```
    catch (Exception e)
{System.out.println(e.getMessage());}
```

Then we use the Connection class to connect to a database. We also obtain the database metadata to find out more about the database.

```
try {
    con = DriverManager.getConnection(url);
    dma = con.getMetaData(); //get the meta data
}catch (Exception e) {
    System.out.println(e.getMessage());
}
```

If we want to list the names of the tables in the database, we then need to call the *getTables* method on the database metadata class, which returns a ResultSet object. Finally, to get the list of names, we must iterate through that object, making sure that we obtain only user table names, and exclude internal system tables.

```
public String[] get TableNames() {
        String[] tbnames = null;
        Vector tname = new Vector();
    //add the table names to a Vector
    //since we don't know how many there are
        try {
            results = new Results(dma.getTables(catalog,
                                    null, "%", types));
        } catch (Exception e) {
            System.out.println(e);
        }

            while (results.hasMoreElements())
            tname.addElement(results.getColumnValue
                                ("TABLE_NAME"));

        //copy the table names into a String array
        tbnames = new String[tname.size()];
        for (int i=0; i< tname.size(); i++)
            tbnames[i] = (String)tname.elementAt(i);
        return tbnames;
    }
```

This quickly becomes quite complex to manage, and we haven't even issued any queries.

We can make one simplifying assumption: The exceptions that all of these database class methods throw do not need complex handling. For the most part, the methods will work without error unless the network connection to the database fails. Thus we can wrap all of these methods in classes in which we simply print out the infrequent errors and take no further action.