```java
package DBLayer;
import ModelLayer.*;

import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;

public class DBTransfer implements IFDBTransfer
{
   private Connection con;

   // Creates a new instance of DBPlasmDisease
   public DBTransfer()
   {
         con = DbConnection.getInstance().getDBcon();
   }


   @Override
   public ArrayList<Transfer> getAllTransfer(boolean
   retriveAssociation)
   {
         try
         {
               return miscWhere("", retriveAssociation);
         }
         catch (Exception e)
         {
               e.printStackTrace();
         }
         return null;
   }


   @Override
   public Transfer searchTransfersByTransferId(int transferId,
   boolean retriveAssociation)
   {
         String wClause = "  transferId = '" + transferId + "'";
         return singleWhere(wClause, retriveAssociation);
   }

   @Override
   public Transfer searchTransfersByCageNo(int cageNo, boolean
   retriveAssociation)
   {
         String wClause = "  cageNo = '" + cageNo + "'";
         return singleWhere(wClause, retriveAssociation);
   }


   @Override
   public ArrayList<Transfer> findTransfersByEmployeeId(int
   employeeId, boolean retriveAssociation)
   {
         String wClause = "  employeeId = '" + employeeId + "'";
         return miscWhere(wClause, retriveAssociation);
   }


   @Override
   public ArrayList<Transfer> findTransfersBydiseaseId(int
   diseaseId, boolean retriveAssociation)
```

```
65    {
66          String wClause = "  diseaseId = '" + diseaseId + "'";
67          return miscWhere(wClause, retriveAssociation);
68    }
69
70
71    @Override
72    public ArrayList<Transfer> findtransferByDate(String
73    transdate, boolean retriveAssociation)
74    {
75          String wClause = "  transferDate = '" + transdate + "'";
76                return miscWhere(wClause, retriveAssociation);
77    }
78
79
80    //update transfer.
81    @Override
82    public int updateTransfer(Transfer transferObj)
83    {
84          int rc = -1;
85          Transfer transfer = transferObj;
86
87          String query="UPDATE mfTransfer SET "+
88                      "cageNo ='" + transfer.getCage().getCageNo() + "', " +

89                      "diseaseId ='" + getDiseaseValue(transfer) + "', " +
90                      "employeeId ='" + transfer.getEmployee().getEmployeeID()
91                      + "', " +
92                      "transferDate ='" + transfer.getTransferDate() + "' "  +
93                      "WHERE transferId = '" + transfer.getTransferId() + "' ";
94
95        System.out.println("Update query:" + query);
96          try
97          {
98                Statement stmt = con.createStatement();
99                stmt.setQueryTimeout(5);
100               rc = stmt.executeUpdate(query);
101               stmt.close();
102         }
103         catch(Exception e)
104         {
105               System.out.println(e.getMessage());
106         }
107
108         return(rc);
109   }
110
111
112
113   //delete from transfer using id.
114   public int deleteTransfer(int transferId)
115     {
116         int rc = -1;
117         String delete = "DELETE FROM mfTransfer where
118         transferId ='" + transferId + "'";
119         System.out.println(delete);
120         try {
121               Statement stmt = con.createStatement();
122             //stmt.setQueryTimeout(5);
123             rc = stmt.executeUpdate(delete);
124             stmt.close();
125         }
126         catch(SQLException sqlE)
127         {
```

```
128             System.out.println(sqlE.getMessage());
129         }
130         catch(Exception e)
131         {
132               System.out.println(e.getMessage());
133         }
134
135         return rc;
136      }
137
138
139
140   @Override
141   public int insertTransfer(Transfer transferObj)
142   {
143         //call to get next transferId.
144         int nextTransferId = GetMax.getMaxId("Select max(
145         transferId) from mfTRansfer");
146         nextTransferId = nextTransferId + 1;
147         System.out.println("next transferId = " +  nextTransferId);
148
149         int rc = -1;
150         int diseaseValue = getDiseaseValue(transferObj);
151         String query="INSERT INTO mfTransfer(transferId, cageNo,
152         diseaseId, employeeId, transferDate)  VALUES('"+
153                     nextTransferId  + "','"  +
154                     transferObj.getCage().getCageNo() + "','" +
155                     diseaseValue + "','" +
156                     transferObj.getEmployee().getEmployeeID() + "','" +
157                     getTodaysDate() + "') ";
158         System.out.println("insert : " + query);
159
160         try
161         {
162               //insert new transfer with cagenum, diseaseid, employeeId.
