Using a Database with JDBC

How to access a database from within Java using JDBC.
JDBC provides low-level access to a database.

For many applications, using *Object-Relational Mapping* (ORM) is much productive.

A Tiny Example

- A Todo database containing tasks to do.
- What's a Todo?
- "id" uniquely identifies the Todo and will be the *primary key* in the TODO table.

```
        Todo
id: int
title: String
done: boolean
```

What You Need

1. Database software
2. A JDBC database "driver" for your database
3. A database or at least a place to create a database

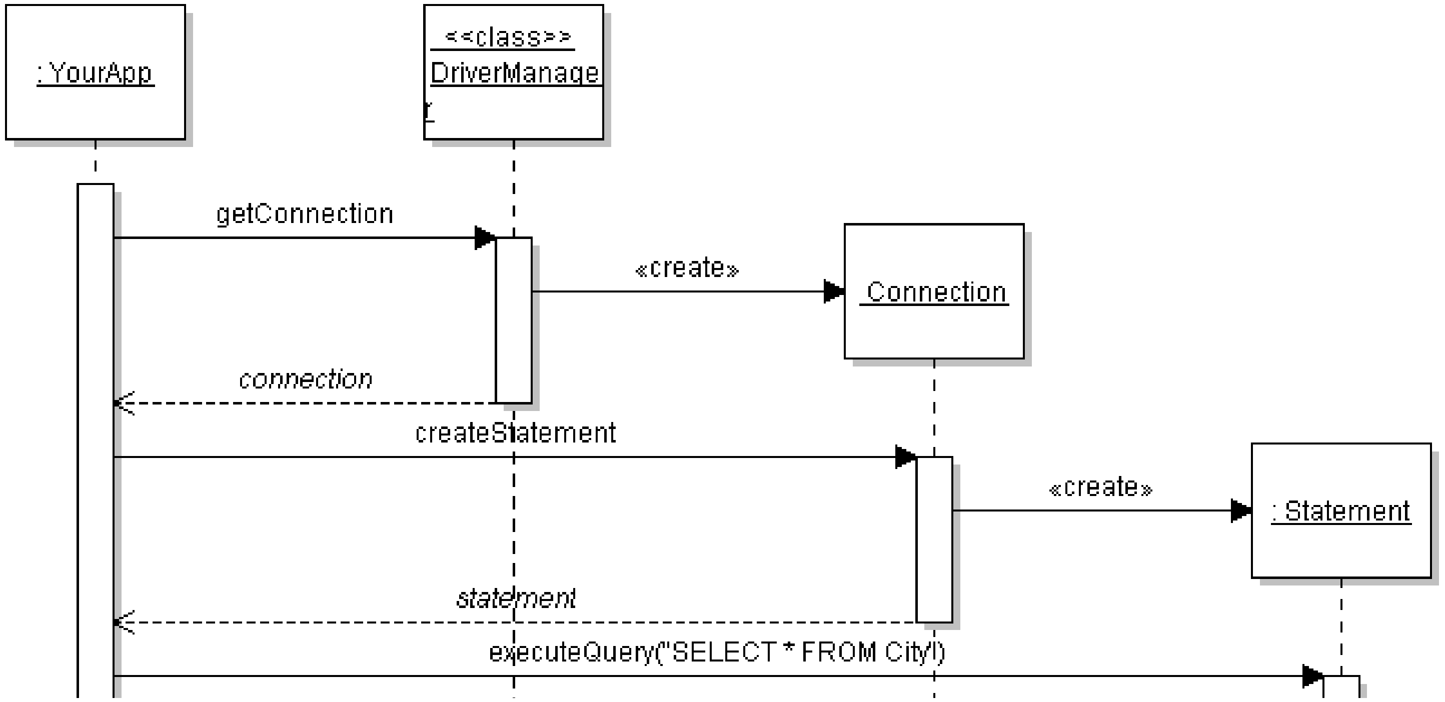
Accessing a Database using JDBC

Java has a standard programming interface for accessing databases, called the *Java DataBase Connectivity* (JDBC) API*.*

