```java
package model.dao;

import java.sql.Connection;

public abstract class DAO<T> {
    protected String TABLE_NAME;
    protected static Connection conn;

    public DAO() throws Exception {
        this("");
    }

    public DAO(String tableName) throws Exception {
        conn = ConnectionFactory.getConn();
        setTableName(tableName);
    }

    public void setTableName(String name) {
        TABLE_NAME = name;
    }

    public String getTableName() {
        return TABLE_NAME;
    }

    public String getAllQuery() {
        return String.format("select * from %s", getTableName());
    }

    public List<T> getAll() throws Exception {
        return getAll(0);
    }

    public List<T> getAll(Integer limit) throws Exception {
        Connection conn = ConnectionFactory.getConn();
        List<T> result = new ArrayList<T>();
        String limitQuery = "";
//      String limitQuery = (limit != null && limit > 0) ? "limit " + limit : "";
        String query = String.format("%s %s", getAllQuery(), limitQuery);

        PreparedStatement stmt = conn.prepareStatement(query);
        ResultSet r = stmt.executeQuery();
```

```
49
50          while (r.next()) {
51              T i = createBean(r);
52              if (!result.contains(i)) {
53                  result.add(i);
54              }
55          }
56
57          ConnectionFactory.closeConn(r, stmt, conn);
58          return result;
59      }
60
61      public List<T> getAllBy(String by, String val, Boolean like)
    throws Exception {
62          return getAllBy(by, val, like, 0);
63      }
64
65      public List<T> getAllBy(String by, String val, Boolean like,
    Integer limit) throws Exception {
66          Connection conn = ConnectionFactory.getConn();
67          List<T> result = new ArrayList<T>();
68          String limitQuery = "";
69  //      String limitQuery = (limit != null && limit > 0) ? "limit
    " + limit : "";
70          String equal = like ? "like" : "=";
71
72          val = like ? "'%" + val + "%'" : val;
73          by = this.convertToColumnName(by);
74
75          for (String[] str: getColumns().values()) {
76              if (str[0] == by) {
77                  if (str[1] == "str") {
78                      if (val.charAt(0) != '\'' &&
    val.charAt(val.length()-1) != '\'') {
79                          val = "'" + val + "'";
80                      }
81                  }
82              }
83          }
84
85          String query = String.format("%s where %s %s %s %s",
    getAllQuery(), by, equal, val, limitQuery);
86
87          PreparedStatement stmt = conn.prepareStatement(query);
```

```
88          ResultSet r = stmt.executeQuery();
89
90          while (r.next())
91          {
92              T i = createBean(r);
93              if (!result.contains(i)) {
94                  result.add(i);
95              }
96          }
97          ConnectionFactory.closeConn(r, stmt, conn);
98          return result;
99      }
100
101     public List<T> getAllByMultiple(String val, Boolean like,
    Integer limit) throws Exception {
102         Connection conn = ConnectionFactory.getConn();
103         List<T> result = new ArrayList<T>();
104         String limitQuery = "";
105 //      String limitQuery = (limit != null && limit > 0) ? "limit
    " + limit : "";
106         String equal = like ? "like" : "=";
107
108         val = like ? "'%" + val + "%'" : val;
109
110         String whereQuery = "";
111
112         int idx=0;
113         for (String[] str: getSearchColumns()) {
114             String or = "OR";
115             if (str[1] == "str") {
116                 if (val.charAt(0) != '\'' &&
    val.charAt(val.length()-1) != '\'') {
117                     val = "'" + val + "'";
118                 }
119             }
120
121             if (idx == getSearchColumns().size()-1) {
122                 or = "";
123             }
124
125             whereQuery += String.format(" %s %s %s %s ", str[0],
    equal, val, or);
126             idx++;
127         }
```

```
128
129         String query = String.format("%s where %s %s",
    getAllQuery(), whereQuery, limitQuery);
130         PreparedStatement stmt = conn.prepareStatement(query);
131         ResultSet r = stmt.executeQuery();
132 
133         while (r.next())
134         {
135             T i = createBean(r);
136             if (!result.contains(i)) {
137                 result.add(i);
138             }
139         }
140         ConnectionFactory.closeConn(r, stmt, conn);
141         return result;
142     }
143 
144     public T findOne(String val) throws Exception {
145         return this.findOneBy("id", val, false);
146     }
147 
148     public T findOneBy(String by, String val, Boolean like) throws
    Exception {
149         return this.findOneBy(by, val, like, 0);
150     }
151 
152     public T findOneBy(String by, String val, Boolean like,
    Integer limit) throws Exception {
153         return this.getAllBy(by, val, like, limit).get(0);
154     }
155 
156     public abstract Map<String, String[]> getColumns();
157     public abstract String convertToColumnName(String by);
158     public abstract List<String[]> getSearchColumns();
159     public abstract T createBean(ResultSet r) throws Exception;
160 }
161 
```