLType datatype [ORA9i]

XML documents are stored in the Oracle database either by using XMLType column or XMLType table. Using these options indicates to the database that it should manage XML content. Figure 9 explains the creation of a table with an XMLType column.

```
CREATE TABLE ttboard
(KEYVALUE varchar2 (10) primary key,
   XMLCOLUMN xmltype
);
```

Figure 9: Creating table with XMLType column

Figure 10 explains the creation of table with an XMLType column. The code in the Figure 10 is taken from the Deliverable 2 in Appendix.

```
CREATE TABLE ttboard OF XMLType;
```

Figure 10: Creating table with XMLType table.

To store an XML document in an XMLType table or column we need to set up the following:

- Connect as system/manager
- Create directory XMLDIR as '/padmini/xmlsamples'.
- Grant read permission on directory xmldir to all (public) with grant option.

To store an XML document in an XMLType table or column the XML document must first be converted into an XMLType instance. This is done using the getDocument () PL/SQL function. This function is shown in the Figure 11.The code in the Figure 11 is taken from the Deliverable 2 in Appendix.

```
create or replace function getDocument (filename varchar2) return clob authid
current_user is

    xbfile bfile;
    xclob clob;
    begin

        xbfile  := bfilename('XMLDIR', filename);
        dbms_lob.open (xbfile);
        dbms_lob.createtemporary(xclob,TRUE,dbms_lob.session);
      dbms_lob.loadfromfile(xclob,xbfile, dbms_lob.getlength(xbfile));
      dbms_lob.close(xbfile);
      return xclob;

   end;
```

Figure 11: getDocument function

Figure 12 shows the insertion of an XML document into XMLType table ttboard. The code in the Figure 12 is taken from the Deliverable 2 in Appendix.

```
INSERT INTO ttboard VALUES(XMLTYPE(getDocument('board.xml')));
```

Figure 12: Inserting XML document into an XMLType table

3.4.2 Retrieving XML data using XMLType datatype [ORA9i]

After storing the collection of XML documents into an XMLType table or column, the next step is retrieving them. Oracale 9i uses the concept of XPath to traverse through all the sub elements in the root element. Oracle XML uses the XPath in conjunction with the extract (), extractValue (), and existsNode () functions.

The *existsNode ()* function verifies the presence of a node in the XML document. The existsNode () function returns true if the document contains the node specified by the XPath expression supplied to the function. The existsNode () function is commonly used in the WHERE clause of SQL SELECT, UPDATE, or DELETE statements. In these cases the existsNode () function determines which of the XML documents stored in the table the SQL statement will process.

The *extractValue ()* function is used to retrieve the value of a text node or attribute associated with an XPath Expression. It returns a scalar data type. Only the extractValue () function can return the value of a single node. The extractValue () function can also be used in the WHERE clause of a SELECT, UPDATE, or DELETE statement. It is useful to perform joins between XMLType tables or tables containing XMLType columns and other relational tables.

The *extract ()* function is used retrieve all nodes in an element node specified in the XPath expression. The nodes are returned as an instance of XMLType. The results of extract () can be either a Document or a Document fragment.

3.4.3 Updating XML Documents

XML documents can be updated using the *updateXML ()* function. The updateXML () function updates an attribute value, node, text node, or node tree. The target for the update operation is identified using an XPath expression. The code in Figure 13 illustrates the update operation. The code in the Figure 13 is taken from the Deliverable 2 in Appendix.

```
UPDATE ttboard t SET value (t) = updateXML(value(t),'/board/row3/column1/text()','O');
```

Figure 13: Updating XML document.

3.5 Deliverable 2: Playing Tic-Tac-Toe Game and Storing data in XMLTYPE Table

Deliverable: Create a language to represent a play on a tic-tac-toe board and write a program that connects to an Oracle database using its XML features to play a tic-tac-toe using this language.

The main purpose of this deliverable is to familiarize myself with Oracle 9i, Client-Server programming, and JDBC. The deliverable primarily deals with storing and retrieving of XML documents in Oracle9i using XMLType column or table. This deliverable allowed me to explore new concepts in Oracle9i, i.e. the interface between Oracle9i and XML.

This deliverable is designed for creating a play on a TicTacToe Board. Two players will play the TicTacToe game. The client sends each player's move to the server. The server accesses the database and updates the data in the XMLTYPE table. Stream sockets are used in establishing communication between client and server.

This application communicates with the database using the Java Database Connectivity (JDBC). Computers communicate with each other by using the Transport Control Protocol (TCP). Data transmitted is accompanied by addressing information that identifies the computer and the port for which it is destined. A server registered to a port receives all data destined for that port.

The JDBC API is an application used to access a database resource from Java. In this deliverable, it is used to establish connection with relational database (Oracle9i). SQL is the standard language for accessing relational database.

This deliverable consists of four classes TicTacToeClient.java, TicTacToeServer.java, TicTacToeServerThread.java, and StringDecoder.java. Implementation details of all these classes are shown in Appendix. Client program display the TicTacToe board and allows the players to enter their choice. Then the player’s move is sent to the server. The player name, row number, and column number are sent to the server. Server decodes this message using StringDecoder class and divides it into tokens. Then Server evaluates the each player’s move, makes a connection with database, and updates the board in the database. The server prepares the update statement by concatenating the palyer name, row number, and column number to update the board in the database. It creates the XPath expression to update the specified node. Game will reach the end when one of the two players places either ‘X’, or ‘O’ diagonally, vertically, or horizontally. The end of the game status will be shown at the client side user interface. If one player already occupies a cell, the second user will not be allowed to occupy the same cell. The board is described in the database as an XML document. It is stored in the XMLType table. The code in Figure 14 shows the establishment of connection with database.

```
Connection connection;
Statement statement;
DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
String username="scott",password="tiger",dbURL="jdbc:oracle:oci:@pgdev";
connection=DriverManager.getConnection(dbURL,username,password);
statement=connection.createStatement();
String sql="UPDATE ttboard t SET value(t) = updateXML(value(t),'/board/row";
sql+=moveNumber[1];
sql+="/column";
sql+=moveNumber[2];
sql+="/text()','";
sql+=moveNumber[0];
sql+="')";
System.out.println("Update Statement="+sql);
statement.executeUpdate(sql);
statement.close();
connection.close();
```

Figure 14: Connection with database

Figure 15 shows the state of a board in the database before playing the game.

```
SQL> set long10000
SQL> select * from ttboard;
SYS_NC_ROWINFO$
--------------------------------------------------
<board>
        <row1>
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row1>
        <row2>
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row2>
        <row3>
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row3>
</board>
```

Figure 15: State of board before the game

Figure 16 shows the client program's output after completing the TicTacToe game. In this figure player 'X' wins the game by placing all the three values diagonally.

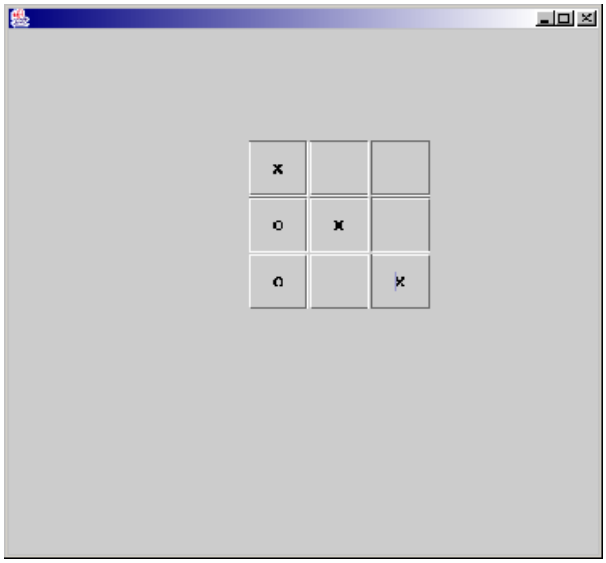

Figure 16: Client program's output

Figure 17 shows the state of a board in the database after completing the game.

```
SQL> set long10000
SQL> select * from ttboard;
SYS_NC_ROWINFO$
--------------------------------------------
<board>
        <row1>
                <column1>X</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row1>
        <row2>
                <column1>O</column1>
                <column2>X</column2>
                <column3>B</column3>
        </row2>
        <row3>
                <column1>O</column1>
                <column2>B</column2>
                <column3>X</column3>
        </row3>
</board>
```

Figure 17: State of a board after game

4.Palm OS Concepts

Personal Digital Assistants (PDAs) unique features like small size and elegant interface made people to move towards these handheld devices. These devices do not have a hard disk to store programs and applications and have very Random Access Memory (RAM). The unique features of handheld devices require special Operating System such as Palm OS. Palm OS is the predominant technology in the field of handheld devices and is developed by PalmSource Inc. Palm OS technology broadly includes communication technologies used in the devices such as PDAs, cell phones, and pagers. Following paragraphs describe various communication techniques used in these devices.

