5.12

Answer:

It prints out the manger of "dog", that manager's manager, etc.
until we reach a manager who has no manager.

5. 13

a. we need to know the number of attributes and names of attributes of r to decide the number and names of columns in the table.

b. We can use the JDBC methods getColumnCount() and getColumn-Name(int) to get the required information.

C. The method is showing as follows:

```
static void printTable(string r)
{
  try
  { Class.forName("oracle.jdbc.driver.OracleDriver");
    Connection conn = DriverManager.getConnection(
        "jdbc:oracle:thin:@db.yale.edu:2000:univdb",
        user, password);
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.ExecuteQuery(r);
    ResultSet rs = stmt.ExecuteQuery(r);
    ResultSetMetaData rsmd = rs.getMetaData();
    int count = rsmd.getColumnCount();
    System.out.println("<tr>");
    for (int i=1; i<=count; i++){
        System.out.println("<td>" + rsmd.getColumn
    }                              Name(i)+"</td>");
    System.out.println("</tr>");
    while (rs.next()) { system.out.println("<tr>");
    for (int i=1; i<=count; i++){System.out.println("<td>"+
                rs.getString(i)+"</td>");}
      System.out.println("</tr>");
  }
```

```
        Stmt.close();
        conn.close();
    }
    Catch ( SQLException sqle)
    {
        System.out.println("SQLException:" + sqle);
    }
}
```

5.14

a. same as 5.13

b. The function SQLNumResultCols(hstmt, &numColumn) can be used to find the number of columns in a statement, while the function SQLColAttribute() can be used to find the name, type and other information about any column of a result set. set, and the names.