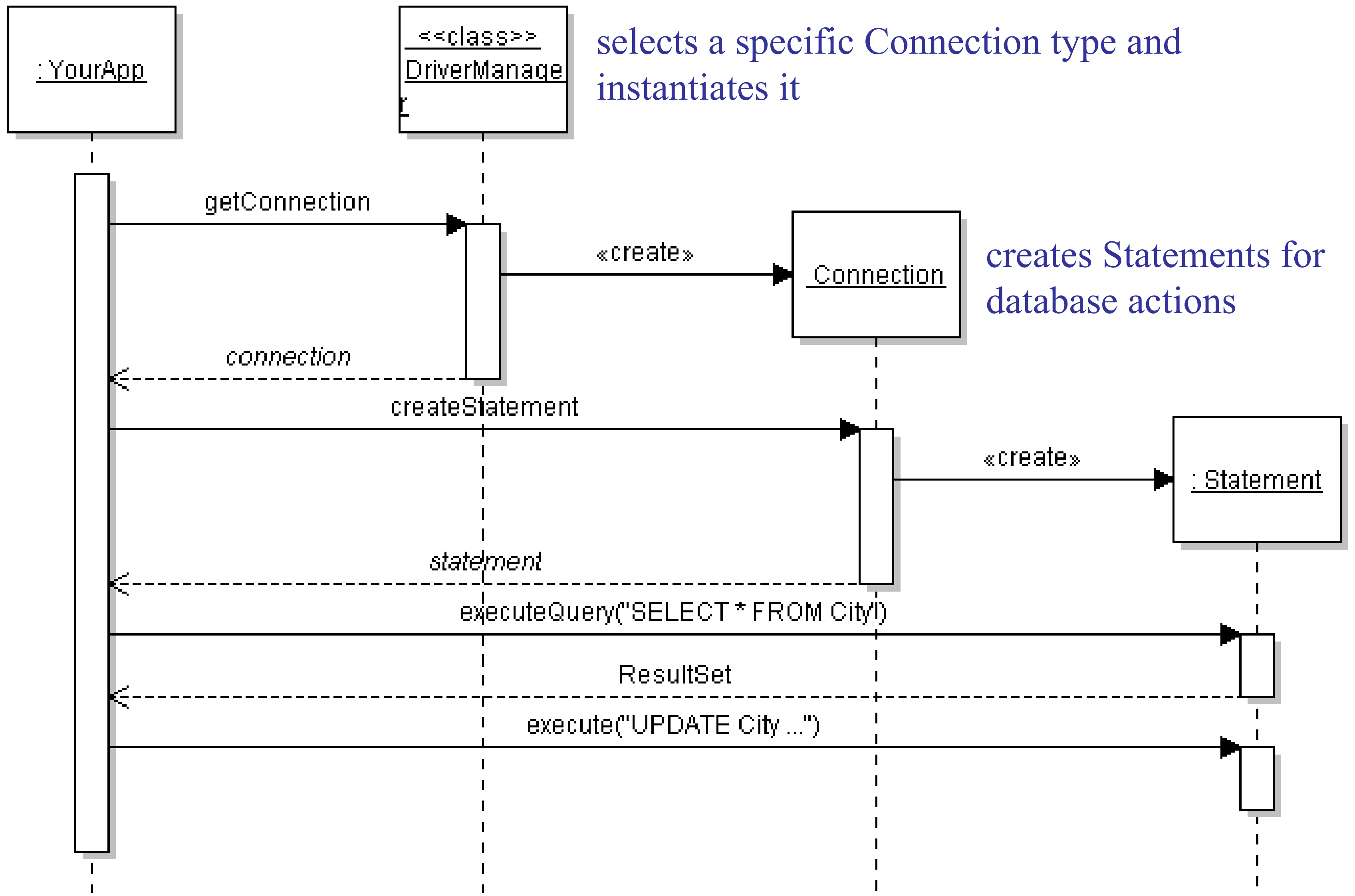
- JDBC is included in the Java JDK.
- Can connect to almost any database. All you need is a *JDBC driver* for your database.
- JDBC is very *low level* and we won't cover it in detail.
- But, to understand the mechanism we'll look at one brief example: saving and retrieving Todos.

