```
 1 package expenses.manager;
 2 import static  expenses.manager.Log.logger;
 8 /**
 9 *
10 * @author Avital Chen
11 * Model Class that implements the IModel Interface
12 * In the Class we call the database
13 */
14 public class Model implements IModel {
15         public Derbydb db;
16         private String currMonth;
17         private String currYear;
18     public Model() {
19         synchronized(logger){
20              logger.info("new Model");
21          }
22         db = new Derbydb();
23         Date today = new Date();
24         Calendar cal = Calendar.getInstance();
25         cal.setTime(today);
26         int year = cal.get(Calendar.YEAR);
27         int month = cal.get(Calendar.MONTH);
28         currYear = String.valueOf(year);
29         currMonth = String.valueOf(month+1);
30     }
31     /**
32     *
33     * setIncome(String date, double amount)
34      *  Insert an income to the income table in the database
35      *  @param date the date of the income
36      *  @param amount the amount of the income
37      */
38     @Override
39
40     public void setIncome(String date, double amount) {
41         String incomeAmountAsString = String.valueOf(amount);
42         String sql = "insert into INCOME "+ "(date, amount) " +
  "values ('" + date+ "'," +incomeAmountAsString+ ")";
43         try
44         {
45             db.statement = db.connection.createStatement();
46             db.statement.executeUpdate(sql);
47             synchronized(logger){
48             logger.info("insert income: " + sql);
49             }
50
51             db.statement.close();
52         }
53         catch (SQLException sqlExcept)
```

```
54          {
55              sqlExcept.printStackTrace();
56          }
57      }
58      /**
59      *
60      * setExpenses(String type, String date,double amount)
61       *  Insert an expense to the expenses table in the database
62       *  @param type the type of the expense
63       *  @param date the date of the expense
64       *  @param amount the amount of the expense
65       */
66      @Override
67      public void setExpenses(String type, String date,double
    amount) {
68          String expensesAmountAsString = String.valueOf(amount);
69          String sql = "insert into EXPENSES "+ "(type, date,
    amount) " + "values ('"+type+"', '"+ date+"',"+
    expensesAmountAsString +")";
70          try
71          {
72              db.statement = db.connection.createStatement();
73
74              db.statement.executeUpdate(sql);
75              synchronized(logger){
76              logger.info("insert Expenses: " + sql);
77              }
78              db.statement.close();
79          }
80          catch (SQLException sqlExcept)
81          {
82              sqlExcept.printStackTrace();
83          }
84
85      }
86      /**
87      *
88      * getTotalExpensesCurrYear()
89       *  @return the sum of all the expenses this current year
90       */
91
92      @Override
93      public double getTotalExpensesCurrYear() {
94          double returnedResult = 0.0;
95          String sql = "SELECT SUM(amount), YEAR(date) FROM
    EXPENSES "
96                          + "WHERE YEAR(date) = " + currYear
97                          + " GROUP BY YEAR(date)";
98          try
```

```
 99         {
100             db.statement = db.connection.createStatement();
101             synchronized(logger){
102             logger.info("execute query: " + sql);
103             }
104             ResultSet results = db.statement.executeQuery(sql);
105             while(results.next())
106             {
107             returnedResult = results.getDouble(1);
108               synchronized(logger){
109                     logger.info("query result: " +
    returnedResult);
110                   }
111             }
112             results.close();
113             db.statement.close();
114         }
115         catch (SQLException sqlExcept)
116         {
117             sqlExcept.printStackTrace();
118         }
119          return returnedResult;
120 
121     }
122     /**
123     *
124     * getTotalIncomeCurrYear()
125      *  @return the sum of all the income this current year
126      */
127     @Override
128     public double getTotalIncomeCurrYear() {
129 
130         double returnedResult = 0.0;
131         String sql = "SELECT SUM(amount), YEAR(date) FROM Income
    "
132                 +  "WHERE YEAR(date) = " + currYear
133                 + " GROUP BY YEAR(date)";
134         try
135         {
136             db.statement = db.connection.createStatement();
137             synchronized(logger){
138             logger.info("execute query: " + sql);
139             }
140             ResultSet results = db.statement.executeQuery(sql);
141             while(results.next())
142             {
143                 returnedResult = results.getDouble(1);
144                 synchronized(logger){
145                     logger.info("query result: " +
```

```
returnedResult);
146                     }
147             }
148             results.close();
149             db.statement.close();
150         }
151         catch (SQLException sqlExcept)
152         {
153             sqlExcept.printStackTrace();
154         }
155         return returnedResult;
156     }
157     /**
158     *
159     * getTotalExpensesOverall()
160      *  @return the sum of all the expenses overall
161      */
162     @Override
163     public double getTotalExpensesOverall() {
164         double returnedResult = 0.0;
165         String sql = "SELECT SUM(amount) FROM EXPENSES ";
166         try
167         {
168             db.statement = db.connection.createStatement();
169             synchronized(logger){
170             logger.info("execute query: " + sql);
171             }
172             ResultSet results = db.statement.executeQuery(sql);
173             while(results.next())
174             {
175             returnedResult = results.getDouble(1);
176             synchronized(logger){
177                     logger.info("query result: " +
returnedResult);
178                     }
179             }
180             results.close();
181             db.statement.close();
182         }
183         catch (SQLException sqlExcept)
184         {
185             sqlExcept.printStackTrace();
186         }
187         return returnedResult;
188     }
189     /**
190     *
191     * getTotalIncomeOverall()
192      *  @return the sum of all the income overall
```

```
193      */
194     @Override
195     public double getTotalIncomeOverall() {
196         double returnedResult = 0.0;
197         String sql = "SELECT SUM(amount) FROM INCOME ";