163               Statement stmt = con.createStatement();
164           stmt.setQueryTimeout(5);
165         rc = stmt.executeUpdate(query);
166           stmt.close();
167
168           // Check if Cage number is in DiseaseReport DB, if
169           so delete it in DiseaseReport DB.
170               int cagenumber = transferObj.getCage().getCageNo();
171           IFDBDiseaseReport diseasereport = new
172           DBDiseaseReport();
173           try
174           {
175                     if(diseasereport.searchDiseaseReportByCageNumber(
176                     cagenumber, false) != null)
177                     {
178                           rc = diseasereport.deleteDiseaseReportWithCageNo(
179                           cagenumber);
180                     }
181               }
182           catch (Exception e)
183           {
184                     System.out.println(e.getMessage());
185               }
186
187         }//end try.
188       catch(SQLException ex)
189       {
190               System.out.println("Transfer not created"+ex.
191               getMessage());
```

```
192       }
193
194         return rc;
195   }
196
197
198   @Override
199   public int deleteTransfer(Transfer transferObj)
200   {
201         int rc=-1;
202         String query="DELETE FROM mfTransfer WHERE cageNo = '" +
203         transferObj.getCage().getCageNo() + "'";
204         System.out.println(query);
205         try
206         { // delete from Transfer
207               Statement stmt = con.createStatement();
208               stmt.setQueryTimeout(5);
209               rc = stmt.executeUpdate(query);
210               stmt.close();
211         }//slut try
212         catch(Exception ex)
213         {
214               System.out.println("Delete exception in mfTransfer db:
215               "+ex.getMessage());
216         }
217
218         return rc;
219   }
220
221   //alternative
222 /*      public int insertTransfer(Transfer transferObj)
223   {
224         int rc = -1;
225         PreparedStatement pstmt = null;
226         String insert = "INSERT INTO mfTransfer(transferId,
227         cageNo, diseaseId, employeeId, transferDate)"+
228         "values(?,?,?,?,?)";
229         System.out.println(insert);
230         try
231         {
232               pstmt = con.prepareStatement(insert);
233               pstmt.setInt(1, transferObj.getTransferId());
234               pstmt.setInt(2, transferObj.getCageNo());
235               pstmt.setInt(3, transferObj.getDiseaseId());
236               pstmt.setInt(4, transferObj.getEmployeeId());
237               pstmt.setString(5, transferObj.getTransferDate());
238               rc = pstmt.executeUpdate();
239         } catch(SQLException sqlE) {
240             System.out.println("SQL Error");
241             System.out.println(sqlE.getMessage());
242         } catch(Exception e) {
243             e.getMessage();
244         }
245         return rc;
246   }
247               */
248
249
250
251   //used when we select only one Transfer.
252   private Transfer singleWhere(String wClause, boolean
253   retrieveAssociation)
254   {
255         ResultSet results;
```

```
256         Transfer transferObj = null;
257         String query = buildQuery(wClause);
258         System.out.println(query);
259
260         try   {
261               //read the transfer from the database.
262               Statement stmt = con.createStatement();
263               stmt.setQueryTimeout(5);
264               results = stmt.executeQuery(query);
265
266               if( results.next() )
267               {
268                     transferObj = buildTransfer(results);
269
270                     if(retrieveAssociation)
271                     {
272                           // Based upon Cage number retreive a complete Cage
273                           object from DBCage
274                           IFDBCage dbcage = new DBCage();
275                           transferObj.setCage(dbcage.findCage(transferObj.
276                           getCage().getCageNo(), false));
277
278                           // Based upon Disease Id retreive a complete Disease
279                           object from DBDisease
280                     if (transferObj.getBitedisease() != null)
281                     {
282                           IFDBBiteDisease bitedisease = new
283                           DBBiteDisease();
284                           transferObj.setBitedisease(bitedisease.
