```java
package com.company;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;

public class TimeDateCheck extends Thread{
    MysqlExecute mysqlExecute = new MysqlExecute();

    private final String localTime;
    private final int userId;
    private final int timeFormatLength;

    public TimeDateCheck(int userId) throws SQLException {
        this.userId = userId;
        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        LocalDateTime updatedTime = LocalDateTime.now();
        localTime = dateTimeFormatter.format(updatedTime);
        timeFormatLength = localTime.length();
    }

    private synchronized ResultSet getScheduleDate() throws SQLException { //return resultSet of TaskList table
        //synchronized to prevent the two Threads accessing ResultSet at the same time (just to make sure)
        ResultSet scheduledDate = mysqlExecute.mysqlQuery("SELECT manualScheduleDate, manualScheduleName, interestScheduleId FROM TaskList WHERE userId = " + userId);
        if (scheduledDate == null) {
```

```
31              throw new SQLException();
32          }
33          return scheduledDate;
34      }
35
36      public void printLocalTime() {
37          System.out.println("\nCurrent Time: " +
    localTime);
38      }
39
40      private boolean isScheduleOverdue(int
    firstSignificantUnit, int secondSignificantUnit,
    int thirdSignificantUnit){
41          if(firstSignificantUnit < 0){
42              return true;
43          } else if (firstSignificantUnit== 0){
44              if(secondSignificantUnit < 0){
45                  return true;
46              } else if(secondSignificantUnit == 0){
47                  return thirdSignificantUnit < 0;
48              } else{
49                  return false;
50              }
51          } else{
52              return false;
53          }
54      }
55
56      private final ArrayList<String>
    overdueSchedules = new ArrayList<>();
57      //have to be used in decrementOverdueTasks
    function also other than the function below
58      public synchronized void informSchedules() { //
    inform schedules in a day-to-day classification
59          ArrayList<Integer> interestScheduleIds =
    new ArrayList<>();
60
61          //variable i is for counting number of rows
     representing interest-based schedule;
62          int i = 0;
63          try {
```

```
64              ResultSet scheduledDate =
   getScheduleDate();
65              scheduledDate.next();
66              while ((scheduledDate.getString(3) !=
   null)) {
67                  interestScheduleIds.add(
   scheduledDate.getInt(3));
68                  scheduledDate.next();
69                  i++;
70                  if (scheduledDate.isAfterLast()) {
71                      break;
72                  }
73                   //i's initial value is 0 because
   i is incremented before going to examine the next
   row in this loop.
74              }
75              HashMap<String, String>
   interestScheduleHM = getInterestScheduleHM(
   interestScheduleIds);
76              scheduledDate.absolute(1); //reset the
    cursor
77
78              String localDateStr = localTime.
   substring(0, timeFormatLength - 9);//print
   interest-based schedules that are due.
79              if (scheduledDate.getString(3) != null
   ) { //make sure 1st row is interest-based schedule
80                  for (Map.Entry<String, String>
   interestScheduleHMMap : interestScheduleHM.
   entrySet()) {
81                      if (interestScheduleHMMap.
   getValue().substring(0, interestScheduleHMMap.
   getValue().length() - 9).equals(localDateStr)) {
82                          System.out.println("\t
   Based on your interest, It is " +
   interestScheduleHMMap.getKey() + " Today   \t");
83                      }
84                  }
85                  scheduledDate.absolute(i);//set
   the cursor so that it reads manually added
   schedules
```

```
 86                 }
 87
 88                 //store schedules due in arrayList and
     print them out
 89                 ArrayList<String> schedulesToday = new
     ArrayList<>();
 90                 ArrayList<String> schedulesTomorrow =
    new ArrayList<>();
 91                 ArrayList<String> schedulesNext7Days
     = new ArrayList<>();
 92                 ArrayList<String> schedulesThisWeek =
    new ArrayList<>();
 93
 94                 while (scheduledDate.next()) {
 95                     String dbScheduleDate =
    scheduledDate.getString(1);
 96                     String dbScheduleName =
    scheduledDate.getString(2);
 97
 98                     String localYearAndMonth =
    localTime.substring(0, timeFormatLength - 12);
 99                     String dbScheduleYearAndMonth =
    dbScheduleDate.substring(0, timeFormatLength - 12
    );
100
101                     boolean ifDateEquals =
    dbScheduleDate.substring(0, timeFormatLength - 9).
    equals(localDateStr);
102
103                     if (ifDateEquals) {
104                         schedulesToday.add("\t   You
    have \"" + dbScheduleName + "\" in " +
    dbScheduleDate + ". Have a great today!   ");
105                         //can be duplicated with
    overdue schedules
106
107                         int localHour = Integer.
    parseInt(localTime.substring(timeFormatLength - 8
    , timeFormatLength - 6));
108                         int dbHour = Integer.parseInt(
    dbScheduleDate.substring(timeFormatLength - 8,
```

```
108 timeFormatLength - 6));
109
110                     int localMin = Integer.
    parseInt(localTime.substring(timeFormatLength - 5
    , timeFormatLength - 3));
111                     int dbMin = Integer.parseInt(
    dbScheduleDate.substring(timeFormatLength - 5,
    timeFormatLength - 3));
112
113                     int localSec = Integer.
    parseInt(localTime.substring(timeFormatLength - 2
    ));
114                     int dbSec = Integer.parseInt(
    dbScheduleDate.substring(timeFormatLength - 2));
115
116                     if(isScheduleOverdue(dbHour -
    localHour, dbMin - localMin, dbSec - localSec)){
117                         overdueSchedules.add("\t
       You have \"" + dbScheduleName + "\" that was
    due " + dbScheduleDate + " today. Please do your
    work.   ");
118                     }
119
120                 } else {
121                     int localYear = Integer.
    parseInt(localTime.substring(0, timeFormatLength
     - 15));
122                     int dbYear = Integer.parseInt(
    dbScheduleDate.substring(0, timeFormatLength - 15
    ));
123
124                     int localMonth = Integer.
    parseInt(localTime.substring(timeFormatLength - 14
    , timeFormatLength - 12));
125                     int dbMonth = Integer.parseInt
    (dbScheduleDate.substring(timeFormatLength - 14,
    timeFormatLength - 12));
126
127                     int localDay = Integer.
    parseInt(localTime.substring(timeFormatLength - 11
    , timeFormatLength - 9));
```

```
128                     int dbDay = Integer.parseInt(
    dbScheduleDate.substring(timeFormatLength - 11,
    timeFormatLength - 9));
129
130                     int dayDiff = (dbDay -
    localDay);
131
132                     if (isScheduleOverdue(dbYear
     - localYear, dbMonth - localMonth, dbDay -
    localDay)) {
133                         //only overdue if all month
    , year, day is different but does not count hours
    , minutes, or seconds
134                         overdueSchedules.add("\t
    You have \"" + dbScheduleName + "\" that was due "
     + dbScheduleDate + ". Please do your work.   ");
135                     }
136                     //both conditions below are
    not overdue schedules
137                     else if (
    dbScheduleYearAndMonth.equals(localYearAndMonth
    ) && (dbDay - localDay) == 1) {
138                         schedulesTomorrow.add("\t
       You have \"" + dbScheduleName + "\" due " +
    dbScheduleDate + ". Have a great Tomorrow!   ");
139                     } else if (
    dbScheduleYearAndMonth.equals(localYearAndMonth
    ) && dayDiff <= 7) {
140                         //doesn't work in when the
     day goes to next month
141                         schedulesNext7Days.add("\t
       You have \"" + dbScheduleName + "\" due " +
    dbScheduleDate + ". Have a great day!   ");
142                     }
143
144                     Calendar localCalendar =
    Calendar.getInstance();
145                     localCalendar.set(localYear,
    localMonth, localDay, 0, 0, 0);
146
147                     Calendar dbCalendar = Calendar
```

```
147 .getInstance();
148                     dbCalendar.set(dbYear, dbMonth
    , dbDay, 0 ,0, 0);
149 
150                     //get week of year to get what
     schedules are in this week.
151                     int localWeekOfYear =
    localCalendar.get(Calendar.WEEK_OF_YEAR);
152                     int dbWeekOfYear = dbCalendar.
    get(Calendar.WEEK_OF_YEAR);
153                     //compare between local date
    and date in DB to find out if schedule is in this
    week.
154                     if (localWeekOfYear ==
    dbWeekOfYear && localYear == dbYear) {
155                         schedulesThisWeek.add("\t
       You have \"" + dbScheduleName + "\" in " +
    dbScheduleDate + ". Have a great week!   "); //can
     be duplicate of overdue schedules
156                     }
157                 }
158             }
159 
160             System.out.println("\n(complete the
    schedule to remove items below)\nSchedules for
    Today:");
161             for (String s : schedulesToday) {
162                 System.out.println(s);
163             }
164 
165             System.out.println("\nSchedules for
    Tomorrow:");
166             for (String s : schedulesTomorrow) {
167                 System.out.println(s);
168             }
169 
170             System.out.println("\nSchedules for
    the next 7 Days:");
171             for (String s : schedulesNext7Days) {
172                 System.out.println(s);
173             }
```

```
174
175             System.out.println("\nOverdue
    Schedules: (Please complete these works ASAP):");
176             for (String s : overdueSchedules) {
177                 System.out.println(s);
178             }
179
180             System.out.println("\nSchedules for
    this Week:");
181             for (String s : schedulesThisWeek) {
182                 System.out.println(s);
183             }
184         } catch (SQLException exception) {
185             exception.printStackTrace();
186         }
187     }
188     private synchronized static HashMap<String,
    String> getInterestScheduleHM(ArrayList<Integer>
    eventIds) throws SQLException{
189     //return arrayList that has name and date of
    interest based schedules from their event ids.
190         MysqlExecute mysqlExecute = new
    MysqlExecute();
191         HashMap<String,String> interestScheduleHM
     = new HashMap<>();
192         for (Integer eventId : eventIds) {
193             ResultSet interestSchedule =
    mysqlExecute.mysqlQuery("SELECT eventName,
    eventDate FROM InterestSchedule WHERE eventId = \"
    " + eventId + "\";");
194             interestSchedule.next();
195             interestScheduleHM.put(
    interestSchedule.getString(1), interestSchedule.
    getString(2));
196         }
197         return interestScheduleHM;
198     }
199
200     public synchronized void
    notifyGranularSchedules() throws SQLException { //
    notify schedules that in minute to minute
```

```
200 granularity.
201         //set the cursor where manually added
    schedules start
202         ResultSet scheduledDate = getScheduleDate
    ();
203         scheduledDate.next();
204         while ((scheduledDate.getString(3) != null
    )) {
205             scheduledDate.next();
206             if (scheduledDate.isAfterLast()) {
207                 break;
208             }
209         }
210         scheduledDate.previous();
211         while (scheduledDate.next()) {
212             String dbScheduleDate = scheduledDate.
    getString(1);
213             String dbScheduleName = scheduledDate.
    getString(2);
214
215             String localDateStr = localTime.
    substring(0, timeFormatLength - 9);
216             boolean ifDateEquals = dbScheduleDate.
    substring(0, timeFormatLength - 9).equals(
    localDateStr); //if year, month, day equals
217
218             int dbDateLen = timeFormatLength - 3;
219
220             int minLocal, minDB;
221             minLocal = Integer.parseInt(localTime.
    substring(timeFormatLength - 2));
222             minDB = Integer.parseInt(
    dbScheduleDate.substring(dbDateLen - 2, dbDateLen
    ));
223
224             int hourToMinLocal, hourToMinDB;
225             hourToMinLocal = Integer.parseInt(
    localTime.substring(timeFormatLength - 8,
    timeFormatLength - 6)) * 60;//hour value to
    minutes
226             hourToMinDB = Integer.parseInt(
```

```
226 dbScheduleDate.substring(dbDateLen - 5, dbDateLen
     - 3)) * 60;//hour value to minutes
227
228             int minTotalDiff = (minDB +
    hourToMinDB) - (minLocal + hourToMinLocal); //
    minute difference
229
230             if (minTotalDiff < 0) {
231                 continue;
232             }
233
234             if (dbScheduleDate.substring(0,
    timeFormatLength - 3).equals(localTime)) {
235                 System.out.println("\n   You are
    scheduled to have " + dbScheduleName + " Right Now
    !   ");
236             } else if (ifDateEquals && (
    minTotalDiff <= 10)) {
237                 System.out.println("\n   You are
    scheduled to have " + dbScheduleName + " within 10
     minutes. Hurry!   ");
238             } else if (ifDateEquals && (
    minTotalDiff <= 30)) {
239                 System.out.println("\n   You are
    scheduled to have " + dbScheduleName + " within 30
     minutes. Get your pace up!   ");
240             } else if (ifDateEquals && (
    minTotalDiff <= 60)) {
241                 System.out.println("\n   You are
    scheduled to have \"" + dbScheduleName + "\"
    within 1 hour. Work or prepare on it if you haven'
    t done it!   ");
242             } else if (ifDateEquals && (
    minTotalDiff <= 150)) {
243                 System.out.println("\n✔ You are
    scheduled to have \"" + dbScheduleName + "\"
    within 2 hour and 30 minutes. Hope that this
    reminds your schedule! ✔");
244             } else if (ifDateEquals && (
    minTotalDiff <= 720)) {
245                 System.out.println("\n   You are
```

```
245 scheduled to have \"" + dbScheduleName + "\"
    within 12 hours. It's just a reminder. you don't
    have to worry about it right know.   ");
246             }
247         }
248     }
249
250     public synchronized void
    periodicallyGiveBonusStreaks() throws SQLException
    {
251         ResultSet currentDateResultSet =
    mysqlExecute.mysqlQuery("SELECT CAST(NOW() AS DATE
    )");
252         if(currentDateResultSet == null){
253             throw new SQLException();
254         }
255         currentDateResultSet.next();
256         String currentDateString =
    currentDateResultSet.getString(1);
257
258         ResultSet databaseTimeNowString =
    mysqlExecute.mysqlQuery("SELECT todayDate FROM
    DateTable");
259         if(databaseTimeNowString == null){
260             throw new SQLException();
261         }
262         databaseTimeNowString.next();
263         if(!currentDateString.equals(
    databaseTimeNowString.getString(1))) {
264             System.out.println("\nThe Database's
    time agreement failed Time/TimeZone\n\texiting..."
    );
265             System.exit(0);
266         }
267
268         ResultSet
    databaseTimeYesterdayStringResultSet =
    mysqlExecute.mysqlQuery("SELECT yesterdayDate FROM
     DateTable");
269         if(databaseTimeYesterdayStringResultSet
     == null){
```

```
270             throw new SQLException();
271         }
272         databaseTimeYesterdayStringResultSet.next
    ();
273         String databaseTimeYesterdayString =
    databaseTimeYesterdayStringResultSet.getString(1);
274 
275         ResultSet
    userLastStreakDateRecordResultSet = mysqlExecute.
    mysqlQuery("SELECT lastStreakTimeStampDate FROM
    UserList WHERE userId = " + userId);
276         if(userLastStreakDateRecordResultSet ==
    null){
277             throw new SQLException();
278         }
279         userLastStreakDateRecordResultSet.next();
280         String userLastStreakDateRecord =
    userLastStreakDateRecordResultSet.getString(1);
281 
282         if(userLastStreakDateRecord.equals(
    databaseTimeYesterdayString)){//If the last time
    the user completed a task was yesterday,
283             giveBonusStreaks();
284         } else{
285             System.out.println("No task was
    completed yesterday    Please try to get more work
     done!");
286         }
287     }
288 
289     public synchronized void giveBonusStreaks()
    throws SQLException{
290         ResultSet previousDailyStreakResultSet =
    mysqlExecute.mysqlQuery("SELECT
    yesterdayDailyStreak FROM UserList WHERE userId
     = " + userId);
291         if(previousDailyStreakResultSet == null){
292             throw new SQLException();
293         }
294         previousDailyStreakResultSet.next();
295         int previousDailyStreak =
```

```
295 previousDailyStreakResultSet.getInt(1);
296
297         ResultSet currentDailyStreakResultSet =
    mysqlExecute.mysqlQuery("SELECT todayDailyStreak
    FROM UserList WHERE userId = " + userId);
298         if(currentDailyStreakResultSet == null){
299             throw new SQLException();
300         }
301         currentDailyStreakResultSet.next();
302         int currentDailyStreak =
    currentDailyStreakResultSet.getInt(1);
303
304         ResultSet userStreakAccumulatedResultSet
     = mysqlExecute.mysqlQuery("SELECT streak FROM
    UserList WHERE userId = " + userId);
305         if(userStreakAccumulatedResultSet == null
    ){
306             throw new SQLException();
307         }
308         userStreakAccumulatedResultSet.next();
309         int userStreakAccumulated =
    userStreakAccumulatedResultSet.getInt(1);
310
311         int newUserStreak = userStreakAccumulated
     + ((int) (Math.round(currentDailyStreak * 1.5)));
312         if(currentDailyStreak >
    previousDailyStreak){ //implicit double result
313             if(mysqlExecute.mysqlUpdate("UPDATE
    UserList SET streak = " + newUserStreak + " WHERE
    userId = " + userId) == -1){
314                 throw new SQLException();
315             }
316             System.out.println("\nYou have done
    more work than Yesterday! Keep up the good work!
    Bonus Streak for You   \n");
317
318             AccountManager accountManager = new
    AccountManager();
319             accountManager.updateStreakLevel(
    newUserStreak, userId);
320         }
```

```
321
322
323     }
324
325     public synchronized void
    decrementStreaksForOverdueTasks(int userId) throws
     SQLException{
326         ResultSet
    isOverdueDecrementAppliedResultSet = mysqlExecute.
    mysqlQuery("SELECT isOverdueDecrementApplied FROM
    DateTable");
327         if(isOverdueDecrementAppliedResultSet ==
    null){
328             throw new SQLException();
329         }
330         isOverdueDecrementAppliedResultSet.next();
331         boolean isOverdueDecrementAppliedBool =
    isOverdueDecrementAppliedResultSet.getBoolean(1);
332
333         if(!isOverdueDecrementAppliedBool) {
334             ResultSet streaksResultSet =
    mysqlExecute.mysqlQuery("SELECT streak FROM
    UserList WHERE userId = " + userId);
335             streaksResultSet.next();
336             int decrementingStreaks =
    overdueSchedules.size();
337             if (decrementingStreaks != 0) {
338                 int decrementedStreaks =
    streaksResultSet.getInt(1) - decrementingStreaks;
339
340                 if (mysqlExecute.mysqlUpdate("
    UPDATE UserList SET streak = " +
    decrementedStreaks + " WHERE userId = " + userId
    ) == -1) {
341                     throw new SQLException();
342                 }
343                 System.out.println("Lost the
    streak down to " + Math.abs(decrementedStreaks) +
    " Streaks! To not lose Streaks anymore, please
    complete overdue tasks!");
344             }
```

```
345         }
346     }
347 
348     public synchronized void updateDailyStreak(int
     userId) throws SQLException{
349         ResultSet databaseTimeNowResultSet =
    mysqlExecute.mysqlQuery("SELECT todayDate FROM
    DateTable");
350         if(databaseTimeNowResultSet == null){
351             throw new SQLException();
352         }
353         databaseTimeNowResultSet.next();
354         String databaseTimeNowString =
    databaseTimeNowResultSet.getString(1);
355 
356         ResultSet
    databaseLastDailyStreakDateResultSet =
    mysqlExecute.mysqlQuery("SELECT
    lastStreakTimeStampDate FROM UserList WHERE userId
     = " + userId);
357         if(databaseLastDailyStreakDateResultSet
     == null){
358             throw new SQLException();
359         }
360         databaseLastDailyStreakDateResultSet.next
    ();
361         String databaseLastDailyStreakDateString
     = databaseLastDailyStreakDateResultSet.getString(
    1);
362 
363         if(!databaseLastDailyStreakDateString.
    equals(databaseTimeNowString)){
364             ResultSet newTodayResultSet =
    mysqlExecute.mysqlQuery("SELECT CAST(NOW() AS DATE
    )");
365             if(newTodayResultSet == null){
366                 throw new SQLException();
367             }
368             newTodayResultSet.next();
369             String newTodayString =
    newTodayResultSet.getString(1);
```

```
370             if(mysqlExecute.mysqlUpdate("UPDATE
    UserList SET lastStreakTimeStampDate = \"" +
    newTodayString + "\" WHERE userId = " + userId
    ) == -1){
371                 throw new SQLException();
372             }
373         }
374     }
375 }
376
```