Listing 1: Student.java

```java
package exp6;

import java.io.Serializable;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 * 学生类
 *
 * @version 2015-6-10
 * @author Kingfree
 */
public class Student implements Serializable {

    private static final long serialVersionUID = 1L;

    private int id;
    private String name;
    private int os, math, java;
    private int classId;

    public Student() {

    }

    public Student(int id, int classId, String name, int os, int math, int java) {
        setId(id);
        setClassId(classId);
        setName(name);
        setOs(os);
        setMath(math);
        setJava(java);
    }

    private static int parseScore(int score) {
        return Math.abs(score) % 101;
    }

    public int getOs() {
        return os;
    }

    public void setOs(int os) {
        this.os = parseScore(os);
    }

    public int getMath() {
        return math;
    }

```

```
54      public void setMath(int math) {
55          this.math = parseScore(math);
56      }
57
58      public int getJava() {
59          return java;
60      }
61
62      public void setJava(int java) {
63          this.java = parseScore(java);
64      }
65
66      private void setId(int id) {
67          this.id = id;
68      }
69
70      private void setName(String name) {
71          this.name = name;
72      }
73
74      public int getId() {
75          return id;
76      }
77
78      public String getName() {
79          return name;
80      }
81
82      @Override
83      public String toString() {
84          return getName() + " " + getId() + " os:" + getOs() + " math:"
85                  + getMath() + " java:" + getJava();
86      }
87
88      private static Connection conn = null;
89
90      static {
91          try {
92              conn = DBUtils.openConnection();
93
94              List<String> sql = new ArrayList<String>();
95
96              // sql.add("DROP TABLE IF EXISTS students");
97              sql.add("CREATE TABLE IF NOT EXISTS students(id INTEGER PRIMARY KEY, c
98                      + ", name STRING, os INTEGER, math INTEGER, java INTEGER)");
99
100             // sql.add("INSERT INTO students VALUES(1, 1, '水树', 90, 90, 90)");
101             // sql.add("INSERT INTO students VALUES(2, 1, '田村', 89, 72, 88)");
102             // sql.add("INSERT INTO students VALUES(3, 2, '堀江', 80, 70, 91)");
103             // sql.add("INSERT INTO students VALUES(4, 3, '小仓', 78, 67, 70)");
104             // sql.add("INSERT INTO students VALUES(5, 5, '东山', 60, 88, 70)");
105             // sql.add("INSERT INTO students VALUES(6, 5, '种田', 77, 89, 70)");
106             // sql.add("INSERT INTO students VALUES(7, 4, '钉宫', 98, 78, 83)");
107             // sql.add("INSERT INTO students VALUES(8, 4, '喜多村', 84, 67, 71)");
108             // sql.add("INSERT INTO students VALUES(9, 4, '阿澄', 87, 68, 92)");
```

```
109             // sql.add("INSERT INTO students VALUES(10, 4, '花泽', 69, 67, 82)");
110
111             DBUtils.executeAsBatch(conn, sql);
112         } catch (SQLException e) {
113         }
114     }
115
116     public static int getMaxId() {
117         String sql = "SELECT MAX(id) FROM students";
118         try {
119             int id = DBUtils.queryObject(conn, sql, Integer.class);
120             return id;
121         } catch (Exception e) {
122             return 0;
123         }
124     }
125
126     public static List<Student> selectAll() {
127         String sql = "SELECT * FROM students";
128         try {
129             List<Student> you = DBUtils.queryBeanList(conn, sql,
130                     new StudentIResultSetCall());
131             return you;
132         } catch (Exception e) {
133             return null;
134         }
135     }
136
137     public static List<Student> selectByClass(Classe cls) {
138         String sql = "SELECT * FROM students WHERE class_id = ?";
139         try {
140             List<Student> you = DBUtils.queryBeanList(conn, sql, Student.class,
141                     cls.getId());
142             return you;
143         } catch (Exception e) {
144             return null;
145         }
146     }
147
148     public static Student selectById(int id) {
149         String sql = "SELECT * FROM students WHERE id = ?";
150         try {
151             Student you = DBUtils.queryBean(conn, sql, Student.class, id);
152             return you;
153         } catch (Exception e) {
154             return null;
155         }
156     }
157
158     public static List<Student> selectByName(String name) {
159         String sql = "SELECT * FROM students WHERE name LIKE ?";
160         try {
161             List<Student> you = DBUtils.queryBeanList(conn, sql,
162                     new StudentIResultSetCall(), "%" + name + "%");
163             return you;
```

```
164         } catch (Exception e) {
165             return null;
166         }
167     }
168 
169     public static boolean deleteById(int id) {
170         String sql = "DELETE FROM students WHERE id = ?";
171         try {
172             int res = DBUtils.execute(conn, sql, id);
173             if (res == 1) {
174                 return true;
175             } else {
176                 return false;
177             }
178         } catch (Exception e) {
179             return false;
180         }
181     }
182 
183     public static List<Student> selectOrberBy(String col) {
184         String sql = new String("SELECT * FROM students ORDER BY `" + col
185                 + "` DESC");
186         try {
187             List<Student> you = DBUtils.queryBeanList(conn, sql,
188                     new StudentIResultSetCall());
189             return you;
190         } catch (Exception e) {
191             return null;
192         }
193     }
194 
195     public static boolean insert(Student stu) {
196         String sql = "INSERT INTO students (id, class_id, name, os, math, java) VA
197         try {
198             int res = DBUtils.execute(conn, sql, stu.getId(), stu.getClassId(),
199                     stu.getName(), stu.getOs(), stu.getMath(), stu.getJava());
200             if (res == 1) {
201                 return true;
202             } else {
203                 return false;
204             }
205         } catch (Exception e) {
206             return false;
207         }
208     }
209 
210     public int getClassId() {
211         return classId;
212     }
213 
214     public void setClassId(int classId) {
215         this.classId = classId;
216     }
217 }
218 
```

```
219 class StudentIResultSetCall implements IResultSetCall<Student> {
220     public Student invoke(ResultSet rs) throws SQLException {
