Classic Approaches
The Preprocessor Approach (1 / 2)

A common C program line:   `#include <stdio.h>`

But that is not C; rather, it’s a . . .

The Preprocessor Approach (2 / 2)

Two varieties of embedded SQL:
Cursors

A Problem:
How many tuples will be produced by your query?

Preprocessor Examples

See the Sample Programs! (If available . . . )

Advantage:

Disadvantage(s):
The Library Approach

Advantage:

Disadvantage:

ODBC vs. JDBC

ODBC:

JDBC:
JDBC

Core capabilities:

Some related technologies:

Using JDBC (1 / 4)

1. Establish connection to a data source
Using JDBC (2 / 4)

(c) Connect to the DBMS

Connection dbConnect = DriverManager.getConnection ("jdbc:oracle:thin:@host.foo.bar.com:1234:oracle", "username", "password" );

Using JDBC (3 / 4)

2. Send SQL statements to that source

Create a Statement object:

    Statement stmt = dbConnect.createStatement();

Ask it to execute the SQL query:

    ResultSet answer = stmt.executeQuery ("SELECT sno, status FROM s");
Using JDBC (4 / 4)

3. Process returned results and messages

JDBC uses cursors, too, but the details are implicit.

Before the first read:

Then, fetch field values by type. E.g.:

Accessing MetaData with JDBC

First, get a \texttt{ResultSetMetaData} object by calling:

Then, fetch the metadata you want to see. E.g.:

To get result cardinality: