```java
package com.company;

import java.security.NoSuchAlgorithmException;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Locale;
import java.util.Scanner;
import java.util.concurrent.atomic.AtomicReference;

public class AccountManager {
    private final Scanner keyboard = new Scanner(System.in);
    private String username; //cannot be declared final
    private final MysqlExecute mysqlExecute;

    public AccountManager(){
        mysqlExecute = new MysqlExecute();
    }
    public String getUsername() {
        return username;
    }
    public int getUserId(String username){
        int userId = 0;
        try{
            ResultSet resultSet = mysqlExecute.mysqlQuery("SELECT userId FROM UserList WHERE username=\"" + username + "\";");
            if(resultSet == null){
                throw new SQLException();
            }
            resultSet.next();
            userId = resultSet.getInt(1);
        } catch (SQLException exception){
            exception.printStackTrace();
        }
        return userId;
    }
```

```
38
39      public void signUp() throws
    NoSuchAlgorithmException {
40          final String signingUsername, password,
    passwordHint, userAddStatement;
41          final int interestedBranchId;
42          System.out.println("Please enter your
    username:");
43          signingUsername = keyboard.nextLine();
44          if(signingUsername.equals("admin")){
45              System.out.println("Cannot sign-up to
    Administrator account in client version.");
46              return;
47          }
48
49          System.out.println("Please enter your
    password:");
50          StringBuilder saltString = Sha256Hash.
    generateSaltString();
51          password = Sha256Hash.doubleSHA256(keyboard
    .nextLine(), saltString);
52
53          System.out.println("Please confirm (re-
    enter) your password:");
54          final String passwordConfirm = Sha256Hash.
    doubleSHA256(keyboard.nextLine(), saltString);
55          if(!password.equals(passwordConfirm)){
56              System.out.println("Password Does not
    match. Please Try Again.");
57              return;
58          }
59
60          System.out.println("Please enter your
    password hint (optional):");
61          passwordHint = keyboard.nextLine();
62          System.out.println("Options available: 1.
    Holidays or Other Special Days\t2. Computers\t3.
    Politics");
63          System.out.println("Please choose your
    interested subject:");
64          switch (keyboard.nextLine().toLowerCase(
```

```
64 Locale.ROOT).replaceAll("\\s","").toLowerCase(
   Locale.ROOT)){
65             case "holidays", "otherspecialdays", "
   holidaysorotherspecialdays", "1" ->
   interestedBranchId = 1;
66             case "computers","computer","2" ->
   interestedBranchId = 2;
67             case "politics", "politic", "3" ->
   interestedBranchId = 3;
68             default -> {
69                 System.out.println("Invalid option
   .");
70                 return;
71             }
72         }
73
74         try {
75             ResultSet userList = mysqlExecute.
   mysqlQuery("SELECT username FROM UserList");
76             if(userList == null){
77                 throw new SQLException();
78             }
79             while(userList.next()){
80                 if(signingUsername.equals(userList
   .getString(1))){
81                     System.out.println("Username
   already exists!");
82                     return;
83                 }
84             }
85             userAddStatement = "INSERT INTO
   UserList(username, password, salt, passwordHint,
   interestedBranchId, streak, streakLevel) VALUES (
   \"" + signingUsername + "\",\"" + password + "\",
   \"" + saltString + "\",\"" + passwordHint + "\","
    + interestedBranchId + ", 0, \"Newbie\");";
86             int resultValue = mysqlExecute.
   mysqlUpdate(userAddStatement);
87             if(resultValue == -1){
88                 throw new SQLException();
89             } else if(!