221         Student e = new Student(rs.getInt("id"), rs.getInt("class_id"),
222                 rs.getString("name"), rs.getInt("os"), rs.getInt("math"),
223                 rs.getInt("java"));
224         return e;
225     }
226 }
```

Listing 2: `Classe.java`

```
1  package exp6;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.util.ArrayList;
7  import java.util.List;
8
9  public class Classe {
10
11     private int id;
12     private String name;
13
14     public Classe() {
15     }
16
17     @Override
18     public String toString() {
19         return name;
20     }
21
22     public Classe(int id, String name) {
23         super();
24         this.id = id;
25         this.name = name;
26     }
27
28     public int getId() {
29         return id;
30     }
31
32     public void setId(int id) {
33         this.id = id;
34     }
35
36     public String getName() {
37         return name;
38     }
39
40     public void setName(String name) {
41         this.name = name;
42     }
43
44     private static Connection conn = null;
```

```
45
46      static {
47          try {
48              conn = DBUtils.openConnection();
49
50              List<String> sql = new ArrayList<String>();
51
52              // sql.add("DROP TABLE IF EXISTS classes");
53              sql.add("CREATE TABLE IF NOT EXISTS classes(id INTEGER PRIMARY KEY, na
54
55              // sql.add("INSERT INTO classes VALUES(1, '软件1301')");
56              // sql.add("INSERT INTO classes VALUES(2, '软件1302')");
57              // sql.add("INSERT INTO classes VALUES(3, '软件1303')");
58              // sql.add("INSERT INTO classes VALUES(4, '软件1304')");
59              // sql.add("INSERT INTO classes VALUES(5, '软件1305')");
60
61              DBUtils.executeAsBatch(conn, sql);
62          } catch (SQLException e) {
63          }
64      }
65
66      public static List<Classe> selectAll() {
67          String sql = "SELECT * FROM classes";
68          try {
69              List<Classe> you = DBUtils.queryBeanList(conn, sql,
70                      new ClassIResultSetCall());
71              return you;
72          } catch (Exception e) {
73              return null;
74          }
75      }
76
77      public static Classe selectById(int id) {
78          String sql = "SELECT * FROM classes WHERE id = ?";
79          try {
80              Classe you = DBUtils.queryBean(conn, sql, Classe.class, id);
81              return you;
82          } catch (Exception e) {
83              return null;
84          }
85      }
86
87  }
88
89  class ClassIResultSetCall implements IResultSetCall<Classe> {
90      public Classe invoke(ResultSet rs) throws SQLException {
91          Classe e = new Classe(rs.getInt("id"), rs.getString("name"));
92          return e;
93      }
94  }
```

Listing 3: DBUtils.java

```
1  package exp6;
2
```

```java
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Time;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 数据库工具类
 * 参考 https://github.com/felixyin/DBUtil/
 *
 * @version 2015-5-14
 * @author Kingfree
 */
public final class DBUtils {

    private DBUtils() {
    }

    static {
        try {
            Class.forName("org.sqlite.JDBC");
        } catch (ClassNotFoundException e) {
            System.out.println("驱动加载出错!");
        }
    }

    private static Connection con = null;

    public static Connection openConnection() throws SQLException {
        if (null == con || con.isClosed()) {
            con = DriverManager.getConnection("jdbc:sqlite:bin/sample.db");
        }
        return con;
    }

    public static void closeConnection() throws SQLException {
        try {
            if (null != con)
                con.close();
        } finally {
            con = null;
            System.gc();
        }
    }

```

```
58      public static List<Map<String, Object>> queryMapList(Connection con,
59              String sql) throws SQLException, InstantiationException,
60              IllegalAccessException {
61          List<Map<String, Object>> lists = new ArrayList<Map<String, Object>>();
62          Statement preStmt = null;
63          ResultSet rs = null;
64          try {
65              preStmt = con.createStatement();
66              rs = preStmt.executeQuery(sql);
67              ResultSetMetaData rsmd = rs.getMetaData();
68              int columnCount = rsmd.getColumnCount();
69              while (null != rs && rs.next()) {
70                  Map<String, Object> map = new HashMap<String, Object>();
71                  for (int i = 0; i < columnCount; i++) {
72                      String name = rsmd.getColumnName(i + 1);
73                      Object value = rs.getObject(name);
74                      map.put(name, value);
75                  }
76                  lists.add(map);
77              }
78          } finally {
79              if (null != rs)
80                  rs.close();
81              if (null != preStmt)
82                  preStmt.close();
83          }
84          return lists;
85      }
86
87      public static List<Map<String, Object>> queryMapList(Connection con,
88              String sql, Object... params) throws SQLException,
89              InstantiationException, IllegalAccessException {
90          List<Map<String, Object>> lists = new ArrayList<Map<String, Object>>();
91          PreparedStatement preStmt = null;
92          ResultSet rs = null;
93          try {
94              preStmt = con.prepareStatement(sql);
95              for (int i = 0; i < params.length; i++)
96                  preStmt.setObject(i + 1, params[i]);// 下标从1开始
97              rs = preStmt.executeQuery();
98              ResultSetMetaData rsmd = rs.getMetaData();
99              int columnCount = rsmd.getColumnCount();
100             while (null != rs && rs.next()) {
101                 Map<String, Object> map = new HashMap<String, Object>();
102                 for (int i = 0; i < columnCount; i++) {
103                     String name = rsmd.getColumnName(i + 1);
104                     Object value = rs.getObject(name);
105                     map.put(name, value);
106                 }
107                 lists.add(map);
108             }
109         } finally {
110             if (null != rs)
111                 rs.close();
112             if (null != preStmt)
```

```
113                 preStmt.close();
114             }
115             return lists;
116         }
117 
118         public static <T> List<T> queryBeanList(Connection con, String sql,
119                 Class<T> beanClass) throws SQLException, InstantiationException,
120                 IllegalAccessException {
121             List<T> lists = new ArrayList<T>();
122             Statement stmt = null;
