

TASK 2 (Select)

```
1. package com.JDBC;
2. import java.util.*;
3. import java.sql.*;
4.
5. /** Select produces SQL SELECT statements.
6. * <p>Most of the clauses of a SELECT statement are supported,
7. * including SELECT DISTINCT, FROM, WHERE, ORDER BY.
8. * The WHERE subclauses can use either columns or values as
9. * the right hand side of expressions.
10. */
11. public class Select
12. {
13.     /** indicates if this select should use DISTINCT keyword */
14.     public final static boolean DISTINCT = true;
15.
16.     /** for ORDER BY clause */
17.     public final static int DESC = 1;
18.
19.     /** ANSI SQL standard operators and keywords */
20.     public final static String AND = " AND ";
21.     public final static String OR = " OR ";
22.
23.     private final static String __DISTINCT = " DISTINCT ";
24.     private final static String __DOT = ".";
25.     public final static String __SEPARATOR = " , ";
26.
27.     public Vector resultTargets = new Vector();
28.     private Vector selectClauses = new Vector();
29.     private Vector fromClauses = new Vector();
30.     private Vector orderByClauses = new Vector();
31.     boolean isDistinct = false;
32.     boolean isDescending = false;
33.
34.     /** Constructor.
35.     */
36.     public Select () {
37.         this (!DISTINCT);
38.     }
39.
40.     /** Add a subclause to the SELECT clause
41.     * @param table
42.     * @param col
43.     */
44.     public Select select (Table table, Column col) {
45.         String target = null;
46.         target = qualifyName (table, col);
47.
48.         // add it into our list of subclauses for the select
49.         this.selectClauses.addElement (target);
50.         this.resultTargets.addElement (col);
51.         return this;
52.     }
53.
54.     /** Add a table name to the FROM clause
55.     */
56.     public Select from (Table table) {
57.         this.fromClauses.addElement (table.getName());
58.         return this;
59.     }
60.
61.     /** Specify a column whose values should select the order of the
62.     * results.
63.     * @param table
64.     * @param column
65.     */
66.     public Select orderBy (Table table, Column column) {
67.         this.orderByClauses.addElement(qualifyName(table, column));
68.         return this;
69.     }
70.
71.     /** Build the query out of all of the clauses that have
72.     * been established, and do it against the database
73.     * @param db
74.     * @return
75.     * @exception java.sql.SQLException
76.     */
77.     public Enumeration query(Database db)
78.     throws java.sql.SQLException {
79.         String thequery = toString();
80.
81.         // Get the answer
```

```

82.         java.sql.ResultSet rs = db.query(thequery);
83.
84.         return new Enumerator(rs);
85.     }
86.
87.     /**
88.     * @return - the query as a string
89.     */
90.     public String toString() {
91.         StringBuffer thequery = new StringBuffer();
92.         Enumeration clauses = null;
93.
94.         thequery.append ("SELECT ");
95.         if (this.isDistinct)
96.             thequery.append ( __DISTINCT);
97.
98.         clauses = this.selectClauses.elements();
99.         while (clauses.hasMoreElements()) {
100.            thequery.append ((String)clauses.nextElement());
101.            if (clauses.hasMoreElements())
102.                thequery.append ( __SEPARATOR);
103.        }
104.
105.        thequery.append (" FROM ");
106.        clauses = this.fromClauses.elements();
107.        while (clauses.hasMoreElements()) {
108.            thequery.append ((String)clauses.nextElement());
109.            if (clauses.hasMoreElements())
110.                thequery.append(__SEPARATOR);
111.        }
112.
113.        String where = super.toString();
114.        if ((where != null) && (!where.equals(""))) {
115.            thequery.append (" WHERE ");
116.            thequery.append (where);
117.        }
118.
119.        clauses = this.orderByClauses.elements();
120.        if (clauses.hasMoreElements())
121.            thequery.append (" ORDER BY ");
122.
123.        while (clauses.hasMoreElements()) {
124.            thequery.append ((String)clauses.nextElement());
125.            if (clauses.hasMoreElements())
126.                thequery.append(__SEPARATOR);
127.        }
128.        if (this.isDescending)
129.            thequery.append(" DESC ");
130.
131.        return thequery.toString();
132.    }
133.
134.    /** Build a fully qualified name of the form:
135.    * table.column
136.    * @param table
137.    * @param column
138.    * @return
139.    */
140.    public static String qualifyName (Table table, Column column) {
141.        String name = null;
142.        name = table.getName() + __DOT + column.getName();
143.        return name;
144.    }
145. }

```

```

-----
/** This adapter class is defined as part of DBQuery, because
 * it needs to access the ResultSet of the query method.
 */
class Enumerator implements java.util.Enumeration {
}

```