```

```
 89 addInterestedSchedulesToTaskList(getUserId(
    signingUsername),interestedBranchId)){
 90                 System.out.println("Failed to
    allocate automatically-generated Interest-based
    schedules.");
 91             } else{
 92                 System.out.println("User " +
    signingUsername + " successfully registered.");
 93             }
 94         } catch (SQLException exception){
 95             exception.printStackTrace();
 96         }
 97     }
 98     private boolean
    addInterestedSchedulesToTaskList(int userId, int
    branchId){
 99         if(userId == 1){
100             System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
101             return false;
102         }
103         try {
104             ArrayList<Integer> eventIds = new
    ArrayList<>();
105             ResultSet eventsSet = mysqlExecute.
    mysqlQuery("SELECT eventId, eventName, eventDate
    FROM InterestSchedule WHERE branchId = " +
    branchId + ";");
106             if (eventsSet == null) {
107                 return false;
108             }
109             while (eventsSet.next()) {
110                 eventIds.add(eventsSet.getInt(1));
111             }
112             for (int eventId : eventIds) {
113                 if (mysqlExecute.mysqlUpdate("
    INSERT INTO TaskList(userId, interestScheduleId)
    VALUES (" + userId + "," + eventId +  ");") == -1
    ) {
114                     return false;
```

```
115                 }
116             }
117         } catch (SQLException exception){
118             return false;
119         }
120         return true;
121     }
122 
123     public void removeAccount(int userId) throws
    SQLException {
124         if(userId == 1){
125             System.out.println("You cannot delete
    Administrator's account on the client version.");
126             return;
127         }
128         if(mysqlExecute.mysqlUpdate("DELETE FROM
    UserList WHERE userId = " + userId + ";") == -1){
129             throw new SQLException();
130         }
131         System.out.println("User \"" + username +
    "\" successfully deleted. Hope we see again.");
132     }
133 
134     public boolean login() throws SQLException,
    NoSuchAlgorithmException{
135         final String password, usernameTemp; //
    usernameTemp for security reasons
136         System.out.println("Please enter your
    username:");
137         usernameTemp = keyboard.nextLine();
138         if(usernameTemp.equals("admin")){
139             System.out.println("Cannot login as
    Administrator in a client version.");
140             return false;
141         }
142 
143         if(usernameTemp.equals("")){
144             System.out.println("Username cannot be
     a Blank");
145             return false;
146         }
```

```
147
148         String saltString = Sha256Hash.
    getSaltString(usernameTemp);
149         boolean isUsernameWrong = saltString.
    equals("-1");
150
151         System.out.println("Please enter your
    password:");
152         password = Sha256Hash.doubleSHA256(
    keyboard.nextLine(), new StringBuilder(saltString
    ));
153
154         try {
155             ResultSet resultSet = mysqlExecute.
    mysqlQuery("SELECT username, password FROM
    UserList;");
156             if(resultSet == null){
157                 return false;
158             }
159             if(!isUsernameWrong) {
160                 while (resultSet.next()) {
161                     if (resultSet.getString(1).
    equals(usernameTemp) && resultSet.getString(2).
    equals(password)) {
162                         username = usernameTemp;
163                         System.out.println("
    Welcome " + username + "!\tNow you are logged in
    as " + username + ".");
164                         return true;
165                     }
166                 } //implicit else -> loginSucceed
     = false
167             }
168             System.out.println("Login failed.\n
    Please check your username and password.");
169         } catch (SQLException exception) {
170             exception.printStackTrace();
171         }
172         return false;
173     }
174
```

```
175     public void printStreakAndStreakLevel(int
    userId) throws SQLException {
176         ResultSet streaksAndItsNumberResultSet =
    mysqlExecute.mysqlQuery("SELECT userId, streak,
    streakLevel FROM UserList");
177         if(streaksAndItsNumberResultSet == null){
178             throw new SQLException();
179         }
180         streaksAndItsNumberResultSet.next();
181         while(streaksAndItsNumberResultSet.next
    ()){
182             if(userId ==
    streaksAndItsNumberResultSet.getInt(1)){
183                 System.out.println("You have " +
    streaksAndItsNumberResultSet.getInt(2) + " streaks
    .\t\tYou are currently '"+
    streaksAndItsNumberResultSet.getString(3) + "'
    Level   ");
184                 return;
185             }
186         }
187
188     }
189
190     public boolean logout(){ //erase user's data
    for security. (final and method scope)
191         username = "";
192         return true;
193     }
194
195     interface EnterDateInterface {
196         boolean formatStringDate(String month);
197         boolean formatNumberedDate();
198         boolean formatThisDateOfWeekDate();
199         boolean formatNextDateOfWeekDate();
200         boolean formatTomorrowDate();
201         boolean formatTodayDate();
202
203     }
204
205     private String enterDateSchedule(){ //function
```

```
205  for taking and processing user-friendly date
    input
206
207          System.out.println("Proper formats (put
    only 1 digit of month if the month is before
    november)\t(enter \"tmr\" or \"today\" for
    scheduling tomorrow and today each):\nString
    format: MM dd YYYY HH:mm:ss\texample: feb 15 2021
    01:02:03  or\nNumber format: MM/dd/YYYY HH:mm:ss\t
    example: 2/15/2021 01:02:03\nDate of Week format
    : (put \"nxt\" to indicate next week) +
    abbreviated_string of date_of_week\texample: \"nxt
     fri\" & \"tue\"");
208
209          AtomicReference<String> date = new
    AtomicReference<>();
210          boolean isFormattingSuccess = false;
211
212          do {
213              date.set(keyboard.nextLine().
    toLowerCase(Locale.ROOT));
214              int dateEnteredLength = date.get().
    length();
215              EnterDateInterface formatDateNumber =
    new EnterDateInterface() {
216
217                  final SimpleDateFormat timeFormat
     = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
218
219                 @Override
220                  public boolean formatStringDate(
    String month) {
221                      //format string form of date
222                      //form of string: Ex. feb 15
    2021 HH:mm:ss
223
224                      if (dateEnteredLength == 20
    ) { //if day number is 2 digit
225                          //negative string index
    value is the same for both length after the
    varying digit day number
```

```
226                         date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-" + month + "-" + date.
    get().substring(dateEnteredLength - 16,
    dateEnteredLength - 14) +
227                                 " " + date.get().
    substring(dateEnteredLength - 8, dateEnteredLength
     - 6) + ":" + date.get().substring(
    dateEnteredLength - 5, dateEnteredLength - 3) +
    ":"
228                                 + date.get().
    substring(dateEnteredLength - 2));
229                     } else if (dateEnteredLength
     == 19) { //if day number is 1 digit
230                         date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-" + month + "-0" + date
    .get().charAt(4) +
231                                 " " + date.get().
    substring(dateEnteredLength - 8, dateEnteredLength
     - 6) + ":" + date.get().substring(
    dateEnteredLength - 5, dateEnteredLength - 3) +
    ":"
232                                 + date.get().
    substring(dateEnteredLength - 2));
233                     } else {
234                         System.out.println("The
    entered date does not follow the format. Please
    recheck");
235                         return false;
236                     }
237                     return true;
238                 }
239
240                 @Override
241                 public boolean formatNumberedDate
    () {
242                     //format number form of date
243                     //form of number: Ex. 2/15/
    2021 HH:mm:ss
244                     String monthDigit = date.get
```

```
244 ().substring(0, 2);
245                     String dayDigit = date.get().
    substring(2, 5);
246                     int monthDigitCount = 0;
247                     int dayDigitCount = 0;
248 
249                     for(int i = 0; i < 2; i++){
250                         int monthCharacter =
    monthDigit.charAt(i);
251                         int dayCharacter =
    dayDigit.charAt(i);
252                         if(Character.toString(
    monthCharacter).matches("-?\\d+")){
253                             monthDigitCount++;
254                         }
255                         if(Character.toString(
    dayCharacter).matches("-?\\d+")){
256                             dayDigitCount++;
257                         }
258                     }
259 
260                     if(monthDigitCount == 2){ //if
     month digit is 2...
261                         if(dayDigitCount == 2 &&
    dateEnteredLength == 19){ //and day is digit is 2
262                             date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-" + date.get().
    substring(0 , dateEnteredLength - 17) + "-" + date
    .get().substring(dateEnteredLength - 16,
    dateEnteredLength - 14) +
263                                     " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
     - 3) + ":"
264                                     + date.get().
    substring(dateEnteredLength - 2));
265                         } else if(dayDigitCount
     == 1 && dateEnteredLength == 18){ //and day is
    digit 1
```

```
266                                 date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-" + date.get().
    substring(0 , dateEnteredLength - 16) + "-0" +
    date.get().charAt(3) +
267                                         " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
     - 3) + ":"
268                                         + date.get().
    substring(dateEnteredLength - 2));
269                             } else {
270                                 System.out.println("
    The entered date does not follow the format.
    Please recheck");
271                                 return false;
272                             }
273                         } else if(monthDigitCount == 1
    ){ //if month digit is 1...
274                             if(dayDigitCount == 2 &&
    dateEnteredLength == 18){ //and day is digit 2
275                                 date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-0" + date.get().charAt(
    0) + "-" + date.get().substring(dateEnteredLength
     - 16, dateEnteredLength - 14) +
276                                         " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
     - 3) + ":"
277                                         + date.get().
    substring(dateEnteredLength - 2));
278                             } else if(dayDigitCount
     == 1 && dateEnteredLength == 17){ //and day is
    digit 1
279                                 date.set(date.get().
    substring(dateEnteredLength - 13,
    dateEnteredLength - 9) + "-0" + date.get().charAt(
    0) + "-0" + date.get().charAt(2) +
```

```
280                                         " " + date.get
    ().substring(dateEnteredLength - 8,
    dateEnteredLength - 6) + ":" + date.get().
    substring(dateEnteredLength - 5, dateEnteredLength
     - 3) + ":"
281                                             + date.get().
    substring(dateEnteredLength - 2));
282                             } else {
283                                 System.out.println("
    The entered date does not follow the format.
    Please recheck");
284                                 return false;
285                             }
286                         }  else {
287                             System.out.println("The
    entered date does not follow the format. Please
    recheck");
288                             return false;
289                         }
290                         return true;
291                     }
292
293                     @Override
294                     public boolean
    formatThisDateOfWeekDate(){
295                         Calendar timeNowCal = Calendar
    .getInstance();
296                         int timeNowDayOfWeek =
    timeNowCal.get(Calendar.DAY_OF_WEEK);
297
298                         Calendar timeScheduled =
    Calendar.getInstance();
299                         timeScheduled.set(Calendar.
    HOUR, 11);
300                         timeScheduled.set(Calendar.
    MINUTE, 59);
301                         timeScheduled.set(Calendar.
    SECOND, 59);
302
303                         timeScheduled.set(Calendar.
    DAY_OF_WEEK, timeNowDayOfWeek);
```

```
304
305                     date.set(timeFormat.format(
    timeScheduled.getTime()));
306                     return true;
307                 }
308
309                 @Override
310                 public boolean
    formatNextDateOfWeekDate(){
311                     Calendar timeNowCal = Calendar
    .getInstance();
312                     int timeNowWeekOfYear =
    timeNowCal.get(Calendar.WEEK_OF_YEAR);
313
314                     Calendar timeScheduled =
    Calendar.getInstance();
315                     timeScheduled.set(Calendar.
    HOUR, 11);
316                     timeScheduled.set(Calendar.
    MINUTE, 59);
317                     timeScheduled.set(Calendar.
    SECOND, 59);
318
319                     timeScheduled.set(Calendar.
    WEEK_OF_YEAR, timeNowWeekOfYear + 1);
320
321                     String dateOfWeekEntered =
    date.get().substring(date.get().length() - 4).
    strip();
322                     switch (dateOfWeekEntered) {
323                         case "mon" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 2);
324
325                         case "tues", "tue" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 3);
326
327                         case "wed" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 4);
328
329                         case "thu", "thur", "hurs"
     -> timeScheduled.set(Calendar.DAY_OF_WEEK, 5);
```

```
330
331                         case "fri" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 6);
332
333                         case "sat" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 7);
334
335                         case "sun" ->
    timeScheduled.set(Calendar.DAY_OF_WEEK, 1);
336                     }
337
338                     date.set(timeFormat.format(
    timeScheduled.getTime()));
339                     return true;
340                 }
341
342                 @Override
343                 public boolean formatTodayDate(){
344                     Calendar timeScheduled =
    Calendar.getInstance();
345                     timeScheduled.set(Calendar.
    HOUR, 11);
346                     timeScheduled.set(Calendar.
    MINUTE, 59);
347                     timeScheduled.set(Calendar.
    SECOND, 59);
348
349                     date.set(timeFormat.format(
    timeScheduled.getTime()));
350                     return true;
351                 }
352
353                 @Override
354                 public boolean formatTomorrowDate
    (){
355                     Calendar timeScheduled =
    Calendar.getInstance();
356                     timeScheduled.set(Calendar.
    HOUR, 11);
357                     timeScheduled.set(Calendar.
    MINUTE, 59);
```

```
358                     timeScheduled.set(Calendar.
    SECOND, 59);
359                     timeScheduled.set(Calendar.
    DAY_OF_MONTH, timeScheduled.get(Calendar.
    DAY_OF_MONTH) + 1);
360
361                     date.set(timeFormat.format(
    timeScheduled.getTime()));
362                     return true;
363                 }
364             };
365
366
367             if (!(date.get().substring(0,1).
    matches("-?\\d+"))) {
368                 switch (date.get().substring(0, 3
    )) {
369                     case "tmr", "tom" ->
    isFormattingSuccess = formatDateNumber.
    formatTomorrowDate();
370
371                     case "tod" ->
    isFormattingSuccess = formatDateNumber.
    formatTodayDate();
372
373                     case "jan" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("01");
374
375                     case "feb" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("02");
376
377                     case "mar" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("03");
378
379                     case "apr" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("04");
380
```

```
381                     case "may" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("05");
382
383                     case "jun" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("06");
384
385                     case "jul" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("07");
386
387                     case "aug" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("08");
388
389                     case "sep" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("09");
390
391                     case "oct" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("10");
392
393                     case "nov" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("11");
394
395                     case "dec" ->
    isFormattingSuccess = formatDateNumber.
    formatStringDate("12");
396
397                     case "nxt", "nex" ->
    isFormattingSuccess = formatDateNumber.
    formatNextDateOfWeekDate();
398
399                     case "mon", "tue", "wed", "thu
    ", "fri", "sat", "sun" -> isFormattingSuccess =
    formatDateNumber.formatThisDateOfWeekDate();
400                     default -> {
401                         //implicit
```

```
401 isFormattingSuccess = false
402                     }
403                 }
404             } else{
405                 isFormattingSuccess =
    formatDateNumber.formatNumberedDate();
406             }
407
408         } while (!isFormattingSuccess);
409
410         return String.valueOf(date);
411     }
412
413     public void addManualSchedule(int userId)
    throws SQLException {
414         if(userId == 1){
415             System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
416             return;
417         }
418         String descrpSchedule;
419         final String dateSchedule;
420         final String manualScheduleAddStatement;
421         System.out.println("To add a schedule,
    first provide a description of the event:");
422         descrpSchedule = keyboard.nextLine();
423
424         System.out.println("Second, provide the
    due date of the event:");
425         dateSchedule = enterDateSchedule();
426
427         descrpSchedule = SchedulePriority.
    addPriority(descrpSchedule); //assign priority
428
429         manualScheduleAddStatement = "INSERT INTO
    TaskList(userId,manualScheduleName,
    manualScheduleDate) VALUES (" + userId + ",\"" +
    descrpSchedule + "\",\"" + dateSchedule + "\");";
430         int resultValue = mysqlExecute.mysqlUpdate
    (manualScheduleAddStatement);
```

```
431             if (resultValue == -1) {
432                 throw new SQLException();
433             } else{
434                 System.out.println("Schedule " +
    descrpSchedule + " successfully added.");
435             }
436         }
437 
438         public void editSchedule(int userId) throws
    SQLException, InterruptedException {
439             if(userId == 1){
440                 System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
441                 return;
442             }
443 
444             printTasks(userId,username); //give info
    about schedules
445 
446             System.out.println("Do you want to Update/
    Edit a schedule by its Task ID or its  Name?");
447             String removeMethodOption = keyboard.
    nextLine().replaceAll("\\s","").toLowerCase(Locale
    .ROOT);
448 
449             final String newScheduleDate;
450             String newScheduleName;
451 
452             if(removeMethodOption.equals("taskid")){
453                 System.out.println("Please select from
     schedules above: (by its Task ID)\n(caution:
    cannot delete interest-based schedule by its Task
    ID)");
454                 String enteredTaskId = keyboard.
    nextLine();
455 
456                 System.out.println("Enter the new
    description for the new schedule: ");
457                 newScheduleName = keyboard.nextLine
    (); //string is an object so value retains
```

```
458             System.out.println("Enter the new due
    date for the new schedule: ");
459             newScheduleDate = enterDateSchedule();
460 
461             newScheduleName = SchedulePriority.
    addPriority(newScheduleName); //assign priority
462 
463             ResultSet taskIdsDB = mysqlExecute.
    mysqlQuery("SELECT taskId FROM TaskList WHERE
    userId = " + userId);
464             if(taskIdsDB == null){
465                 throw new SQLException();
466             }
467 
468             while(taskIdsDB.next()){
469                 if(enteredTaskId.equals(taskIdsDB.
    getString(1))){
470                     //for checking same
    information update twice
471                     ResultSet
    manualScheduleNameAndDate= mysqlExecute.mysqlQuery
    ("SELECT manualScheduleName, manualScheduleDate
    FROM TaskList WHERE TaskId = \"" + enteredTaskId
     + "\"");
472                     if(manualScheduleNameAndDate
     == null){
473                         throw new SQLException();
474                     }
475                     while (
    manualScheduleNameAndDate.next()){
476                         if(
    manualScheduleNameAndDate.getString(1).equals(
    newScheduleName) && manualScheduleNameAndDate.
    getString(2).equals(newScheduleDate)){
477                             System.out.println("
    Cannot update a schedule with duplicate or same
    information.");
478                             return;
479                         }
480                     }
481 
```

```
482                     if(mysqlExecute.mysqlUpdate("
    UPDATE TaskList SET manualScheduleName = \"" +
    newScheduleName + "\", manualScheduleDate = \"" +
    newScheduleDate + "\" WHERE userId = " + userId +
    " AND taskId = " + enteredTaskId) == -1){
483                         throw new SQLException();
484                     }
485                     System.out.println("Schedule
    information successfully changed.");
486                     return;
487                 }
488             }
489         } else if(removeMethodOption.equals("name"
    )) {
490             System.out.println("Please select from
     schedules above: (by its Name)\n(caution: cannot
    delete interest-based schedule by its Task ID)");
491             String enteredName = keyboard.nextLine
    ();
492
493             System.out.println("Enter the new
    description for the new schedule: ");
494             newScheduleName = keyboard.nextLine
    (); //string is an object so value retains
495             System.out.println("Enter the new due
    date for the new schedule: ");
496             newScheduleDate = keyboard.nextLine();
497
498             ResultSet namesDB = mysqlExecute.
    mysqlQuery("SELECT manualScheduleName FROM
    TaskList WHERE userId = " + userId);
499             if (namesDB == null) {
500                 throw new SQLException();
501             }
502
503             while (namesDB.next()) {
504                 if (enteredName.equals(namesDB.