123             ResultSet rs = null;
124             Field[] fields = null;
125             try {
126                 stmt = con.createStatement();
127                 rs = stmt.executeQuery(sql);
128                 fields = beanClass.getDeclaredFields();
129                 for (Field f : fields)
130                     f.setAccessible(true);
131                 while (null != rs && rs.next()) {
132                     T t = beanClass.newInstance();
133                     for (Field f : fields) {
134                         String name = f.getName();
135                         try {
136                             Object value = rs.getObject(name);
137                             setValue(t, f, value);
138                         } catch (Exception e) {
139                         }
140                     }
141                     lists.add(t);
142                 }
143             } finally {
144                 if (null != rs)
145                     rs.close();
146                 if (null != stmt)
147                     stmt.close();
148             }
149             return lists;
150         }
151 
152         public static <T> List<T> queryBeanList(Connection con, String sql,
153                 Class<T> beanClass, Object... params) throws SQLException,
154                 InstantiationException, IllegalAccessException {
155             List<T> lists = new ArrayList<T>();
156             PreparedStatement preStmt = null;
157             ResultSet rs = null;
158             Field[] fields = null;
159             try {
160                 preStmt = con.prepareStatement(sql);
161                 for (int i = 0; i < params.length; i++)
162                     preStmt.setObject(i + 1, params[i]);
163                 rs = preStmt.executeQuery();
164                 fields = beanClass.getDeclaredFields();
165                 for (Field f : fields)
166                     f.setAccessible(true);
167                 while (null != rs && rs.next()) {
```

```
168                 T t = beanClass.newInstance();
169                 for (Field f : fields) {
170                     String name = f.getName();
171                     try {
172                         Object value = rs.getObject(name);
173                         setValue(t, f, value);
174                     } catch (Exception e) {
175                     }
176                 }
177                 lists.add(t);
178             }
179         } finally {
180             if (null != rs)
181                 rs.close();
182             if (null != preStmt)
183                 preStmt.close();
184         }
185         return lists;
186     }
187 
188     public static <T> List<T> queryBeanList(Connection con, String sql,
189             IResultSetCall<T> qdi) throws SQLException {
190         List<T> lists = new ArrayList<T>();
191         Statement stmt = null;
192         ResultSet rs = null;
193         try {
194             stmt = con.createStatement();
195             rs = stmt.executeQuery(sql);
196             while (null != rs && rs.next())
197                 lists.add(qdi.invoke(rs));
198         } finally {
199             if (null != rs)
200                 rs.close();
201             if (null != stmt)
202                 stmt.close();
203         }
204         return lists;
205     }
206 
207     public static <T> List<T> queryBeanList(Connection con, String sql,
208             IResultSetCall<T> qdi, Object... params) throws SQLException {
209         List<T> lists = new ArrayList<T>();
210         PreparedStatement preStmt = null;
211         ResultSet rs = null;
212         try {
213             preStmt = con.prepareStatement(sql);
214             for (int i = 0; i < params.length; i++)
215                 preStmt.setObject(i + 1, params[i]);
216             rs = preStmt.executeQuery();
217             while (null != rs && rs.next())
218                 lists.add(qdi.invoke(rs));
219         } finally {
220             if (null != rs)
221                 rs.close();
222             if (null != preStmt)
```

```
223                 preStmt.close();
224             }
225             return lists;
226         }
227 
228         public static <T> T queryBean(Connection con, String sql, Class<T> beanClass)
229                 throws SQLException, InstantiationException, IllegalAccessException {
230             List<T> lists = queryBeanList(con, sql, beanClass);
231             if (lists.size() != 1)
232                 throw new SQLException("SQLError: 期待一行返回值，却返回了太多行！");
233             return lists.get(0);
234         }
235 
236         public static <T> T queryBean(Connection con, String sql,
237                 Class<T> beanClass, Object... params) throws SQLException,
238                 InstantiationException, IllegalAccessException {
239             List<T> lists = queryBeanList(con, sql, beanClass, params);
240             if (lists.size() != 1)
241                 throw new SQLException("SQLError: 期待一行返回值，却返回了太多行！");
242             return lists.get(0);
243         }
244 
245         @SuppressWarnings("unchecked")
246         public static <T> List<T> queryObjectList(Connection con, String sql,
247                 Class<T> objClass) throws SQLException, InstantiationException,
248                 IllegalAccessException {
249             List<T> lists = new ArrayList<T>();
250             Statement stmt = null;
251             ResultSet rs = null;
252             try {
253                 stmt = con.createStatement();
254                 rs = stmt.executeQuery(sql);
255                 label: while (null != rs && rs.next()) {
256                     Constructor<?>[] constor = objClass.getConstructors();
257                     for (Constructor<?> c : constor) {
258                         Object value = rs.getObject(1);
259                         try {
260                             lists.add((T) c.newInstance(value));
261                             continue label;
262                         } catch (Exception e) {
263                         }
264                     }
265                 }
266             } finally {
267                 if (null != rs)
268                     rs.close();
269                 if (null != stmt)
270                     stmt.close();
271             }
272             return lists;
273         }
274 
275         public static <T> List<T> queryObjectList(Connection con, String sql,
276                 Class<T> objClass, Object... params) throws SQLException,
277                 InstantiationException, IllegalAccessException {
```

```
278         List<T> lists = new ArrayList<T>();
279         PreparedStatement preStmt = null;
280         ResultSet rs = null;