JDBC Overview

1. Create a Connection to the database.
2. Create a *Statement* using the Connection.
3. Use the *Statement* to execute SQL commands.
4. Use the results.



JDBC Overview



selects a specific Connection type and instantiates it

creates Statements for database actions

JDBC Code

```
static final String URL = "jdbc:mysql://dbserver/tododb";
static final String USER = "student";
static final String PASSWORD = "secret";
// 1.  Get a Connection to the database.
Connection connection =
     DriverManager.getConnection( URL, USER, PASSWORD );

// 2.  Create a Statement

Statement statement = connection.createStatement();

// 3.  Execute the Statement with SQL command.

ResultSet rs = statement.executeQuery("SELECT * FROM todo");

// 4. Use the Result.

while ( rs.next( )  )  {

   String name = rs.getString("title");
```

Connecting to a Database in Java (1)

`java.sql.Connection` is a standard interface for connecting to any database.

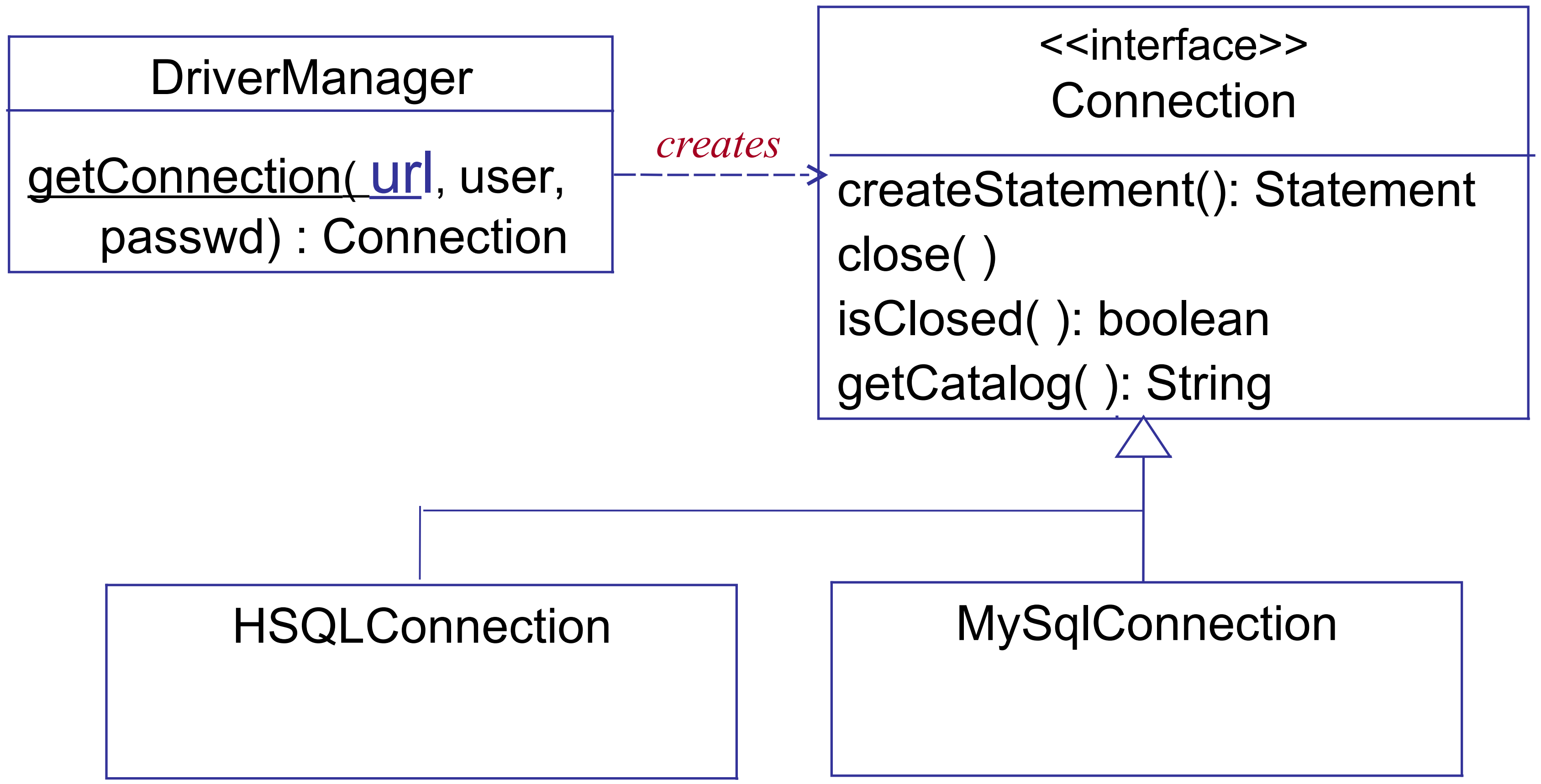
Each database type requires its **own jdbc driver** that *implements* this interface.