The significance of World Wide Web and Internet-based applications has motivated the Palm devices to introduce wireless applications. Wireless communications concept has included in devices using Palm III and above versions. Starting from the Palm III, Palm devices can communicate via the industry-standard Infrared Data Association (IrDA) protocol. This is a low-level communications protocol, and it can be used in infrared enabled devices like PDAs, cell phones, pagers, and even desktop or laptop computers.
Infrared Data Association (IrDA) infrared communication protocols can be used to send an arbitrary “thing”, or data object from one device to another. More recently, the high-level communication protocol like Object Exchange protocol (OBEX) is used to exchange objects between devices.

The Object Exchange (OBEX) [MSK03] protocol is an efficient high-level communication protocol that enables a wide range of devices to exchange data in a simple and spontaneous manner. The OBEX protocol is developed by members of the Infrared Data Association to connect a full range of devices that support IrDA protocols. But, it is not limited to use in an IrDA environment. OBEX can be used in PCs, pagers, PDAs, phones, printers, cameras, auto-tellers, information kiosks, calculators, data collection devices, watches, home electronics, industrial machinery, medical instruments, automobiles, and office equipment. To support these applications, OBEX is designed to transfer flexibly defined “objects”. Finally, OBEX can be used to perform complex tasks such as database transactions and synchronization.

Bluetooth is another technology used to provide communication between wireless devices. Bluetooth is a standard for short-range radio communication. It is a built in technology used in the devices, and it can provide communication between two Bluetooth enabled devices. Evolution of wireless communication between the devices has led to the popularity of two-player and Multi-player games.

PDA games are a rapidly growing market. Since the initial release of Palm devices, PDAs have been a part of the gaming industry. Initially, the gaming market has been dominated by simple games that are played on PC and GameBoy family. Recently, PDAs are playing a vital role in the market by implementing the mobile games and multiplayer games. In this project, we will develop a Pokeman-style game played on wireless devices that uses a stripped down XML language that we create to communicate with a centralized Oracle Database.

4.1 Creating Palm OS Applications

Palm OS applications can be created on different environments like Metrowerks CodeWarrior, GNU PRC-tools, and Falch.net Developer studio. We are using GNU PRC-tools to create and run these applications. Figure 18 illustrates the steps for creating a Palm OS application using GNU PRC-tools.

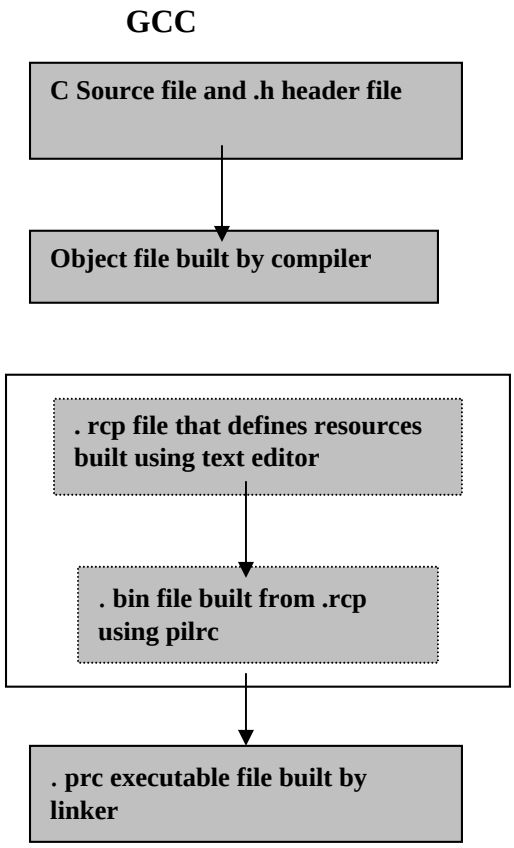

Figure 18: Creating Palm OS application using GCC

As shown in the Figure 18 Palm OS application is a combination of different files. The source code for application is stored in .C file. In addition, each application will use a header file with .h extension to define constants used in the application. Compiler will create an object file after compiling the source code. The .rcp file will create all resources required for the application. Pilrc application will be used to convert .rcp file to .bin file. Finally, the linker will combine the contents of the object file with the .bin resource file to build .prc executable application.

4.2 Deliverable 3: Storing and Displaying Person's Info

Deliverable: Write a Palm Pilot program that takes a person's first name, last name, and phone number and store in a Palm Database. This program should allow a person to see list of all names currently stored in the database.

The main purpose of this deliverable is to acquire knowledge about Palm Pilot programming. This Deliverable covered numerous concepts in Palm Pilot programming: Forms, Lists, Text fields, and Palm OS databases. This deliverable helped me to take an exhaustive tour of Palm Pilot programming.
Deliverable 3 explains the concepts of writing rcp files to create all interface components, programming all interface components in the rcp file, and creating Palm OS databases to store person's info in the database. Delivearble 3 uses the following components: Forms, Text fields, Buttons, Alerts, and List Interface. Figure 19 explains the code for creating Form resource.

```
FORM ID MainForm 0 0 160 160
MENUID MainMenuBar
USABLE
BEGIN
   TITLE "Person's Details"
   LABEL "FName:" AUTOID AT (0 17) FONT 1
   LABEL "LName:" AUTOID AT (0 34) FONT 1
   LABEL "Phone:" AUTOID AT (0 51) FONT 1
   FIELD ID FirstNameField AT (40 17 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   FIELD ID LastNameField AT (40 34 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   FIELD ID PhoneNumField AT (40 51 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   BUTTON "SAVE"   ID SaveButton AT (30 117 50 30)
   BUTTON "LIST" ID ListButton
      AT (PrevRight+5 PrevTop 50 30)
   GRAFFITISTATEINDICATOR AT (140 PrevTop)
END
```

Figure 19: Creating main form with textfields and buttons.

The form in Figure 19 represents the main form. It contains three labels with title Fname, Lname and Phone, three text fields to enter the person's details, and two buttons.
Clicking on the save button will save the person's info into the database, and clicking on the list button will display all records stored in the database. Figure 20 shows the user interface implemented using resources.

Figure 20: Mainform User Interface

Figure 21 shows the Mainform after entering the person's info.