281         try {
282             preStmt = con.prepareStatement(sql);
283             for (int i = 0; i < params.length; i++)
284                 preStmt.setObject(i + 1, params[i]);
285             rs = preStmt.executeQuery();
286             label: while (null != rs && rs.next()) {
287                 Constructor<?>[] constor = objClass.getConstructors();
288                 for (Constructor<?> c : constor) {
289                     String value = rs.getObject(1).toString();
290                     try {
291                         @SuppressWarnings("unchecked")
292                         T t = (T) c.newInstance(value);
293                         lists.add(t);
294                         continue label;
295                     } catch (Exception e) {
296                     }
297                 }
298             }
299         } finally {
300             if (null != rs)
301                 rs.close();
302             if (null != preStmt)
303                 preStmt.close();
304         }
305         return lists;
306     }
307
308     public static <T> T queryObject(Connection con, String sql,
309             Class<T> objClass) throws SQLException, InstantiationException,
310             IllegalAccessException {
311         List<T> lists = queryObjectList(con, sql, objClass);
312         if (lists.size() != 1)
313             throw new SQLException("SQLError: 期待一行返回值, 却返回了太多行!");
314         return lists.get(0);
315     }
316
317     public static <T> T queryObject(Connection con, String sql,
318             Class<T> objClass, Object... params) throws SQLException,
319             InstantiationException, IllegalAccessException {
320         List<T> lists = queryObjectList(con, sql, objClass, params);
321         if (lists.size() != 1)
322             throw new SQLException("SQLError: 期待一行返回值, 却返回了太多行!");
323         return lists.get(0);
324     }
325
326     public static int[] executeAsBatch(Connection con, List<String> sqlList)
327             throws SQLException {
328         return executeAsBatch(con, sqlList.toArray(new String[] {}));
329     }
330
331     public static int[] executeAsBatch(Connection con, String[] sqlArray)
332             throws SQLException {
```

```
        Statement stmt = null;
        try {
            stmt = con.createStatement();
            for (String sql : sqlArray) {
                stmt.addBatch(sql);
            }
            return stmt.executeBatch();
        } finally {
            if (null != stmt) {
                stmt.close();
            }
        }
    }

    public static int[] executeAsBatch(Connection con, String sql,
            Object[][] params) throws SQLException {
        PreparedStatement preStmt = null;
        try {
            preStmt = con.prepareStatement(sql);
            for (int i = 0; i < params.length; i++) {
                Object[] rowParams = params[i];
                for (int k = 0; k < rowParams.length; k++) {
                    Object obj = rowParams[k];
                    preStmt.setObject(k + 1, obj);
                }
                preStmt.addBatch();
            }
            return preStmt.executeBatch();
        } finally {
            if (null != preStmt) {
                preStmt.close();
            }
        }
    }

    public static int execute(Connection con, String sql) throws SQLException {
        Statement stmt = null;
        try {
            stmt = con.createStatement();
            return stmt.executeUpdate(sql);
        } finally {
            if (null != stmt)
                stmt.close();
        }
    }

    public static int execute(Connection con, String sql, Object... params)
            throws SQLException {
        PreparedStatement preStmt = null;
        try {
            preStmt = con.prepareStatement(sql);
            for (int i = 0; i < params.length; i++)
                preStmt.setObject(i + 1, params[i]);
            return preStmt.executeUpdate();
        } finally {
```

```
388             if (null != preStmt)
389                 preStmt.close();
390         }
391     }
392
393     private static <T> void setValue(T t, Field f, Object value)
394             throws IllegalAccessException {
395         if (null == value)
396             return;
397         String v = value.toString();
398         String n = f.getType().getName();
399         if ("java.lang.Byte".equals(n) || "byte".equals(n)) {
400             f.set(t, Byte.parseByte(v));
401         } else if ("java.lang.Short".equals(n) || "short".equals(n)) {
402             f.set(t, Short.parseShort(v));
403         } else if ("java.lang.Integer".equals(n) || "int".equals(n)) {
404             f.set(t, Integer.parseInt(v));
405         } else if ("java.lang.Long".equals(n) || "long".equals(n)) {
406             f.set(t, Long.parseLong(v));
407         } else if ("java.lang.Float".equals(n) || "float".equals(n)) {
408             f.set(t, Float.parseFloat(v));
409         } else if ("java.lang.Double".equals(n) || "double".equals(n)) {
410             f.set(t, Double.parseDouble(v));
411         } else if ("java.lang.String".equals(n)) {
412             f.set(t, value.toString());
413         } else if ("java.lang.Character".equals(n) || "char".equals(n)) {
414             f.set(t, (Character) value);
415         } else if ("java.lang.Date".equals(n)) {
416             f.set(t, new Date(((java.sql.Date) value).getTime()));
417         } else if ("java.lang.Timer".equals(n)) {
418             f.set(t, new Time(((java.sql.Time) value).getTime()));
419         } else if ("java.sql.Timestamp".equals(n)) {
420             f.set(t, (java.sql.Timestamp) value);
421         } else {
422             System.out.println("SqlError: 暂时不支持此数据类型，请使用其他类型代替
423         }
424     }
425
426 }
```

Listing 4: IResultSetCall.java

```
1  package exp6;
2
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5
6  /**
7   * 数据集调用接口
8   */
9  public interface IResultSetCall<T> {
10
11     public T invoke(ResultSet rs) throws SQLException;
12
13 }
```

Listing 5: StudentManager.java

```
1  package exp6;
2
3  import java.util.List;
4  import java.util.Map;
5  import java.util.Scanner;
6  import java.util.Set;
7  import java.util.regex.MatchResult;
8  import java.util.regex.Pattern;
9
10 /**
11  * 学生信息管理系统
12  *
13  * @version 2015-6-10
14  * @author Kingfree
15  */
16 public class StudentManager {
17
18     private static Scanner in = new Scanner(System.in);
19
20     public static int mainMenu() {
21         System.out.println("学生信息管理系统");
22         System.out.println("1 显示所有学生信息 2 按学号查找 3 按姓名查找");
23         System.out.println("4 按学号删除 5 按成绩排序 6 退出");
24         System.out.println("7 添加学生");
25         System.out.println("请输入数字(1-6)");