285                           searchBiteDiseaseById(transferObj.
286                           getBitedisease().getDiseaseId(), false));
287                     }
288                     else
289                     {
290                           IFDBPlasmaDisease plasmadisease = new
291                           DBPlasmaDisease();
292                           transferObj.setPlasmadisease(plasmadisease.
293                           searchDiseaseById(transferObj.
294                           getPlasmadisease().getDiseaseId(), false));
295                     }
296
297                           // Based upon Employee Id retreive a complete Employee
298                           object from DBEmployee
299                     IFDBEmployee employee = new DBEmployee();
300                           transferObj.setEmployee(employee.findEmployeeByID(
301
    transferObj.getEmployee().getEmployeeID(), false));
302
303                           }
304               }
305
306               stmt.close();
307         }
308         catch(Exception ex)
309         {
310               System.out.println("Query Exception " + ex.getMessage());
311         }
312
313         return transferObj;
314   }
315
316
317
318   //method to build the query
```

```
319
320   private String buildQuery(String wClause)
321   {
322         String query = "SELECT transferId, cageNo, diseaseId,
323         employeeId, transferDate FROM mfTransfer";
324         if (wClause.length()>0)
325               query=query+" WHERE "+ wClause;
326         return query;
327   }
328
329
330   private Transfer buildTransfer(ResultSet results)
331   {
332         Transfer transObj = new Transfer();
333
334         //columns from mfTransfer table are used.
335         try
336         {
337               transObj.setTransferId(results.getInt("transferId"));
338               transObj.setCage(new Cage(results.getInt("cageNo")));
339
340               int valueOfDisease = results.getInt("diseaseId");
341               if (valueOfDisease == 1600)
342                   transObj.setBitedisease(new BiteDisease(
343                   valueOfDisease));
344               else
345                     transObj.setPlasmadisease(new PlasmaDisease(
346                     valueOfDisease));
347
348               transObj.setEmployee(new Employee(results.getInt(
349               "employeeID")));
350               transObj.setTransferDate(results.getString("transferDate")
351               );
352         }
353         catch (Exception ex)
354         {
355               System.out.println(ex.getMessage());
356               return null;
357         }
358
359         return transObj;
360   }
361
362
363 /*      //method to build transfer object.
364   private Transfer buildTransfer(ResultSet results)
365     {
366         try {
367               Transfer transferObj = new Transfer(
368                     results.getInt("transferId"),
369                     results.getInt("cageNo"),
370                     results.getInt("diseaseId"),
371                     results.getInt("employeeId"),
372                     results.getString("transferDate"));
373
374               return transferObj;
375
376         } catch (SQLException ex) {
377         // TODO Auto-generated catch block
378               ex.printStackTrace();
379         }
380         return null;
381   }
382   */
```

```
383
384   private ArrayList<Transfer> miscWhere(String wClause,
385   boolean retrieveAssociation)
386   {
387         ResultSet results;
388       ArrayList<Transfer> listT = new ArrayList <Transfer>();
389       String query = buildQuery(wClause);
390       System.out.println(query);
391       try {
392                     // read transfer from the database
393               Statement stmt = con.createStatement();
394               stmt.setQueryTimeout(5);
395               results = stmt.executeQuery(query);
396
397               while(results.next())
398                     {
399                     Transfer transferObj = buildTransfer(results);
400
401                           listT.add(transferObj);
402
403               }//end while
404
405               stmt.close();
406               if(retrieveAssociation)
407               {
408                     for(Transfer transferObj : listT)
409                     {
410                                 // Based upon Cage number retreive a complete
Cage
411                                 object from DBCage
412                           IFDBCage dbcage = new DBCage();
413
      transferObj.setCage(dbcage.findCage(transferObj.
414                                 getCage().getCageNo(), false));
415
416                                 // Based upon Disease Id retreive a complete
Disease
417                                 object from DBDisease
418                           if (transferObj.getBitedisease() != null)
419                           {
420                                 IFDBBiteDisease bitedisease = new
421                                 DBBiteDisease();
422                                 transferObj.setBitedisease(bitedisease.
423                                 searchBiteDiseaseById(transferObj.