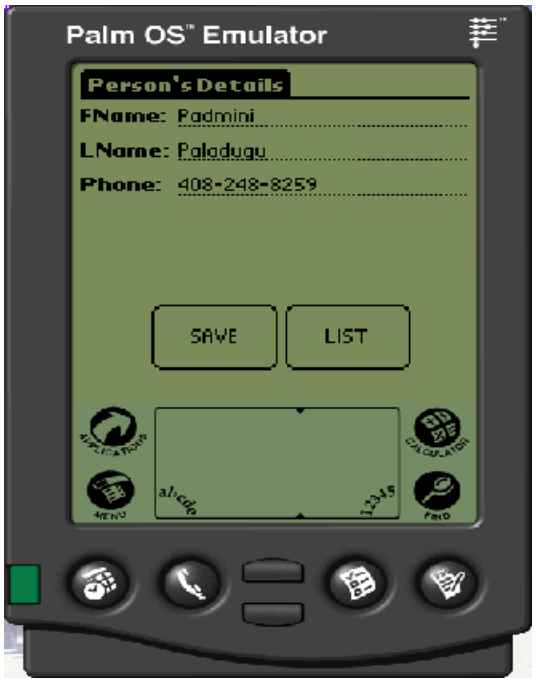

Figure 21: Person's Info

After entering the person's details, clicking on the save button will create a database, generate record for all these three text fields, and write record into the database. DatabaseManager.c file in Appendix will give implementation details about creating database and saving records into the database. Once the records are stored into the database, we can retrieve them by clicking on the list button in the Mainform.Figure 21 shows all the records stored in the database.

Figure 22: List of records

5. Conclusion

XML is language-independent and platform-independent. XML is a standardized text format for representing structured information on the web. Oracle XML completely absorbs the XML data model into Oracle9*i* database. It provides new access methods for navigating and querying XML.Using Oracle XML we will acquire all the advantages of relational database technology and XML technology at the same time.

Today single player games and double player games on PDAs are available on the market. The invention of distributed games has opened up the new avenues of entertainment for users. By giving several options to the players this distributed gaming system envelops the current technologies like XML, Oracle, and Bluetooth technologies.
For this project, a Pokeman-style game to be played on wireless devices is to be developed. A stripped down eXtensible Markup Language (XML) is created to facilitate communication between wireless device and a centralized Oracle Database. Various properties of our language and game set-up will be tested for efficiency and playability.
Communication between players can be achieved using beaming and Bluetooth technologies.

6.References

[EN00] Elmasri, R. A., & Navathe, S. B. (2000). Fundamentals of Database Systems (3rd ed.). Addison-Wesley.

[F00] Foster, L.R. (2000). Palm OS Programming Bible. Hungry Minds, Inc.

[HM02] Harold, R. E., & Means, S.W. (2002). XML in a Nutshell (2nd ed.). O'Reilly.

[KBGP01] Kaljuvee, O., Buyukkokten, O., Garcia-Molina, H., Paepcke, A. (2001,April). Efficient Web Form Entry on PDAs.Proceedings of the tenth international conference on World Wide Web, 663-672.

[LLF98] Leventhal, M., Lewis, D., & Fuchs, M. (1998). Designing XML Internet applications. PrenticeHall.

[MSK03] Megowan, P., Suvak, D., Kogan, D. (2003). Infrared Data Association. (IrDA.) Object Exchange Protocol (OBEX.). *Extended Systems, Inc Microsoft Corporation.*

[ORA9i] Oracle9*i* XML Database Developer's Guide - Oracle XML DB.

[RG02] Ramakrishnan, R., & Gehrke, J. (2002). Database Management Systems (3rd ed.). McGraw-Hill.

[SM01] Sundaresan, N., Moussa, R. (2001,May). Algorithms and Programming Models for Efficient Representation of XML for Internet Applications. Proceedings of the tenth international conference on World Wide Web, 366-375.

[S01] Suciu, D. (2001,September). On Database Theory and XML.SIGMOD Record (8), 39-45.

[TVBSSZ02] Tatarinov, I., Viglas, D.S., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C. (2002,June). Storing and Querying Ordered XML using a Relational Database System. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 204-215.

[WML01] (2001,March). Wireless Developer Network - WMLScript Tutorial http://www.wirelessdevnet.com/channels/wap/training/wmlscript.html. W3C.

Appendix

The code below is complete implementation of our deliverables.

Deliverable1

```
/*****************************************************************************
 Class Name: TicTacToeParser.java
 Author:        Padmini Paladugu
 Purpose:      This class is designed to read an XML document using SAXParser. It is responsible for
               fetching Attribute values from XML document and placing Attribute values into a Vector.
 *****************************************************************************/

/* Imported Packages */
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;
import java.util.*;

/*****************************************************************************
This class was designed to override the methods of Default handler class and get attribute values into the
Vector defined in this class It is run from the command line as:java TicTacToeParser tboard1.xml.  Here
tboard1.xml is the filename of an XML documet.
*****************************************************************************/
 public class TicTacToeParser extends DefaultHandler
{

        /** Local variables to store the Attributes found in the XML document*/
        public  String  column1   = "";
        public  String  column2   = "";
        public  String  column3   = "";

        /** Vector to store the values of attributes*/
        public Vector values = new Vector();

        /** Override methods of the DefaultHandler class to gain notification of SAX Events. */
        /*****************************************************
        Receive notification of the beginning of the XML document.
         @exception SAXException.
        *****************************************************/
        public void startDocument( ) throws SAXException
        {
        }

        /*****************************************************
        Receive notification of the end of the document.
         @exception SAXException.
         *****************************************************/
        public void endDocument( ) throws SAXException
        {
        }
```

```
/*************************************************************************
Receives notification of the start of an element in the XML document and Attribute values in an
XML document.
@param namespaceURI specifies the namespace of an XML document.
@param localName specifies the local name of an XML document.
@param qName specifies the qualified name of an XMl document.
@param attr specifies Attributes of an XML document.
@exception SAXException.
*************************************************************************/
public void startElement( String namespaceURI,String localName,String qName,Attributes attr )
 throws SAXException
{

/** Adding attribute values to the Vector values*/
if(localName.equals("column1")||localName.equals("column2")||localName.equals("column3"))
        {
for ( int i = 0; i < attr.getLength(); i++ )
{
                        values.add(attr.getValue(i));
                }
        }

}

/*********************************************************
 Receives notification of the end of an element in the XML document.
 @param namespaceURI specifies the namespace of an XML document.
 @param localName specifies the local name of an XML document.
 @param qName specifies the qualified name of an XMl document.
 @exception SAXException.
 *********************************************************/
public void endElement( String namespaceURI,String localName,String qName ) throws
 SAXException
{

}

/*****************************************************************
Receive notification of character data inside an element in the XML document.
@param ch specifies an array of characters from the XML document.
@param start specifies the starting position in the char array.
@param length specifies number of characters reading from the char array.
@exception SAXException.
*****************************************************************/
public void characters( char[] ch, int start, int length )throws SAXException
{       try
        {
                OutputStreamWriter outw = new OutputStreamWriter(System.out);
                outw.write( ch, start,length );
                outw.flush();
        }
        catch (Exception e)
        {
                e.printStackTrace();
        }
}
```

```
        /*************************************************************
        Read the XML document using FileReader and parse it using the parser.
        @param argv a variable array of type String.
        *************************************************************/
         public static void main( String[] argv )
         {
                try
                {
                        // Create SAX 2 parser...
                        XMLReader xr =
                         XMLReaderFactory.createXMLReader(DEFAULT_SAX_DRIVER);
                        // Set the ContentHandler...
                        TicTacToeParser parser1 = new TicTacToeParser();
                        xr.setContentHandler( parser1 );
                        // Parse the file...
                        xr.parse( new InputSource(new FileReader( "tboard1.xml" )) );
                        TicTacToeBoard tboard =new TicTacToeBoard(parser1.values);
                        tboard.show();
                        System.out.println("Attribute Values : " + parser1.values);
                }
                catch ( Exception e )
                {
                        e.printStackTrace();
                }

         }
         private static final String DEFAULT_SAX_DRIVER = "org.apache.xerces.parsers.SAXParser";

}

/*************************************************************************
Class Name : TicTacToeBoard.java
Author    :    Padmini Paladugu
Purpose   :   This class is designed to display Tic-Tac-Toe Board.
 *************************************************************************/

/*imported Packages*/
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
import java.awt.*;

/** This class was designed to display TicTacToeBoard using JButtons*/
public class TicTacToeBoard extends JFrame
{
         /**Constructor initializes the TicTacToeBoard with given parameter.*/
         public TicTacToeBoard(Vector valueTypes)
         {
                this.getContentPane().setLayout(null);
                 String buttons = valueTypes.get(0).toString();
                System.out.println(buttons);
                int k=0;
                 int x=200;
                 int y=100;
```

```
                for(int i=0;i<3;i++)
                {
                        for(int j=0;j<3;j++)
                        {
                                JButton cij = new JButton(valueTypes.get(k++).toString());
                                cij.setBounds(x+j*51,y+i*51,50,50);
                                this.getContentPane().add(cij);
                        }
                }
                setupEventHandlers();
                setSize(500,500);
        }

        void setupEventHandlers()
        {
                addWindowListener(new WindowHandler());
        }
        public class WindowHandler extends WindowAdapter
        {
                public void windowClosing(WindowEvent e)
                {
                        System.exit(0);
                }
        }

        //main
        /*public static void main(String[] args)
        {
                lab2 f=new lab2(new Vector());
                f.setSize(500,500);
                f.show();
        }*/
}

<!-- tboard1.xml- an XML document was designed to describe
        the state of TicTacToe board. This document describes
        a standard 3X3 TicTacToe Board.
-->
   <?xml version="1.0" standalone="no"?>
   <!DOCTYPE board SYSTEM "tboard1.dtd">
   <board>
        <row1>
                <column1 value="O"/>
                <column2 value="O"/>
                <column3 value="O"/>
        </row1>
        <row2>
                <column1 value="X"/>
                <column2 value="X"/>
                <column3 value="X"/>
        </row2>
        <row3>
                <column1 value="B"/>
                <column2 value="B"/>
                <column3 value="B"/>
```

```
        </row3>
    </board>

    <!-- tboard1.dtd - an XML DTD was designed for
        validating the tboard1.xml document
    -->
    <!ELEMENT board (row1, row2, row3)>
    <!ELEMENT row1 (column1, column2, column3)>
    <!ELEMENT row2 (column1, column2, column3)>
    <!ELEMENT row3 (column1, column2, column3)>
    <!ELEMENT column1 EMPTY>
    <!ELEMENT column2 EMPTY>
    <!ELEMENT column3 EMPTY>
    <!ATTLIST column1 value (O | X | B) #REQUIRED>
    <!ATTLIST column2 value (O | X | B) #REQUIRED>
    <!ATTLIST column3 value (O | X | B) #REQUIRED>
```