26         int sel = in.nextInt();
27         return sel;
28     }
29
30     public static void cli() {
31         int sel = 0;
32         do {
33             sel = mainMenu();
34             switch (sel) {
35             case 1:
36                 showAll();
37                 break;
38             case 2:
39                 findById();
40                 break;
41             case 3:
42                 findByName();
43                 break;
44             case 4:
45                 delById();
46                 break;
47             case 5:
48                 sortScore();
49                 break;
50             case 7:
51                 addStudent();
52                 break;
53             case 6:
```

```
54                  System.out.println("成功退出系统！");
55                  return;
56              }
57          } while (sel != 0);
58      }
59
60      private static void add(Student stu) {
61          if (Student.insert(stu)) {
62              System.out.println("成功添加学生。");
63          } else {
64              System.out.println("添加学生失败！");
65          }
66      }
67
68      public static Student inputStudent(Scanner in) {
69          for (;;) {
70              System.out.print(">");
71              try (Scanner line = new Scanner(in.nextLine())) {
72                  line.findInLine(Pattern
73                          .compile("(\\d+)\\s(\\d+)\\s(.+)\\s(\\d+)\\s(\\d+)\\s(\\d+
74                  try {
75                      MatchResult result = line.match();
76                      int id = Integer.parseInt(result.group(1));
77                      int cid = Integer.parseInt(result.group(2));
78                      String name = result.group(3);
79                      int os = Integer.parseInt(result.group(4));
80                      int math = Integer.parseInt(result.group(5));
81                      int java = Integer.parseInt(result.group(6));
82                      Student you = new Student(id, cid, name, os, math, java);
83                      return you;
84                  } catch (Exception e) {
85                      System.out.println("输入格式有误，请重新输入！");
86                  }
87              }
88          }
89      }
90
91      private static void addStudent() {
92          System.out.println("请输入学生信息：");
93          System.out.println("格式：<学号> <班级> <姓名> <数学成绩> <Java成绩> <操作
94
95          try (Scanner in = new Scanner(System.in)) {
96              Student you = inputStudent(in);
97              System.out.println("成功读入学生 '" + you + "'");
98              add(you);
99          }
100     }
101
102     private static void sortScore() {
103         System.out.println("1 按math成绩 2 按os成绩 3 按java成绩，请输入(1-3)");
104         int sel = in.nextInt();
105         switch (sel) {
106         case 1:
107             sortMath();
108             break;
```

```
109             case 2:
110                 sortOS();
111                 break;
112             case 3:
113                 sortJava();
114                 break;
115             default:
116                 return;
117             }
118         }
119
120         private static void showBy(String col) {
121             print(Student.selectOrberBy(col));
122         }
123
124         private static void sortOS() {
125             showBy("os");
126         }
127
128         private static void sortJava() {
129             showBy("java");
130         }
131
132         private static void sortMath() {
133             showBy("math");
134         }
135
136         private static void delById() {
137             System.out.println("请输入学号：");
138             int id = in.nextInt();
139             Student you = Student.selectById(id);
140             if (you == null) {
141                 System.out.println("没有找到学生！");
142                 return;
143             }
144             System.out.println("你确定删除学生 '" + you + "' 吗？(Y/n)");
145             String s = in.nextLine();
146             s = in.nextLine();
147             char c = s.charAt(0);
148             System.out.println(s);
149             System.out.println(c);
150             if (!(c == 'y' || c == 'Y')) {
151                 return;
152             }
153
154             if (Student.deleteById(id)) {
155                 System.out.println("已删除。");
156             } else {
157                 System.out.println("删除失败！");
158             }
159         }
160
161         private static void findByName() {
162             System.out.println("请输入姓名：");
163             String name = in.next().trim();
```

```
164             System.out.println("查找：'" + name + "'");
165             List<Student> you = Student.selectByName(name);
166             if (you == null) {
167                 System.out.println("没有找到！");
168             } else {
169                 print(you);
170             }
171         }
172
173         private static void findById() {
174             System.out.println("请输入学号：");
175             int id = in.nextInt();
176             Student you = Student.selectById(id);
177             if (you == null) {
178                 System.out.println("没有找到！");
179             } else {
180                 print(you);
181             }
182         }
183
184         private static void showAll() {
185             print(Student.selectAll());
186         }
187
188         public static void gui() {
189             new LoginWindow();
190         }
191
192         public static void cheet() {
193             new MainWindow();
194         }
195
196         /**
197          * 系统启动入口
198          * @param args 指定启动方式
199          */
200         public static void main(String[] args) {
201             String arg = "-gui";
202             if (args.length > 0) {
203                 arg = args[0].toLowerCase();
204             }
205             switch (arg) {
206             case "-gui":
207                 gui();
208                 break;
209             case "-cheet":
210                 cheet();
211                 break;
212             case "-cli":
213                 cli();
214                 break;
215             default:
216                 System.out.println("用法：java ssys.StudentManager [参数]");
217                 System.out.println("参数：\t-gui \t图形界面（默认）");
218                 System.out.println("      \t-cli \t命令行界面");
```

```
        }
    }

    private static void print(Object obj) {
        if (obj instanceof List) {
            List<?> list = (List<?>) obj;
            for (Object o : list) {
                if (o instanceof Map) {
                    @SuppressWarnings("unchecked")
                    Map<String, Object> map = (Map<String, Object>) o;
                    Set<String> set = map.keySet();
