

```
public Msqldriver() throws SQLException {  
    super();  
    DriverManager.registerDriver(this);  
}
```

Der Sun-ODBC- und ein Oracle8-Treiber würden so registriert werden:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
Class.forName("oracle.jdbc.driver.OracleDriver");
```

Neben der mehrfach überladenen Klassenmethode `getConnection()`, die im folgenden Abschnitt behandelt wird, gibt es im Treibermanager u.a. noch die Klassenmethoden

`Enumeration getDrivers()`

gibt die Liste aller registrierten Treiberobjekte zurück

`void deregisterDriver(Driver driver)`

entfernt das erste Treiberobjekt, das mit `driver` übereinstimmt

`void setLogStream(PrintStream out)`

Logbuch- und Protokollmeldungen werden auf `out` ausgegeben; z.B.

`DriverManager.setLogStream(System.out);` // schaltet ein

`DriverManager.setLogStream(null);` // schaltet aus

`void println(String message)`

Ausgabe einer Logbuchmeldung, z.B.

`DriverManager.println("Loading ...");`

Die Verwendung des Treibermanagers ist keineswegs zwingend. Ebenso gut können Treiberobjekte in eigene Verwaltung genommen werden. Für die obigen Beispiele ergeben sich dann als entsprechende Befehlszeilen

```
JdbcOdbcDriver dodbc = new JdbcOdbcDriver();  
Msqldriver dmsql = new Msqldriver();  
OracleDriver dorcl = new OracleDriver();
```

Eine Anmerkung: Im gleichen Verzeichnis, in dem die Kernklassen (`classes.zip`) lagern, kann in einem Verzeichnis mit dem Namen `.hotjava` eine Datei `properties` eingerichtet werden. In dieser Datei können Treiber persistent registriert werden durch Einträge der Art