Deliverable2

```
/*********************************************************************
Class Name: TicTacToeClient.java
Author:     Padmini Paladugu
Purpose:    This class is designed to play Tic-Tac-Toe Board. This
            class will send each player's move to the server and will
            get response back from the server.
*********************************************************************/

/*imported Packages*/
import java.util.*;
import javax.swing.*;
import java.awt.event.*;
import java.io.*;
import java.awt.*;
import java.net.*;

/** This class was designed to play a TicTacToe game.This will send
each player's move to the server using Stream Sockets. */
public class TicTacToeClient extends JFrame
{
        private static String host="localhost";
        private static String confirm="";
        private static Socket clientSocket=null;
        private static PrintWriter out=null;
        private static BufferedReader in=null;
        private static BufferedReader in1=null;
        int count=0;
        JButton[][]  buttonArray;
        String[] users;
        boolean gameOver;
        /**Constructor initializes the TicTacToeClient */
        public TicTacToeClient()
        {
                /** Establishing a Socket Connection*/
                try
                {
```

```
            clientSocket=new Socket(host,8088);
            System.out.println("This is connected to port # 8088 at "+
             clientSocket.getInetAddress());
            out=new PrintWriter(clientSocket.getOutputStream(),true);
            in=new BufferedReader(new
            InputStreamReader(clientSocket.getInputStream()));
          }
          catch(Exception uhe)
          {
           uhe.printStackTrace();
           System.exit(1);
          }
          this.getContentPane().setLayout(null);
          int x=200;
          int y=100;
          buttonArray=new JButton[3][3];
          users=new String[2];
          users[0]="X";
          users[1]="O";
          gameOver=false;
          /** Creating a User Interface with JButtons*/
          for(int i=0;i<3;i++)
          {
                    for(int j=0;j<3;j++)
                    {
                              buttonArray[i][j] = new JButton();
                              buttonArray[i][j].setBounds(x+j*51,y+i*51,50,50);
                              this.getContentPane().add(buttonArray[i][j]);

                              buttonArray[i][j].addActionListener(new ActionListener()
                              {
                                        public void actionPerformed(ActionEvent e)
                                        {
                                                  try
                                                  {
                                                            button_actionPerformed(e);
                                                  }
                                                  catch(Exception ewr)
                                                  {
                                                            ewr.printStackTrace();
                                                            System.exit(1);
                                                  }

                                        }

                              });
                    }
          }
          setupEventHandlers();
          setSize(500,500);
}

/*************************************************************
 This method will check for the proper move for each player and it will
 accept the move
**************************************************************/
```

```
public void button_actionPerformed(ActionEvent e)
{
        JButton button= (JButton)e.getSource();
        for(int i=0;i<3;i++)
        {
                for(int j=0;j<3;j++)
                {
                        if(buttonArray[i][j]==button)
                        {
                                if(!gameOver && getResponse(i,j,users[count%2]))
                                {
                                        if(button.getText().equals(users[0])
                                        ||button.getText().equals(users[1]))
                                        {
                                                JOptionPane.showMessageDialog(null,
                                                "oops!! Please click on different  cell"
                                                ,"Warning" ,
                                                JOptionPane.WARNING_MESSAGE);

                                        }
                                        else
                                        {
                                                button.setText(users[count%2]);
                                                isGameOver();
                                                count++;
                                        }
                                }

                        }

                }
        }
}

 /*****************************************************
 This TicTacToe Game will be over if any player's move equal
 either horizontally, vertically ,or diagonally. This method will
 check for the horizontal, vertical ,or diagonal validity
*****************************************************/
 void isGameOver()
 {
        for(int i=0;i<3;i++)
        {
                String user1 = buttonArray[i][0].getText();
                System.out.println("user" + user1);
                if(user1==null || (!user1.equals(users[0])&&!user1.equals(users[1])))
                        continue;
                for(int j=0;j<3;j++)
                {
                        String user2 = buttonArray[i][j].getText();
                        if(user2==null|| (!user2.equals(users[0])&&!user2.equals(users[1])))
                                break;
                        if(!user2.equals(user1))
                                break;
                        if(j==2)
                        {
```

```
                        System.out.println("GameOver");
                        JOptionPane.showMessageDialog(null,"hoooo!! Game Over",
                        "Message",JOptionPane.INFORMATION_MESSAGE);
                        gameOver=true;
                        closeConnection();
                        return;
                }
        }
}

for(int j=0;j<3;j++)
{
        String user1 = buttonArray[0][j].getText();
        if(user1==null|| (!user1.equals(users[0])&&!user1.equals(users[1])))
                continue;
        for(int i=0;i<3;i++)
        {
                String user2 = buttonArray[i][j].getText();
                if(user2==null||(!user2.equals(users[0])&&!user2.equals(users[1])))
                break;
                if(!user2.equals(user1))
                break;
                if(i==2)
                {
                        System.out.println("GameOver");
                        JOptionPane.showMessageDialog(null,"hoooo!! Game Over",
                        "Message",JOptionPane.INFORMATION_MESSAGE);
                        gameOver=true;
                        closeConnection();
                        return;
                }
        }
}
String user1=buttonArray[0][0].getText();
if(user1!=null&& (user1.equals(users[0])||user1.equals(users[1])))
{
        for(int i=0;i<3;i++)
        {
                String user2 = buttonArray[i][i].getText();
                if(user2==null|| (!user2.equals(users[0])&&!user2.equals(users[1])))
                break;
                if(!user2.equals(user1))
                break;
                if(i==2)
                {
                        System.out.println("GameOver");
                        JOptionPane.showMessageDialog(null,"hoooo!! Game Over",
                        "Message",JOptionPane.INFORMATION_MESSAGE);
                        gameOver=true;
                        closeConnection();
                        return;
                }
        }
}
if(buttonArray[0][2].getText()!=null && buttonArray[1][1].getText()!=null &&
buttonArray[2][0].getText()!=null)
```

```
		{
			if(buttonArray[0][2].getText().equals(buttonArray[1][1].getText())&&
			(buttonArray[0][2].getText().equals(users[0])||
			buttonArray[0][2].getText().equals(users[1]))&&
			buttonArray[1][1].getText().equals
			(buttonArray[2][0].getText()))
			{
				System.out.println("GameOver");
				JOptionPane.showMessageDialog(null,"hoooo!! Game Over",
				"Message",JOptionPane.INFORMATION_MESSAGE);
				gameOver=true;
				closeConnection();
				return;
			}
		}
}

void setupEventHandlers()
{
		addWindowListener(new WindowHandler());

}

/*****************************************************
This method will send the player's move to the server and will get the
response back from the server
*****************************************************/
boolean getResponse(int row,int column, String user)
{
		try
		{
		  out.println(user +"*" + (row+1) +"*"+(column+1));
		}
		catch(Exception ioe)
		{
		  ioe.printStackTrace();
		  System.exit(1);
		}
		try
		{
			confirm=in.readLine();
			System.out.println("returned value "+confirm);
		}
		catch(Exception ioe)
		{
		  ioe.printStackTrace();
		  System.exit(1);
		}

		return true;
}
public class WindowHandler extends WindowAdapter
{
		public void windowClosing(WindowEvent e)
		{
			System.exit(0);
```

```
                }
        }

   /*********************************************
  This method will close the connection for socket,inputstream,
  and outputstream
**************************************************/
        public static void closeConnection()
        {
                try
                {
                        out.close();
                        in.close();
                        clientSocket.close();
                }
                catch(Exception uhe)
                {
                        uhe.printStackTrace();
                        System.exit(1);
                }

        }
        //main
        public static void main(String[] args)
        {
                TicTacToeClient f=new TicTacToeClient();
                f.setSize(500,500);
                f.show();
        }
}

/*****************************************************************
Class Name : TicTacToeServerThread.java
Author     : Padmini Paladugu
Purpose    : This class will create a thread for each client, and
          it will serve multiple clients simultaneously.
 *****************************************************************/

/**Imported packages*/
import java.net.*;
import java.io.*;
import java.sql.*;
import java.util.*;
import javax.sql.*;
import java.math.*;

/*****************************************************************
This class will accept the request from the client and it will
decode the move into strings using the StringDecoder class.This class will establish a connection with the
database using JDBC and updates
the data in the XMLType table
*****************************************************************/
public class TicTacToeServerThread extends Thread
{
        private static Socket clientSocket = null;
        private static PrintWriter out=null;
```

```
private static BufferedReader in=null;
String moveNumber[]=new String[5];
public TicTacToeServerThread(Socket clientSocket1)
{
        clientSocket=clientSocket1;
}
public void run()
{
         runPlayer();
}

public synchronized void runPlayer()
{       try
        {
                out = new PrintWriter(
                clientSocket.getOutputStream(),true);
                in = new BufferedReader(
                new InputStreamReader
                (clientSocket.getInputStream()));
        }
        catch (IOException e)
        {
                e.printStackTrace();
                System.exit(1);
        }
        Connection connection=null;
        Statement statement=null;
        while(true)
        {
                try
                {
                        String move=in.readLine();
                        System.out.println("Line="+move);
                        /**Decoding the input
                        string using StringDecoder Class*/
                        StringDecoder sd=new StringDecoder();
                        moveNumber=sd.getMeStrings(move);
                        for(int i=0;i<3;i++)
                        {
                                System.out.println("Parsed String
                                = "+moveNumber[i]);
                        }
                        /** Esablishing a connection with JDBC
                        using OracleDriver*/
                        DriverManager.registerDriver(new
                        oracle.jdbc.OracleDriver());
                        String username="scott",
                             password="tiger",
                             dbURL="jdbc:oracle:oci:@pgdev";
                        connection=DriverManager.getConnection
                        (dbURL,username,password);
                        statement=connection.createStatement();
                        String sql="UPDATE ttboard t SET value(t)
                        = updateXML(value(t),'/board/row";
                        sql+=moveNumber[1];
                        sql+="/column";
```

```
                                        sql+=moveNumber[2];
                                        sql+="/text()='";
                                        sql+=moveNumber[0];
                                        sql+="')";
                                        System.out.println("Update
                                         Statement="+sql);
                                        statement.executeUpdate(sql);
                                        out.println("Good_Move");

                                }
                                catch (Exception eh)
                                {
                               /**Closing the connection with the
                               database*/
                                        closeConnection();
                                          try
                                          {
                                                  statement.close();
                                                  connection.close();
                                          }
                                          catch(Exception e)
                                          {
                                          }
                                        break;
                                }
                }//end while

        }//end runPlayer method

        /**Closing the connection with the socket ,inputstream,
        and output stream*/
        public static void closeConnection()
        {
                try
                {
                        out.close();
                        in.close();
                        clientSocket.close();
                }
                catch (IOException e)
                {

                }
        }

}//end class

/**********************************************************
Class Name : TicTacToeServer.java
Author     : Padmini Paladugu
Purpose    : This class will create a Server socket and
listens to the client's requests.
**********************************************************/
/**Imported packages*/
import java.net.*;
import java.io.*;
```

```
import java.sql.*;
import java.util.*;
public class TicTacToeServer
{
    public static void main(String args[])
    {
      TicTacToeServer ttserver=new TicTacToeServer();
      ttserver.startServer();
    }

    public static void startServer()
    {
                    Socket clientSocket = null;
                    ServerSocket serverSocket=null;
                    PrintWriter out=null;
                    BufferedReader in=null;

                    try
                    {
                            serverSocket = new ServerSocket(8088);
                    }
                    catch (IOException e)
                    {
                            e.printStackTrace();
                            System.exit(1);
                    }
                    while(true)
                    {
                            try
                            {
                                    System.out.println("Waiting for client
                                            request");
                                    clientSocket = serverSocket.accept();
                                    System.out.println("Processing");
                                    TicTacToeServerThread ttthread=
                                  new TicTacToeServerThread(clientSocket);
                                    ttthread.start();
                            }
                            catch (IOException e)
                            {
                                            e.printStackTrace();
                                    System.exit(1);

                            }


                    }//end while
    }

}



/*************************************************************
Class Name : StringDecoder.java
Author     : Padmini Paladugu
```

```
Purpose    : This class will parse the given string into tokens.
*****************************************************************/
import java.util.*;
public class StringDecoder
{
  String str[]=new String[20];
  int i=0;
  public String[] getMeStrings(String s)
  {
    StringTokenizer st = new StringTokenizer(s,"*");
    while (st.hasMoreTokens())
    {
       System.out.println("i value        "+i);
       try
       {
        str[i]=st.nextToken("*");
       }
       catch(NoSuchElementException nn){}
       i++;
    }
    return str;
  }
}
/************************************
```