198         try
199         {
200             db.statement = db.connection.createStatement();
201             synchronized(logger){
202             logger.info("execute query: " + sql);
203             }
204             ResultSet results = db.statement.executeQuery(sql);
205             while(results.next())
206             {
207             returnedResult = results.getDouble(1);
208             synchronized(logger){
209                     logger.info("query result: " +
    returnedResult);
210                 }
211             }
212             results.close();
213             db.statement.close();
214         }
215         catch (SQLException sqlExcept)
216         {
217             sqlExcept.printStackTrace();
218         }
219         System.out.println(returnedResult);
220         return returnedResult;
221     }
222     /**
223     *
224     * getTotalInclomeByMonth()
225      *  @return the sum of income grouped by month this year
226      */
227     @Override
228     public Vector getTotalInclomeByMonth() {
229         int index = 0;
230         Vector<Vector> returnedResult = new Vector<Vector>();
231 
232         String sql = "SELECT SUM(amount), YEAR(date), MONTH(date)
    FROM INCOME "
233                 + "WHERE YEAR(date) = " + currYear
234                 + " GROUP BY YEAR(date), MONTH"
235                 + "(date)";
236         try
237         {
238             db.statement = db.connection.createStatement();
239             synchronized(logger){
```

```
240             logger.info("execute query: " + sql);
241             }
242             ResultSet results = db.statement.executeQuery(sql);
243             while(results.next())
244             {
245             Vector v = new Vector();
246             v.add(0, results.getDouble(1));
247             v.add(1, results.getString(3));
248             returnedResult.add(index, v);
249             ++index;
250             }
251             results.close();
252             db.statement.close();
253         }
254         catch (SQLException sqlExcept)
255         {
256             sqlExcept.printStackTrace();
257         }
258         synchronized(logger){
259             logger.info("query result: " + returnedResult);
260           }
261         return returnedResult;
262     }
263     /**
264     *
265     * getTotalExpensesByMonth()
266      *  @return the sum of expenses grouped by month this year
267      */
268     @Override
269     public Vector getTotalExpensesByMonth() {
270         int index = 0;
271         Vector<Vector> returnedResult = new Vector<Vector>();
272         String sql = "SELECT SUM(amount), YEAR(date), MONTH(date)
    FROM EXPENSES "
273                 + "WHERE YEAR(date) = " + currYear
274                 + " GROUP BY YEAR(date), MONTH"
275                 + "(date)";
276         try
277         {
278             db.statement = db.connection.createStatement();
279             synchronized(logger){
280             logger.info("execute query: " + sql);
281             }
282             ResultSet results = db.statement.executeQuery(sql);
283             while(results.next())
284             {
285                 Vector v = new Vector();
286                 v.add(0, results.getDouble(1));
287                 v.add(1, results.getString(3));
```

```
288
289                 returnedResult.add(index, v);
290             }
291             results.close();
292             db.statement.close();
293         }
294         catch (SQLException sqlExcept)
295         {
296             sqlExcept.printStackTrace();
297         }
298
299         synchronized(logger){
300             logger.info("query result: " + returnedResult);
301         }
302         return returnedResult;
303     }
304     /**
305     *
306     * getTotalExpensesByType()
307      *  @return the sum of expenses grouped by type this year
308      */
309     @Override
310     public Vector getTotalExpensesByType() {
311
312         int index = 0;
313         Vector<Vector> returnedResult = new Vector<Vector>();
314         String sql = "SELECT SUM(amount), YEAR(date), type FROM
    EXPENSES "
315                 + "WHERE YEAR(date) = " + currYear
316                 + " GROUP BY YEAR(date), type";
317         try
318         {
319             db.statement = db.connection.createStatement();
320             synchronized(logger){
321             logger.info("execute query: " + sql);
322             }
323             ResultSet results = db.statement.executeQuery(sql);
324             while(results.next())
325             {
326              Vector v = new Vector();
327                 v.add(0, results.getDouble(1));
328                 v.add(0, results.getString(3));
329
330                 returnedResult.add(index, v);
331                 ++index;
332             }
333             results.close();
334             db.statement.close();
335         }
```

```
336         catch (SQLException sqlExcept)
337         {
338             sqlExcept.printStackTrace();
339         }
340         synchronized(logger){
341             logger.info("query result: " + returnedResult);
342         }
343         return returnedResult;
344     }
345     /**
346     *
347     * getTotalExpensesByTypeCurrMonth()
348      *  @return the sum of expenses grouped by type this month
349      */
350     @Override
351     public Vector getTotalExpensesByTypeCurrMonth() {
352         int index = 0;
353         Vector<Vector> returnedResult = new Vector<Vector>();
354         String sql = "SELECT SUM(amount), YEAR(date),MONTH(date),
    type FROM EXPENSES "
355                 + "WHERE YEAR(date) = " + currYear +" AND
    MONTH(date) = " + currMonth
356                 + " GROUP BY YEAR(date) ,MONTH(date), type";
357         try
358         {
359             db.statement = db.connection.createStatement();
360             synchronized(logger){
361             logger.info("execute query: " + sql);
362             }
363             ResultSet results = db.statement.executeQuery(sql);
364             while(results.next())
365             {
366             Vector v = new Vector();
367                 v.add(0, results.getDouble(1));
368                 v.add(0, results.getString(4));
369
370                 returnedResult.add(index, v);
371                 ++index;
372             }
373             results.close();
374             db.statement.close();
375         }
376         catch (SQLException sqlExcept)
377         {
378             sqlExcept.printStackTrace();
379         }
380         synchronized(logger){
381             logger.info("query result: " + returnedResult);
382         }
```

```
383         return returnedResult;
384     }
385 }
386
```