                    for (String key : set) {
                        Object value = map.get(key);
                        System.out.print(key + ":" + value + "\t");
                    }
                    System.out.println();
                } else {
                    System.out.println(o);
                }
            }
            System.out.println("总共查询出数据数量是：" + list.size());
        } else {
            System.out.println(obj);
        }
    }

}
```

Listing 6: MainWindow.java

```
package exp6;

import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Insets;
import java.util.Enumeration;
import java.util.List;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import javax.swing.plaf.FontUIResource;

/**
 * 主窗口
 *
 * @version 2015-6-10
 * @author Kingfree
 */
public class MainWindow extends JFrame {

    private static final long serialVersionUID = 1L;
    MainWindow 主窗口;
```

```
26
27      private static void InitGlobalFont(Font font) {
28          FontUIResource fontRes = new FontUIResource(font);
29          for (Enumeration<Object> keys = UIManager.getDefaults().keys(); keys
30                  .hasMoreElements();) {
31              Object key = keys.nextElement();
32              Object value = UIManager.get(key);
33              if (value instanceof FontUIResource) {
34                  UIManager.put(key, fontRes);
35              }
36          }
37      }
38
39      @Override
40      public Insets getInsets() {
41          Insets squeeze = new Insets(40, 20, 20, 20); // 上左下右
42          return squeeze;
43      }
44
45      MainWindow() {
46          主窗口 = this;
47
48          try {
49              UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
50              InitGlobalFont(new Font("微软雅黑", Font.PLAIN, 12));
51          } catch (Exception e) {
52          }
53
54          主窗口.setTitle("学生信息管理系统");
55          主窗口.setSize(190, 400);
56          主窗口.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
57          主窗口.setLocationRelativeTo(null); // 居中
58          主窗口.setLayout(new GridLayout(10, 1, 10, 10));
59
60          主窗口.add(new JLabel("<html><h2>学生信息管理系统</h2></html>"));
61
62          JButton 班级和学生 = new JButton("班级和学生");
63          班级和学生.addActionListener(new java.awt.event.ActionListener() {
64              public void actionPerformed(java.awt.event.ActionEvent e) {
65                  new ClassWindow();
66              }
67          });
68          主窗口.add(班级和学生);
69
70          JButton 全部学生 = new JButton("列出全部学生");
71          全部学生.addActionListener(new java.awt.event.ActionListener() {
72              public void actionPerformed(java.awt.event.ActionEvent e) {
73                  new StudentWindow();
74              }
75          });
76          主窗口.add(全部学生);
77
78          JButton 按姓名查找学生 = new JButton("按姓名查找学生");
79          按姓名查找学生.addActionListener(new java.awt.event.ActionListener() {
80              public void actionPerformed(java.awt.event.ActionEvent e) {
```

```
81                  String name = JOptionPane.showInputDialog(null, "请输入姓名:", "查找"
82                          JOptionPane.QUESTION_MESSAGE);
83                  if (name != null) {
84                      new StudentWindow(Student.selectByName(name));
85                  }
86              }
87          });
88          主窗口.add(按姓名查找学生);
89
90          JButton 学生排序 = new JButton("按成绩对学生排序");
91          学生排序.addActionListener(new java.awt.event.ActionListener() {
92              public void actionPerformed(java.awt.event.ActionEvent e) {
93                  String[] 选项 = { "数学", "Java", "操作系统" };
94                  String[] 对应 = { "math", "java", "os" };
95                  int sel = JOptionPane.showOptionDialog(null, "请选择排序基准:",
96                          "按成绩排序", JOptionPane.DEFAULT_OPTION,
97                          JOptionPane.QUESTION_MESSAGE, null, 选项, 选项[0]);
98                  if (sel >= 0 && sel < 3) {
99                      new StudentWindow(Student.selectOrberBy(对应[sel]));
100                 }
101             }
102         });
103         主窗口.add(学生排序);
104
105         JButton 添加学生 = new JButton("添加学生");
106         添加学生.addActionListener(new java.awt.event.ActionListener() {
107             public void actionPerformed(java.awt.event.ActionEvent e) {
108                 new StudentAddWindow();
109             }
110         });
111         主窗口.add(添加学生);
112
113         JButton 删除学生 = new JButton("删除学生");
114         删除学生.addActionListener(new java.awt.event.ActionListener() {
115             public void actionPerformed(java.awt.event.ActionEvent e) {
116                 int id = 0;
117                 try {
118                     id = Integer.parseInt(JOptionPane.showInputDialog(null,
119                             "请输入学号:"));
120                 } catch (Exception ee) {
121                     return;
122                 }
123                 Student you = Student.selectById(id);
124                 if (you == null) {
125                     JOptionPane.showMessageDialog(null, "没有找到学号为'" + id
126                             + "'的学生!", "错误", JOptionPane.ERROR_MESSAGE);
127                     return;
128                 }
129                 int sel = JOptionPane.showConfirmDialog(null, "你确定删除学生 '" + yo
130                         + "' 吗?", "确认删除", JOptionPane.YES_NO_OPTION);
131                 if (sel == JOptionPane.YES_OPTION) {
132                     if (Student.deleteById(id)) {
133                         JOptionPane.showMessageDialog(null, "删除成功!", "删除",
134                                 JOptionPane.INFORMATION_MESSAGE);
135                     } else {
```

```
                        JOptionPane.showMessageDialog(null, "删除失败！", "错误",
                                JOptionPane.ERROR_MESSAGE);
                    }
                }
            }
        });
        主窗口.add(删除学生);

        JButton 关于 = new JButton("关于");
        关于.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent e) {
                JOptionPane.showMessageDialog(null,
                        "<html><h1>关于</h1><p>作者： 田劲锋</p>"
                                + "<p>最后修改时间： 2015年6月11日</p></html>", "关
                        JOptionPane.INFORMATION_MESSAGE);
            }
        });
        主窗口.add(关于);

        主窗口.setVisible(true);
    }
}
```

Listing 7: ClassWindow.java

```
package exp6;

import java.awt.BorderLayout;
import java.awt.Insets;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.List;

import javax.swing.DefaultComboBoxModel;
import javax.swing.DefaultListModel;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class ClassWindow extends JFrame {

    private static final long serialVersionUID = 1L;
    private ClassWindow 主窗口 = null;