SQL Commands

```
*************************************/
SQL>CREATE TABLE ttboard OF XMLType;
Table created.
SQL>create directory XMLDIR as '/padmini/xmlsamples';
Directory created.
SQL> grant read on directory xmldir to public with grant option;
Grant succeeded.
SQL> create or replace function getDocument(filename varchar2) return clob
  2     authid current_user is
  3  xbfile bfile;
  4  xclob clob;
  5  begin
  6    xbfile  := bfilename('XMLDIR',filename);
  7    dbms_lob.open(xbfile);
  8
  9    dbms_lob.createtemporary(xclob,TRUE,dbms_lob.session);
 10     dbms_lob.loadfromfile(xclob,xbfile, dbms_lob.getlength(xbfile));
 11     dbms_lob.close(xbfile);
 12     return xclob;
 13  end;
 14  /
Function created.
SQL> INSERT INTO ttboard VALUES(XMLTYPE(getDocument('board.xml')));
1 row created.

SQL> set long10000
SQL> select * from ttboard;
SYS_NC_ROWINFO$
--------------------------------------------
<board>
            <row1>
```

```
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row1>
        <row2>
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row2>
        <row3>
                <column1>B</column1>
                <column2>B</column2>
                <column3>B</column3>
        </row3>
</board>
```