424                                 getBitedisease().getDiseaseId(), false));
425                           }
426                           else
427                           {
428                                 IFDBPlasmaDisease plasmadisease = new
429                                 DBPlasmaDisease();
430                                 transferObj.setPlasmadisease(plasmadisease.
431                                 searchDiseaseById(transferObj.
432                                 getPlasmadisease().getDiseaseId(), false));
433                           }
434
435                                 // Based upon Employee Id retreive a complete
436                                 Employee object from DBEmployee
437                           IFDBEmployee employee = new DBEmployee();
438
      transferObj.setEmployee(employee.findEmployeeByID(
439
      transferObj.getEmployee().getEmployeeID(), false));
440
441                           }
```

```
442                 }//end if
443         }//end try
444         catch(Exception ex)
445         {
446                 System.out.println("Query exception - select: "+ ex.
447                 getMessage());
448                 ex.printStackTrace();
449           }
450
451           return listT;
452     }
453
454
455 /*      //michWere is used whenever we want to select more than
456 one row in transfer table.
457     private ArrayList<Transfer> miscWhere(String wClause,
458     boolean retrieveAssociation) throws Exception
459    {
460          ResultSet results;
461       ArrayList<Transfer> listT = new ArrayList <Transfer>();
462
463       String query = buildQuery(wClause);
464       System.out.println(query);
465       try {
466                       // read transfer from the database
467                 Statement stmt = con.createStatement();
468                 stmt.setQueryTimeout(5);
469                 results = stmt.executeQuery(query);
470
471                 while(results.next())
472                       {
473                     listT.add(buildTransfer(results));
474                 }
475                 stmt.close();
476
477                 if(retrieveAssociation)
478                 {
479                       for(Transfer trans : listT)
480                       {
481
482                           DBCage dbcage = new DBCage();
483                             int cageNo = trans.getCageNo();
484                             if (cageNo != dbcage.findCage(cageNo, false).
485                             getCageNo())
486                             {
487                                   throw new Exception("Case number:[" + cageNo
488                                   + "] not found.");
489                             }
490                             System.out.println("Cage Id is selected");
491
492                             DBEmployee dbemp = new DBEmployee();
493                             int empId = trans.getEmployeeId();
494                             if (empId != dbemp.findEmployeeByID(empId,
495                             false).getEmployeeID())
496                             {
497                                   throw new Exception("Employee Id:[" + empId +
498                                   "] not found.");
499                             }
500                             System.out.println("Employee Id is selected");
501
502                           int diseaseId = trans.getDiseaseId();
503                             if (diseaseId == 1600)
504                             {
505                                 DBBiteDisease dbbite = new DBBiteDisease()
```

```
506                                 ;
507                                 BiteDisease bitedisease = dbbite.
508                                 searchBiteDiseaseById(diseaseId, false);
509                                 System.out.println("BiteDisease Id is
510                                 selected");
511                             }
512                             else if (diseaseId == 1500)
513                             {
514                                   DBPlasmaDisease dbplasma = new
515                                   DBPlasmaDisease();
516                                   PlasmaDisease plasmadisease = dbplasma.
517                                   searchDiseaseById(diseaseId, false);
518                                 System.out.println("PlasmaDisease Id is
519                                 selected");
520                             }
521                             else
522                             {
523                                   throw new Exception("DiseaseId:["+ diseaseId
524                                   + "] not unknown.");
525                             }
526                         }
527                    }
528                    return listT;
529                }
530
531                catch(Exception e)
532                {
533                   System.out.println (e.getMessage());
534                   return null;
535                }
536          }*/
537
538
539     //method for selecting disease type.
540     private int getDiseaseValue(Transfer transferObj)
541     {
542           if(transferObj.getBitedisease() != null)
543                 return transferObj.getBitedisease().getDiseaseId();
544           else
545                 return transferObj.getPlasmadisease().getDiseaseId();
546     }
547
548       //method for getting today's date.
549       private String getTodaysDate()
550       {
551           Calendar calendar = Calendar.getInstance();
552           SimpleDateFormat dateFormat = new SimpleDateFormat(
553           "dd/MM/yyyy");
554           return dateFormat.format(calendar.getTime());
555       }
556
557 }// end of class DBTransfer.
```