    private JPanel 主面板 = null;
    DefaultListModel<Student> 学生模型 = new DefaultListModel<>();
    DefaultComboBoxModel<Classe> 班级模型 = new DefaultComboBoxModel<>();
    JComboBox<Classe> 班级下拉框 = null;
    JList<Student> 学生列表框 = null;

    ClassWindow() {
        super();
        初始化();
```

```
31          主窗口 = this;
32      }
33
34      private void 初始化() {
35          this.setTitle("班级和学生");
36          this.setSize(300, 260);
37          this.setContentPane(取主面板());
38          this.setLocationRelativeTo(null); // 居中
39          this.setVisible(true);
40      }
41
42      @Override
43      public Insets getInsets() {
44          Insets squeeze = new Insets(40, 20, 20, 20); // 上左下右
45          return squeeze;
46      }
47
48      private JPanel 取主面板() {
49          if (主面板 == null) {
50              主面板 = new JPanel();
51              主面板.setLayout(new BorderLayout());
52
53              JPanel 班级面板 = new JPanel();
54              班级面板.setLayout(new BorderLayout());
55              班级面板.add(new JLabel("班级："), BorderLayout.WEST);
56
57              List<Classe> classes = Classe.selectAll();
58              for (Classe cls : classes) {
59                  班级模型.addElement(cls);
60              }
61
62              班级下拉框 = new JComboBox<Classe>(班级模型);
63              班级下拉框.addItemListener(new ItemListener() {
64                  public void itemStateChanged(ItemEvent ie) {
65                      if (ie.getStateChange() == ItemEvent.SELECTED) {
66                          主窗口.changeList((Classe) ie.getItem());
67                      }
68                  }
69              });
70              班级面板.add(班级下拉框);
71
72              JPanel 学生面板 = new JPanel();
73              学生面板.setLayout(new BorderLayout());
74              学生面板.add(new JLabel("学生："), BorderLayout.WEST);
75              学生列表框 = new JList<>(学生模型);
76
77              JScrollPane 列表框面板 = new JScrollPane(学生列表框);
78              学生面板.add(列表框面板);
79
80              主面板.add(班级面板, BorderLayout.NORTH);
81              主面板.add(学生面板);
82
83              if (classes != null && !classes.isEmpty()) {
84                  changeList(classes.get(0));
85              }
```

```
86          }
87          return 主面板;
88      }
89
90      private void changeList(Classe cls) {
91          try {
92              学生模型.clear();
93              for (Student stu : Student.selectByClass(cls)) {
94                  学生模型.addElement(stu);
95              }
96          } catch (Exception e) {
97          }
98      }
99
100 }
```

Listing 8: LoginWindow.java

```
1   package exp6;
2
3   import java.awt.BorderLayout;
4   import java.awt.GridLayout;
5
6   import javax.swing.JButton;
7   import javax.swing.JFrame;
8   import javax.swing.JLabel;
9   import javax.swing.JOptionPane;
10  import javax.swing.JPanel;
11  import javax.swing.JPasswordField;
12  import javax.swing.JTextField;
13  import javax.swing.SwingConstants;
14  import javax.swing.SwingUtilities;
15  import javax.swing.UIManager;
16
17  public class LoginWindow extends JFrame {
18
19      private static final long serialVersionUID = 1L;
20      private JPanel 主面板 = null;
21      private JLabel 用户标签 = null;
22      private JLabel 密码标签 = null;
23      private JTextField 用户名框 = null;
24      private JButton 确认按钮 = null;
25      private JButton 取消按钮 = null;
26      private LoginWindow 主窗口 = null;
27      private JPasswordField 密码框 = null;
28
29      private static String username = "root";
30      private static String password = "root";
31
32      public LoginWindow() {
33          super();
34
35          try {
36              UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
37          } catch (Exception e) {
```

```
38          }
39
40          初始化();
41          主窗口 = this;
42      }
43
44      private void 初始化() {
45          this.setTitle("登录");
46          this.setContentPane(取主面板());
47          pack();
48          this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
49          this.setLocationRelativeTo(null); // 居中
50          this.setVisible(true);
51      }
52
53      private JPanel 取主面板() {
54          if (主面板 == null) {
55              主面板 = new JPanel();
56              主面板.setLayout(new BorderLayout());
57              用户标签 = new JLabel("用户名：", SwingConstants.RIGHT);
58              密码标签 = new JLabel("密码：", SwingConstants.RIGHT);
59              JPanel 输入面板 = new JPanel();
60              输入面板.setLayout(new GridLayout(2, 2));
61              输入面板.add(用户标签);
62              输入面板.add(取用户名框());
63              输入面板.add(密码标签);
64              输入面板.add(取密码框());
65              主面板.add(输入面板, BorderLayout.NORTH);
66              JPanel 确认面板 = new JPanel();
67              确认面板.add(取确认按钮());
68              确认面板.add(取取消按钮());
69              主面板.add(确认面板, BorderLayout.SOUTH);
70          }
71          return 主面板;
72      }
73
74      private JTextField 取用户名框() {
75          if (用户名框 == null) {
76              用户名框 = new JTextField();
77              用户名框.setColumns(8);
78          }
79          return 用户名框;
80      }
81
82      private JPasswordField 取密码框() {
83          if (密码框 == null) {
84              密码框 = new JPasswordField();
85              密码框.setColumns(8);
86              密码框.setEchoChar('*');
87          }
88          return 密码框;
89      }
90
91      private JButton 取确认按钮() {
92          if (确认按钮 == null) {
```

```
93              确认按钮 = new JButton("确定");
94              确认按钮.addActionListener(new java.awt.event.ActionListener() {
95                  public void actionPerformed(java.awt.event.ActionEvent e) {
96                      String username = 主窗口.用户名框.getText().trim().toLowerCase
97                      String password = new String(主窗口.密码框.getPassword());
98                      if (username.equals(LoginWindow.username)
99                              && password.equals(LoginWindow.password)) {
100                         // JOptionPane.showMessageDialog(null, "登录成功");
101                         主窗口.setVisible(false);
102                         SwingUtilities.invokeLater(new Runnable() {
103                             public void run() {
104                                 new MainWindow();
105                             }
106                         });
107                     } else {
108                         JOptionPane.showMessageDialog(null, "登录失败");
109                     }
110                 }
111             });
112         }
113         return 确认按钮;
114     }
115
116     private JButton 取取消按钮() {
117         if (取消按钮 == null) {
118             取消按钮 = new JButton("取消");
119             取消按钮.addActionListener(new java.awt.event.ActionListener() {
120                 public void actionPerformed(java.awt.event.ActionEvent e) {