Deliverable 3

```
/*****************************************************************************
 Project:  Deliverable3 for CS297
 File:     Del3.c
 Purpose:  main C file for Deliverable3
        Does all the event handling for the program
 Author:   Padmini Paladugu
*****************************************************************************/

#include <PalmOS.h>
#ifdef __GNUC__
        #include "callback.h"
#endif
#include "Del3Rsc.h"
#include "DataBaseManager.h"

/*-----------------------------------------------*/
static void SayLastName(UInt16 alertID)
/*
PURPOSE:To verify whether correct data is entered into the database.
checks for the lastname field when we click on the save button.
RECEIVES:Alert ID of the Alert box
RETURNS: nothing
REMARKS: NONE
*/
{
   FormType   *form = FrmGetActiveForm();
   FieldType  *field;
   MemHandle  h;
   field = FrmGetObjectPtr(form, FrmGetObjectIndex(form,LastNameField));
   h = FldGetTextHandle(field);
   if (h)
   {
      Char  *s;
      s = MemHandleLock((void *)h);
      if (*s != '\0')
      {
         FrmCustomAlert(alertID, s, NULL, NULL);
```

```
        }
        else
        {
            FrmCustomAlert(alertID, "whoever you are", NULL,NULL);
        }
        MemHandleUnlock((void *)h);
    }
    else
    {
        FrmCustomAlert(alertID, "whoever you are", NULL, NULL);
    }
}

/*-----------------------------------------------*/
static void CheckSave(Char *s)
/*
PURPOSE:Alert box to display the mapped field
RECEIVES:nothing
RETURNS: nothing
REMARKS: NONE
*/
{

        FrmCustomAlert(CheckAlert, s, NULL, NULL);

}
/*-----------------------------------------------*/
static void MapFirstName()
/*
PURPOSE:Map text field's fname to the personData Struct's fname
RECEIVES:nothing
RETURNS: nothing
REMARKS: NONE
*/
{
    FormType   *form = FrmGetActiveForm();
    FieldType  *field;
    MemHandle  h;

    field = FrmGetObjectPtr(form, FrmGetObjectIndex(form,FirstNameField));
    h = FldGetTextHandle(field);
    if (h)
    {
        Char  *s;

        s = MemHandleLock((void *)h);
        if (*s != '\0')
        {
            StrCopy(personData.fname, s);
            //CheckSave(personData.fname);
            //StrCopy(personData.fname, FldGetTextPtr(fname));
        }

    }

}
```

```
/*-----------------------------------------------*/
static void MapLastName()
/*
PURPOSE:Map text field's lname to the personData Struct's lname
RECEIVES:nothing
RETURNS: nothing
REMARKS: NONE
*/
{
   FormType   *form = FrmGetActiveForm();
   FieldType  *field;
   MemHandle  h;

   field = FrmGetObjectPtr(form, FrmGetObjectIndex(form,LastNameField));
   h = FldGetTextHandle(field);
   if (h)
   {
      Char  *s;

      s = MemHandleLock((void *)h);
      if (*s != '\0')
      {
         StrCopy(personData.lname, s);
         //StrCopy(personData.fname, FldGetTextPtr(fname));
      }

   }

}

/*-----------------------------------------------*/
static void MapPhoneNum()
/*
PURPOSE:Map text field's phone to the personData Struct's phone
RECEIVES:nothing
RETURNS: nothing
REMARKS: NONE
*/
{
   FormType   *form = FrmGetActiveForm();
   FieldType  *field;
   MemHandle  h;


   field = FrmGetObjectPtr(form, FrmGetObjectIndex(form,PhoneNumField));
   h = FldGetTextHandle(field);
   if (h)
   {
      Char  *s;

      s = MemHandleLock((void *)h);
      if (*s != '\0')
      {
         StrCopy(personData.phone, s);
         //CheckSave(personData.phone);
```

```
        //StrCopy(personData.fname, FldGetTextPtr(fname));
      }

   }

}

/*-------------------------------------------------*/
static Boolean MainFormHandleEvent(EventPtr event)
/*
PURPOSE:handle form events and button events in the MainForm
RECEIVES:the current event and process it if it is a frmOpenEvent
        or ctlSelectEvent
RETURNS: true if could process
REMARKS: NONE
*/
{
   Boolean  handled = false;
   FormType  *form;

   #ifdef __GNUC__
          CALLBACK_PROLOGUE;
          #endif


   switch (event->eType)
   {
      case frmOpenEvent:
      {
                                form = FrmGetActiveForm();
                                FrmDrawForm(form);
                                //RetrieveRecord();
                                FrmSetFocus(form, FrmGetObjectIndex(form,FirstNameField));
                                handled=true;
                        }
      break;

      case ctlSelectEvent:
         switch (event->data.ctlSelect.controlID)
         {
            case SaveButton:
               //SayLastName(SaveAlert);
               MapFirstName();
               MapLastName();
               MapPhoneNum();
               SaveRecord();
               FrmGotoForm(MainForm);

               handled = true;
               break;

            case ListButton:
                                                //FrmAlert(ViewListAlert);
                                                        FrmGotoForm(ListForm);
               handled = true;
```

```
                break;

            default:
                break;
        }
        break;

    /*case menuEvent:
        handled =
            MainMenuHandleEvent(event->data.menu.itemID);
        break;*/

    default:
        break;
  }

        #ifdef __GNUC__
        CALLBACK_EPILOGUE;
        #endif

  return handled;
}

/*-------------------------------------------------*/
static Boolean ListFormHandleEvent(EventPtr event)
/*
PURPOSE:Callback function used to handle events related to our List form
RECEIVES:the current event and process it if it is a frmOpenEvent
       or ctlSelectEvent
RETURNS: true if could process
REMARKS: NONE
*/
{
  Boolean  handled = false;
  FormType  *form = FrmGetActiveForm();
  ListType   *list = FrmGetObjectPtr(form, FrmGetObjectIndex(form, RecordsList));
        //UInt16 formID = FrmGetFormId(form);

  #ifdef __GNUC__
        CALLBACK_PROLOGUE;
        #endif


  switch (event->eType)
  {
    case frmOpenEvent:
    {
                        gNumRecords = DmNumRecords(gDetailsDB);
                        LstSetDrawFunction(list,(ListDrawDataFuncType *)&ListDrawRecordNames);
                        LstSetListChoices(list, NULL, gNumRecords);
                        LstSetSelection(list, -1);
                        FrmDrawForm(form);
                        //FrmSetFocus(form, FrmGetObjectIndex(form,RecordsList));
                        handled=true;

          }
```

```
      break;

      case ctlSelectEvent:
         switch (event->data.ctlSelect.controlID)
         {
            case OKButton:
            FrmGotoForm(MainForm);
            handled=true;

            default:
               break;
         }
         break;

      /*case menuEvent:
         handled =
            MainMenuHandleEvent(event->data.menu.itemID);
         break;*/

      default:
         break;
   }

          #ifdef __GNUC__
          CALLBACK_EPILOGUE;
          #endif

   return handled;
}
/*-------------------------------------------------*/
static Boolean ApplicationHandleEvent(EventPtr event)
/*
PURPOSE: called mainly to load forms. This method sets the callback for that form
RECEIVES: the current event and process it if it is a frmLoadEvent
RETURNS: true if could process
REMARKS: NONE
*/
{
   FormType  *form;
   UInt16    formID;
   Boolean   handled = false;


   if (event->eType == frmLoadEvent)
   {
      formID = event->data.frmLoad.formID;
      form = FrmInitForm(formID);
      FrmSetActiveForm(form);

      switch (formID)
      {
         case MainForm:
            FrmSetEventHandler(form, MainFormHandleEvent);
            break;
                              case ListForm:
                              //FrmAlert(WarningAlert);
```

```
                                        FrmSetEventHandler(form, ListFormHandleEvent);
                                        break;
            default:
               break;
         }
         handled = true;
      }

      return handled;
}

/*-------------------------------------------------*/
static Err StartApplication(void)
/*
PURPOSE: called when application first started
RECEIVES: nothing
RETURNS: nothing
REMARKS: NONE
*/
{
            Err error = errNone;
            error = InitDatabase();
            if(error != errNone)
                        return error;
            //FrmAlert(ConfirmationAlert);
   FrmGotoForm(MainForm);
   return error;
}

/*-------------------------------------------------*/
static void StopApplication(void)
/*
PURPOSE: called when application stopped (closes all open forms)
RECEIVES: nothing
RETURNS: nothing
REMARKS: NONE
*/
{
            CloseDatabase();
            FrmCloseAllForms();
}

/*-------------------------------------------------*/
static void EventLoop(void)
/*
PURPOSE: event loop for our app gets events and processes them
RECEIVES: nothing
RETURNS: nothing
REMARKS: NONE
*/
{
   EventType  event;
   UInt16     error;
   do
   {
      EvtGetEvent(&event, evtWaitForever);
```

```
      if (! SysHandleEvent(&event))
         if (! MenuHandleEvent(0, &event, &error))
            if (! ApplicationHandleEvent(&event))
               FrmDispatchEvent(&event);
   } while (event.eType != appStopEvent);
}

/*-------------------------------------------------*/
UInt32 PilotMain(UInt16 launchCode, MemPtr cmdPBP,UInt16 launchFlags)
/*
PURPOSE: main entry point into our application
RECEIVES:launchCode to say how app launched,
RETURNS: error type of how application end (Hopefully errNone)
REMARKS: NONE
*/
{
   Err  err;
   switch (launchCode)
   {
      case sysAppLaunchCmdNormalLaunch:
         if ((err = StartApplication()) == 0)
         {
            EventLoop();
            StopApplication();
         }
         break;

      default:
         break;
   }

   return err;
}

/*********************************************************************