`DriverManager` selects the driver based on URL.

Database	JDBC Driver
MySQL	mysql-connector-java-5.1.7-bin.jar
Derby	derby.jar
Hypersonic SQL (HSQLDB)	hsqldb.jar

DriverManager returns a Connection

url = "jdbc:mysql://hostname/database"



Database URL

The format of a database URL is:

```
String DB_URL = "jdbc:mysql://localhost:3306/tododb";
```

Protocol Sub-protocol Hostname Port DatabaseName

Port is the TCP port number where the database server is listening.

- 3306 is the **default port** for MySQL

Use hostname or `"localhost"` for the local machine.

Database URL

The hostname and port are optional.

For MySQL driver: defaults are localhost and port 3306

Example: These 3 URL refer to the same database

```
"jdbc:mysql://localhost:3306/tododb"
"jdbc:mysql://localhost/tododb"
"jdbc:mysql:///tododb"
```

SQL to save data

To save a "todo" using SQL we would write:

```
sql> INSERT INTO todo(title, done)
           VALUES('Learn JDBC',false);
OK, 1 record added.
```

We didn't specify an ID *because the database is configured to assign the ID automatically.*

id	title	done
1	Go to OOP class	true
11	Learn JDBC	false

JDBC code to save data

To save a "todo" using JDBC in Java:

```
String sql = "INSERT INTO todo(title,done)
          VALUES('Learn JDBC',false)";
// this code may throw SQLException
Statement stmt =
         connection.createStatement();
int count = stmt.executeUpdate( sql );

System.out.printf("Added %d record", count);
```

SQL to retrieve data

To retrieve *all* the Todo in the table we would write:

```
sql> SELECT * FROM todo;
| id    | title                      | done   |
+-------+----------------------------+--------+
| 1     | Go to OOP class            | true   |
| 11    | Learn JDBC                 | false  |
  ...
```

JDBC code to retrieve data

```
String sql = "SELECT * FROM todo";
// this code may throw SQLException
Statement stmt =
         connection.createStatement();
ResultSet rs = stmt.executeQuery( sql );

// print the data
while( rs.next( ) ) {
  int id = rs.getInt("id");
  String title = rs.getString("title");
  boolean done = rs.getBoolean("done");
  System.out.printf("%d: %s (%s)\n",
      id, title, done );
}
```