121                     主窗口.用户名框.setText("");
122                     主窗口.密码框.setText("");
123                 }
124             });
125         }
126         return 取消按钮;
127     }
128
129 }
```

Listing 9: StudentAddWindow.java

```
1  package exp6;
2
3  import java.awt.BorderLayout;
4  import java.awt.GridLayout;
5  import java.awt.Insets;
6  import java.awt.event.ItemEvent;
7  import java.awt.event.ItemListener;
8  import java.util.List;
9
10 import javax.swing.DefaultComboBoxModel;
11 import javax.swing.DefaultListModel;
12 import javax.swing.JButton;
13 import javax.swing.JComboBox;
14 import javax.swing.JFrame;
15 import javax.swing.JLabel;
```

```
16 import javax.swing.JOptionPane;
17 import javax.swing.JPanel;
18 import javax.swing.JTextField;
19
20 public class StudentAddWindow extends JFrame {
21
22     private static final long serialVersionUID = 1L;
23     private StudentAddWindow 主窗口 = null;
24     private JPanel 主面板 = null;
25     DefaultListModel<Student> 学生模型 = new DefaultListModel<>();
26     DefaultComboBoxModel<Classe> 班级模型 = new DefaultComboBoxModel<>();
27     JComboBox<Classe> 班级下拉框 = null;
28
29     StudentAddWindow() {
30         super();
31         初始化();
32         主窗口 = this;
33     }
34
35     private void 初始化() {
36         this.setTitle("添加学生");
37         // this.setSize(300, 400);
38         this.setContentPane(取主面板());
39         this.pack();
40         this.setLocationRelativeTo(null); // 居中
41         this.setVisible(true);
42     }
43
44     @Override
45     public Insets getInsets() {
46         Insets squeeze = new Insets(40, 20, 20, 20); // 上左下右
47         return squeeze;
48     }
49
50     private JPanel 取主面板() {
51         if (主面板 == null) {
52             主面板 = new JPanel();
53             主面板.setLayout(new GridLayout(7, 2, 10, 10));
54
55             JLabel 学号标签 = new JLabel("学号：");
56             JTextField 学号文本框 = new JTextField("" + (Student.getMaxId() + 1));
57             主面板.add(学号标签);
58             主面板.add(学号文本框);
59
60             JLabel 姓名标签 = new JLabel("姓名：");
61             JTextField 姓名文本框 = new JTextField();
62             主面板.add(姓名标签);
63             主面板.add(姓名文本框);
64
65             JLabel 班级标签 = new JLabel("班级：");
66             List<Classe> classes = Classe.selectAll();
67             for (Classe cls : classes) {
68                 班级模型.addElement(cls);
69             }
70             JComboBox<Classe> 班级下拉框 = new JComboBox<Classe>(班级模型);
```

```
71                 主面板.add(班级标签);
72                 主面板.add(班级下拉框);
73
74                 JLabel 数学标签 = new JLabel("数学成绩：");
75                 JTextField 数学文本框 = new JTextField("100");
76                 主面板.add(数学标签);
77                 主面板.add(数学文本框);
78
79                 JLabel Java标签 = new JLabel("Java成绩：");
80                 JTextField Java文本框 = new JTextField("100");
81                 主面板.add(Java标签);
82                 主面板.add(Java文本框);
83
84                 JLabel OS标签 = new JLabel("操作系统成绩：");
85                 JTextField OS文本框 = new JTextField("100");
86                 主面板.add(OS标签);
87                 主面板.add(OS文本框);
88
89                 JButton 添加按钮 = new JButton("添加");
90                 添加按钮.addActionListener(new java.awt.event.ActionListener() {
91                     public void actionPerformed(java.awt.event.ActionEvent e) {
92                         try {
93                             int id = Integer.parseInt(学号文本框.getText());
94                             int cid = ((Classe)班级下拉框.getSelectedItem()).getId();
95                             String name = 姓名文本框.getText().trim();
96                             int os = Integer.parseInt(OS文本框.getText());
97                             int math = Integer.parseInt(数学文本框.getText());
98                             int java = Integer.parseInt(Java文本框.getText());
99                             Student you = new Student(id, cid, name, os, math, java);
100                            if (Student.insert(you)) {
101                                JOptionPane.showMessageDialog(null, "添加成功！", "添加
102                                        JOptionPane.INFORMATION_MESSAGE);
103                                主窗口.setVisible(false);
104                            } else {
105                                JOptionPane.showMessageDialog(null, "添加失败！", "错误
106                                        JOptionPane.ERROR_MESSAGE);
107                            }
108                        } catch (Exception ee) {
109                            return;
110                        }
111                    }
112                });
113                主面板.add(添加按钮);
114
115            }
116            return 主面板;
117        }
118
119    }
```

Listing 10: StudentWindow.java

```
1   package exp6;
2
3   import java.awt.BorderLayout;
```

```java
import java.awt.Insets;
import java.util.List;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;

public class StudentWindow extends JFrame {

    private static final long serialVersionUID = 1L;
    private StudentWindow 主窗口 = null;
    private JPanel 主面板 = null;
    DefaultTableModel 学生模型 = null;
    private List<Student> 学生 = null;

    StudentWindow() {
        super();
        this.学生 = Student.selectAll();
        初始化();
    }

    StudentWindow(List<Student> 学生) {
        super();
        this.学生 = 学生;
        初始化();
    }

    StudentWindow(Student stu) {
        super();
        if (stu != null) {
            this.学生.add(stu);
        }
        初始化();
    }

    private void 初始化() {
        this.setTitle("学生信息");
        this.setContentPane(取主面板());
        this.pack();
        this.setLocationRelativeTo(null); // 居中
        this.setVisible(true);
    }

    @Override
    public Insets getInsets() {
        Insets squeeze = new Insets(40, 20, 20, 20); // 上左下右
        return squeeze;
    }

    private JPanel 取主面板() {
        if (主面板 == null) {
            主面板 = new JPanel();
            主面板.setLayout(new BorderLayout());

```

```
59              String[] 表头 = { "学号", "姓名", "班级", "数学", "Java", "操作系统" }
60              学生模型 = new DefaultTableModel(表头, 0);
61
62              for (Student stu : 学生) {
63                  Object[] a = { stu.getId(), stu.getName(),
64                          Classe.selectById(stu.getClassId()).getName(),
65                          stu.getMath(), stu.getJava(), stu.getOs() };
66                  学生模型.addRow(a);
67              }
68              JTable 学生表格 = new JTable(学生模型);
69              //学生表格.setEnabled(false);
70              学生表格.setAutoCreateRowSorter(true);
71              主面板.add(学生表格.getTableHeader(), BorderLayout.NORTH);
72              主面板.add(学生表格);
73          }
74          return 主面板;
75      }
76
77  }
```