c.

```
// SQLColAttibute.cpp
// compile with: user32.lib odbc32.lib

#define UNICODE

#include <windows.h>
#include <sqlext.h>
#include <strsafe.h>

struct DataBinding {
   SQLSMALLINT TargetType;
   SQLPOINTER TargetValuePtr;
   SQLINTEGER BufferLength;
   SQLLEN StrLen_or_Ind;
};

void printStatementResult(SQLHSTMT hstmt) {
   int bufferSize = 1024, i;
   SQLRETURN retCode;
   SQLSMALLINT numColumn = 0, bufferLenUsed;

   retCode = SQLNumResultCols(hstmt, &numColumn);

   SQLPOINTER* columnLabels = (SQLPOINTER *)malloc( numColumn *
sizeof(SQLPOINTER*) );
   struct DataBinding* columnData = (struct DataBinding*)malloc( numColumn *
sizeof(struct DataBinding) );

   printf( "Columns from that table:\n" );
   for ( i = 0 ; i < numColumn ; i++ ) {
      columnLabels[i] = (SQLPOINTER)malloc( bufferSize*sizeof(char) );

      retCode = SQLColAttribute(hstmt, (SQLUSMALLINT)i + 1, SQL_DESC_LABEL,
columnLabels[i], (SQLSMALLINT)bufferSize, &bufferLenUsed, NULL);
      wprintf( L"Column %d: %s\n", i, (wchar_t*)columnLabels[i] );
   }

   // allocate memory for the binding
   for ( i = 0 ; i < numColumn ; i++ ) {
      columnData[i].TargetType = SQL_C_CHAR;
      columnData[i].BufferLength = (bufferSize+1);
      columnData[i].TargetValuePtr = malloc( sizeof(unsigned
char)*columnData[i].BufferLength );
   }

   // setup the binding
   for ( i = 0 ; i < numColumn ; i++ ) {
      retCode = SQLBindCol(hstmt, (SQLUSMALLINT)i + 1, columnData[i].TargetType,

         columnData[i].TargetValuePtr, columnData[i].BufferLength, &
(columnData[i].StrLen_or_Ind));
   }
```

```
   printf( "Data from that table:\n" );
   // fetch the data and print out the data
   for ( retCode = SQLFetch(hstmt) ; retCode == SQL_SUCCESS || retCode ==
SQL_SUCCESS_WITH_INFO ; retCode = SQLFetch(hstmt) ) {
      int j;
      for ( j = 0 ; j < numColumn ; j++ )
         wprintf( L"%s: %hs\n", columnLabels[j], columnData[j].TargetValuePtr );

      printf( "\n" );
   }
   printf( "\n" );
}

int main() {
   int bufferSize = 1024, i, count = 1, numCols = 5;
   wchar_t firstTableName[1024], * dbName = (wchar_t *)malloc(
sizeof(wchar_t)*bufferSize ), * userName = (wchar_t *)malloc(
sizeof(wchar_t)*bufferSize );
   HWND desktopHandle = GetDesktopWindow();   // desktop's window handle
   SQLWCHAR connStrbuffer[1024];
   SQLSMALLINT connStrBufferLen, bufferLen;
   SQLRETURN retCode;

   SQLHENV henv = NULL;   // Environment
   SQLHDBC hdbc = NULL;   // Connection handle
   SQLHSTMT hstmt = NULL;   // Statement handle

   struct DataBinding* catalogResult = (struct DataBinding*) malloc( numCols *
sizeof(struct DataBinding) );
   SQLWCHAR* selectAllQuery = (SQLWCHAR *)malloc( sizeof(SQLWCHAR) * bufferSize
);

   // connect to database
   retCode = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &henv);
   retCode = SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (SQLCHAR *)
(void*)SQL_OV_ODBC3, -1);
   retCode = SQLAllocHandle(SQL_HANDLE_DBC, henv, &hdbc);
   retCode = SQLSetConnectAttr(hdbc, SQL_LOGIN_TIMEOUT, (SQLPOINTER)10, 0);
   retCode = SQLDriverConnect(hdbc, desktopHandle, L"Driver={SQL Server}",
SQL_NTS, connStrbuffer, 1025, &connStrBufferLen, SQL_DRIVER_PROMPT);
   retCode = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);

   // display the database information
   retCode = SQLGetInfo(hdbc, SQL_DATABASE_NAME, dbName,
(SQLSMALLINT)bufferSize, (SQLSMALLINT *)&bufferLen);
   retCode = SQLGetInfo(hdbc, SQL_USER_NAME, userName, (SQLSMALLINT)bufferSize,
&bufferLen);

   for ( i = 0 ; i < numCols ; i++ ) {
      catalogResult[i].TargetType = SQL_C_CHAR;
      catalogResult[i].BufferLength = (bufferSize + 1);
      catalogResult[i].TargetValuePtr = malloc( sizeof(unsigned
char)*catalogResult[i].BufferLength );
   }

   // Set up the binding. This can be used even if the statement is closed by
closeStatementHandle
   for ( i = 0 ; i < numCols ; i++ )
```

```
      retCode = SQLBindCol(hstmt, (SQLUSMALLINT)i + 1,
catalogResult[i].TargetType, catalogResult[i].TargetValuePtr,
catalogResult[i].BufferLength, &(catalogResult[i].StrLen_or_Ind));

   retCode = SQLTables( hstmt, (SQLWCHAR*)SQL_ALL_CATALOGS, SQL_NTS, L"",
SQL_NTS, L"", SQL_NTS, L"", SQL_NTS );
   retCode = SQLFreeStmt(hstmt, SQL_CLOSE);

   retCode = SQLTables( hstmt, dbName, SQL_NTS, userName, SQL_NTS, L"%",
SQL_NTS, L"TABLE", SQL_NTS );

   for ( retCode = SQLFetch(hstmt) ; retCode == SQL_SUCCESS || retCode ==
SQL_SUCCESS_WITH_INFO ; retCode = SQLFetch(hstmt), ++count )
      if ( count == 1 )
         StringCchPrintfW( firstTableName, 1024, L"%hs",
catalogResult[2].TargetValuePtr );
   retCode = SQLFreeStmt(hstmt, SQL_CLOSE);

   wprintf( L"Select all data from the first table (%s)\n", firstTableName );
   StringCchPrintfW( selectAllQuery, bufferSize, L"SELECT * FROM %s",
firstTableName );

   retCode = SQLExecDirect(hstmt, selectAllQuery, SQL_NTS);
   printStatementResult(hstmt);
}
```