# CMSC 433 – Programming Language Technologies and Paradigms Spring 2007

Façade Pattern

Apr. 05, 2007

## What is it?

- Frequently, as your programs evolve and develop, they grow in complexity.
- Furthermore, there may be a number of complicated subsystems, each of which has its own complex interface.
- The Façade pattern allows you to simplify this complexity by providing a simplified interface to these subsystems.
- This simplification may in some cases reduce the flexibility of the underlying classes, but usually provides all the function needed for all but the most sophisticated users.
- These users can still, of course, access the underlying classes and methods.

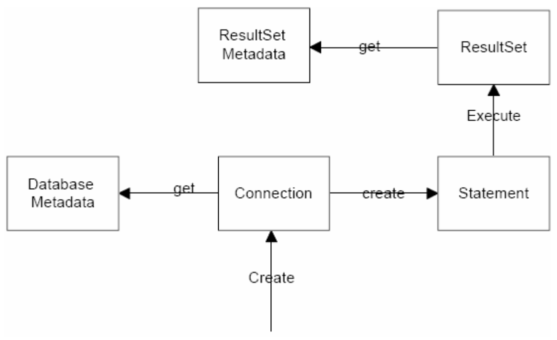
## For Example

- Java provides a set of classes that connect to databases using an interface called JDBC.
- You can connect to any database for which the manufacturer has provided a JDBC connection class -- almost every database on the market.
- Some databases have direct connections using JDBC and a few allow connection to ODBC driver using the JDBC-ODBC bridge class.

## For Example

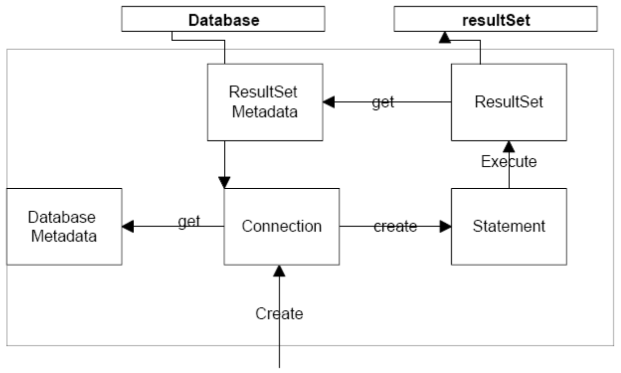
- To connect to a database, you use an instance of the Connection class.
- Then, to find out the names of the database tables and fields, you need to get an instance of the DatabaseMetadata class from the Connection.
- Next, to issue a query, you compose the SQL query string and use the Connection to create a Statement class.
- By executing the statement, you obtain a ResultSet class, and to find out the names of the column rows in that ResultSet, you need to obtain an instance of the ResultsetMetadata class.
- Thus, it can be quite difficult to juggle all of these classes and since most of the calls to their methods throw Exceptions, the coding can be messy.

## In Pictures!



## A Facade

- By designing a Façade consisting of a Database class and a resultSet class (note the lowercase “r”), we can build a much more usable system.



## Database Class

```
Class Database {
 public Database(String driver)()//constructor
 public void Open(String url, String cat);
 public String[] getTableNames();
 public String[] getColumnNames(String table);
 public String getColumnValue(String table,
             String columnName);
   public String getNextValue(String columnName);
   public resultSet Execute(String sql);
}
```

## resultSet

```
class resultSet
{
   public resultSet(ResultSet rset)      //constructor
    public String[] getMetaData();
    public boolean hasMoreElements();
    public String[] nextElement();
    public String getColumnValue(String columnName);
    public String getColumnValue(int i);
 }
```

## Summary

- The Façade pattern shields clients from complex subsystem components and provides a simpler programming interface for the general user.
- However, it does not prevent the advanced user from going to the deeper, more complex classes when necessary.