 File:   Del3Rsc.h
 Project:Deliverable3 for CS297
 Purpose:contains all the resource information for Del3 Program
 Author: Padmini Paladugu
*********************************************************************/

// Main form
#define MainForm            1000
#define ListForm                  1001
//Fields
#define FirstNameField       2001
#define LastNameField        2002
#define PhoneNumField         2003
// Menubar
#define MainMenuBar          3000
// Menu commands
#define MainEditUndo         4000
#define MainEditCut          4001
#define MainEditCopy         4002
#define MainEditPaste        4003
#define MainEditSelectAll     4004
```

```
#define MainEditKeyboard      4006
#define MainEditGraffitiHelp  4007
// Alerts
#define SaveAlert             5000
#define CheckAlert            5001
#define ConfirmationAlert     5002
#define ViewListAlert         5003
#define WarningAlert          5004
//List
#define RecordsList           6300
//Buttons
#define SaveButton            7004
#define ListButton            7005
#define OKButton              7006

/*********************************************************
 File:    Del3.rcp
 Project: Deliverable3 for CS297
 Purpose: resource file for Del3 application
 Author:  Padmini Paladugu
*********************************************************/

#include "Del3Rsc.h"
APPLICATIONICONNAME ID 100 "Details"
APPLICATION         ID 1   "LFhe"
VERSION             ID 1   "1.0"

// icons for application
ICON      "largeicon.bmp"
SMALLICON "smallicon.bmp"

/*
PURPOSE: Main Form on which to enter person's details
REMARKS: NONE
*/
FORM ID MainForm 0 0 160 160
MENUID MainMenuBar
USABLE
BEGIN
   TITLE "Person's Details"
   LABEL "FName:" AUTOID AT (0 17) FONT 1
   LABEL "LName:" AUTOID AT (0 34) FONT 1
   LABEL "Phone:" AUTOID AT (0 51) FONT 1
   FIELD ID FirstNameField AT (40 17 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   FIELD ID LastNameField AT (40 34 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   FIELD ID PhoneNumField AT (40 51 120 15) UNDERLINED
      AUTOSHIFT MAXCHARS 40
   BUTTON "SAVE"   ID SaveButton AT (30 117 50 30)
   BUTTON "LIST" ID ListButton
      AT (PrevRight+5 PrevTop 50 30)
   GRAFFITISTATEINDICATOR AT (140 PrevTop)
END

/*
```

```
PURPOSE: Form used to show the list of records
REMARKS: NONE
*/
FORM ListForm AT (0 0 160 160)
BEGIN
  TITLE "Show List of Records"
  LIST "" RecordsList AT (CENTER 23 144 AUTO) FONT 0 VISIBLEITEMS 10
  BUTTON "OK" OKButton AT (60 PREVTOP+115 AUTO AUTO) FONT 0
END

MENU ID MainMenuBar
BEGIN
   PULLDOWN "Edit"
   BEGIN
      MENUITEM "Undo"       ID MainEditUndo          "U"
          MENUITEM "Cut"        ID MainEditCut           "X"
          MENUITEM "Copy"       ID MainEditCopy          "C"
          MENUITEM "Paste"      ID MainEditPaste         "P"
          MENUITEM "Select All" ID MainEditSelectAll     "S"

          MENUITEM "Keyboard"    ID MainEditKeyboard      "K"
          MENUITEM "Graffiti Help" ID MainEditGraffitiHelp "G"
   END

END

/*
PURPOSE: Alerts for Del3 application
REMARKS: NONE
*/
ALERT ID SaveAlert
INFORMATION
BEGIN
   TITLE   "Save Record"
   MESSAGE "Last Name, ^1."
   BUTTONS "OK"
END

ALERT ID CheckAlert
INFORMATION
BEGIN
   TITLE   "Check Mapped Fields"
   MESSAGE "Mapped field, ^1."
   BUTTONS "OK"
END

ALERT ID ConfirmationAlert
CONFIRMATION
BEGIN
   TITLE "Confirmation Alert"
   MESSAGE "Confirmation--Database Created"
   BUTTONS "OK"
END

ALERT ID ViewListAlert
INFORMATION
```

```
BEGIN
   TITLE "List Alert"
   MESSAGE "View List of Records"
   BUTTONS "OK"
END


ALERT ID WarningAlert
WARNING
BEGIN
   TITLE "Warning Alert"
   MESSAGE "Hello Possible Error"
   BUTTONS "Understand"
END
/*************************************************************
 File:     DataBaseManager.c
 Project:  Deliverable3 for CS297
 Purpose:  Does all the database management for the program
 Author:   Padmini Paladugu
 *************************************************************/
#include <PalmOS.h>
#include "callback.h"
#include "Del3Rsc.h"
#include "DataBaseManager.h"

/*-----------------------------------------------*/
Err InitDatabase(void)
/*
PURPOSE: called to set and initialize the database of the Del3 Program
RECEIVES: nothing
RETURNS: nothing
REMARKS: NONE
*/
{
        UInt32 totalBytes;
   UInt32 dataBytes;
        Err error;
   gDetailsDB = DmOpenDatabaseByTypeCreator(DBType, CreatorID, dmModeReadWrite);
   if(!gDetailsDB)
   {
        error = DmCreateDatabase(0, DBName, CreatorID, DBType, false);
      if(error != errNone)
        return error;
      gDetailsDB =DmOpenDatabaseByTypeCreator(DBType, CreatorID, dmModeReadWrite);
   }
   if(!gDetailsDB)
     return DmGetLastErr();
   DmOpenDatabaseInfo(gDetailsDB, &gDetailsID, NULL, NULL, 0, NULL);
   DmDatabaseSize(0, gDetailsID, &gNumRecords, &totalBytes, &dataBytes);
   return errNone;
}

/*-----------------------------------------------*/
void SaveRecord()
/*
PURPOSE: save record that was last entered into the database
```

```
RECEIVES:nothing
RETURNS: nothing
REMARKS: none
*/
{
   UInt16 index;
   MemHandle recordH;
   FieldData *recordP;
   index = DmFindSortPosition(gDetailsDB, &personData, NULL,
       (DmComparF*) &RecordCompare, 0);
   recordH = DmNewRecord(gDetailsDB, &index, sizeof(FieldData));
   recordP = MemHandleLock(recordH);
   DmWrite(recordP, 0, &personData, sizeof(FieldData));
   MemHandleUnlock(recordH);
   DmReleaseRecord(gDetailsDB, index, true);
}
/*-----------------------------------------------*/
void RetrieveRecord()
{
/*
PURPOSE: retrieve records from the database
RECEIVES:nothing
RETURNS: nothing
REMARKS: none
*/

          MemHandle RecordHandle;
          UInt16 index;
          UInt16 length;
   FieldData *temp;
   Char fullDetails[100];
   index = DmFindSortPosition(gDetailsDB, &personData, NULL,
       (DmComparF*) &RecordCompare, 0);
   gDetailsDB =DmOpenDatabaseByTypeCreator(DBType, CreatorID, dmModeReadOnly);

   if (gDetailsDB == 0)
          WinDrawChars("Error Opening FirstData", StrLen("Error Opening FirstData"),30, 60);
   else
   {
                    gNumRecords = DmNumRecords(gDetailsDB);
             for (index = 0; index < gNumRecords; index++)
             {
                              RecordHandle = DmQueryRecord(gDetailsDB, index);
                    temp = MemHandleLock(RecordHandle);
                    StrCopy(fullDetails,temp->fname);
                    StrCat(fullDetails," ");
                    StrCat(fullDetails,temp->lname);
                    StrCat(fullDetails," ");
          StrCat(fullDetails,temp->phone);
          StrCat(fullDetails,"\0");
          WinDrawChars(fullDetails, StrLen(fullDetails), 5, index * 10 + 70);
                    MemHandleUnlock(RecordHandle);

                  }
   }
```

```
}

/*-------------------------------------------------*/
void CloseDatabase()
/*
PURPOSE: close this application database
RECEIVES:Nothing
RETURNS: None
REMARKS: NONE
*/
{
   DmCloseDatabase(gDetailsDB);
}

/*-------------------------------------------------*/
Int16 RecordCompare(FieldData *r1, FieldData *r2, Int16 other,
  SortRecordInfoPtr rec1SortInfo, SortRecordInfoPtr rec2SortInfo,
  MemHandle appInfoH)
/*
PURPOSE: compares two Records r1 and r2 to see which is largest
       This is called back from DmFindSortPosition
RECEIVES:parameters listed above r1 and r2 are all that used to compare
RETURNS: None
REMARKS: NONE
*/
{
   return StrCompare(r1->fname, r2->fname);
}

/*-------------------------------------------------*/
void ListDrawRecordNames(UInt16 itemNum, RectangleType *bounds,Char **itemsText)
/*
PURPOSE: draw an item in the list
RECEIVES:itemNum - number of record to list, bounds where to draw
RETURNS: None
REMARKS: NONE
*/
{
   MemHandle recordH;
   FieldData *temp;
   Char fullDetails[44];
   #ifdef __GNUC__
          CALLBACK_PROLOGUE;
   #endif
          gDetailsDB =DmOpenDatabaseByTypeCreator(DBType, CreatorID, dmModeReadOnly);
          recordH = DmQueryRecord(gDetailsDB,itemNum);
          temp  = MemHandleLock(recordH);
          StrCopy(fullDetails,temp->fname);
          StrCat(fullDetails," ");
          StrCat(fullDetails,temp->lname);
          StrCat(fullDetails," ");
          StrCat(fullDetails,temp->phone);
          StrCat(fullDetails,"\0");
   WinDrawChars(fullDetails, StrLen(fullDetails), bounds->topLeft.x,bounds->topLeft.y);
   //WinDrawChars(temp->fname, StrLen(temp->fname), bounds->topLeft.x,bounds->topLeft.y);
   MemHandleUnlock(recordH);
```

```
        #ifdef __GNUC__
        CALLBACK_EPILOGUE;
        #endif
}

/***********************************************************************
 File:    DataBaseManager.h
 Project:  Deliverable3 for CS297
 Purpose:  Give global's, struct's, and prtotypes
        for the memory management subsytem of this Deliverable
 Author:   Padmini Paladugu
***********************************************************************/


#ifndef __DATAMANAGER_H__
#define __DATAMANAGER_H__

//Database
#define DBName "RecordDB-LF1b"
#define DBType 'DATA'

//Creator ID
#define CreatorID 'LF1b'

//Database variables
DmOpenRef gDetailsDB;
LocalID gDetailsID;
UInt32 gNumRecords;

//Struct to define fields
typedef struct
{
        Char fname[12];
        Char lname[12];
        Char phone[20];
}FieldData;

FieldData personData;

//Data Managing Prototypes
Err InitDatabase();
void CloseDatabase();
void SaveRecord();
void RetrieveRecord();

// Callback for comparing records
Int16 RecordCompare(FieldData *r1, FieldData *r2, Int16 other,
  SortRecordInfoPtr rec1SortInfo, SortRecordInfoPtr rec2SortInfo,
  MemHandle appInfoH);

// Callback for drawing list
void ListDrawRecordNames(UInt16 itemNum, RectangleType *bounds,
                Char **itemsText);

#endif
/************************
```

File:callback.h

```
Author:Padmini Paladugu
************************/
#ifndef __CALLBACK_H__
        #define __CALLBACK_H__
        /* This is a workaround for a bug in the current version of gcc:
   gcc assumes that no one will touch %a4 after it is set up in
   crt0.o.  This isn't true if a function is called as a callback
   by something that wasn't compiled by gcc (like
   FrmCloseAllForms()).  It may also not be true if it is used as
   a callback by something in a different shared library.
   We really want a function attribute "callback" which will
   insert this progloue and epilogoue automatically.
   - Ian */
   register void *reg_a4 asm("%a4");
   #define CALLBACK_PROLOGUE \
   void *save_a4 = reg_a4; \
   asm("move.l %%a5,%%a4; sub.l #edata,%%a4" : :);
   #define CALLBACK_EPILOGUE reg_a4 = save_a4;

#endif
/*************************************
Makefile
*************************************/
APP      = Del3
APPID    = LFhe
RCP      = Del3.rcp
PRC      = Del3.prc
SRC      = Del3.c

CC       = m68k-palmos-gcc
PILRC    = pilrc
OBJRES   = m68k-palmos-obj-res
BUILDPRC = build-prc

CFLAGS   = -g
all: $(PRC)
$(PRC): grc.stamp bin.stamp;
        $(BUILDPRC) -t appl -o $(PRC) -n "Hello" -c $(APPID) *.grc *.bin
        ls -l *.prc
grc.stamp: $(APP)
        $(OBJRES) $(APP)
        touch $@
$(APP): $(SRC)
        $(CC) $(CFLAGS) $(SRC) DataBaseManager.c -o $(APP)
bin.stamp: $(RCP)
        $(PILRC) $^ $(BINDIR)
        touch $@
%.o: %.c
        $(CC) $(CFLAGS) -c $< -o $@
depend dep:
        $(CC) -M $(SRC) > .dependencies
clean:
        rm -rf *.o $(APP) *.bin *.grc *.stamp *~
veryclean: clean
        rm -rf *.prc *.bak
```