    getString(1))) {
505                     //for checking same
    information update twice
506                     ResultSet
```

```
506 manualScheduleNameAndDate = mysqlExecute.
    mysqlQuery("SELECT manualScheduleName,
    manualScheduleDate FROM TaskList WHERE
    manualScheduleName = \"" + enteredName + "\"");
507                     if (manualScheduleNameAndDate
     == null) {
508                         throw new SQLException();
509                     }
510                     while (
    manualScheduleNameAndDate.next()) {
511                         if (
    manualScheduleNameAndDate.getString(1).equals(
    newScheduleName) && manualScheduleNameAndDate.
    getString(2).equals(newScheduleDate)) {
512                             System.out.println("
    Cannot update a schedule with duplicate or same
    information.");
513                             return;
514                         }
515                     }
516
517                     if (mysqlExecute.mysqlUpdate("
    UPDATE TaskList SET manualScheduleName = \"" +
    newScheduleName + "\", manualScheduleDate = \"" +
    newScheduleDate + "\" WHERE userId = " + userId +
    " AND manualScheduleName = \"" + enteredName + "\"
    ") == -1) {
518                         throw new SQLException();
519                     }
520                     System.out.println("Schedule
    information successfully changed.");
521                     return;
522                 }
523             }
524
525         } else{
526             System.out.println("Wrong Option.");
527             return;
528         }
529         System.out.println("Did not find a
    matching schedule; please Try Again.");
```

```
530         //function should be terminated if there
    is a match or selection method is wrong (task id
    or name)
531     }
532
533     public void completeOrRemoveSchedule(int
    userId) throws SQLException, InterruptedException
     {
534         if(userId == 1){
535             System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
536             return;
537         }
538         printTasks(userId,username); //give info
    about schedules
539
540         System.out.println("Do you want to remove
    a schedule by its Task ID or its Name?");
541         String removeMethodOption = keyboard.
    nextLine().replaceAll("\\s","").toLowerCase(Locale
    .ROOT);
542         if(removeMethodOption.equalsIgnoreCase("
    taskid")){
543             System.out.println("Please select from
     schedules above: (by its Task ID)\n(caution:
    cannot delete interest-based schedule by its Task
    ID)");
544             String removingTaskId = keyboard.
    nextLine();
545             if(mysqlExecute.mysqlUpdate("DELETE
    FROM TaskList WHERE userId = " + userId + " AND
    taskId = " + removingTaskId + ";") == -1){
546                 throw new SQLException();
547             }
548         } else if(removeMethodOption.
    equalsIgnoreCase("name")){
549             System.out.println("Please select from
     schedules above: (by its Name)\n(caution: cannot
    delete interest-based schedule by its Task ID)");
550             String removingTaskId = keyboard.
```

```
550 nextLine();
551             if(mysqlExecute.mysqlUpdate("DELETE
    FROM TaskList WHERE userId = " + userId + " AND
    manualScheduleNam" +
552                     "e = \"" + removingTaskId + "
    \";") == -1){
553                 throw new SQLException();
554             }
555         } else {
556             System.out.println("Wrong option.");
557             return;
558         }
559         System.out.println("Task completed/deleted
    .");
560
561         ResultSet streaksResultSet = mysqlExecute.
    mysqlQuery("SELECT streak FROM UserList WHERE
    userid = " + userId);
562         if(streaksResultSet == null){
563             throw new SQLException();
564         }
565         streaksResultSet.next();
566         int newStreaks = streaksResultSet.getInt(1
    ) + 1;
567
568         ResultSet todayDailyStreakResultSet =
    mysqlExecute.mysqlQuery("SELECT todayDailyStreak
    FROM UserList WHERE userid = " + userId);
569         if(todayDailyStreakResultSet == null){
570             throw new SQLException();
571         }
572         todayDailyStreakResultSet.next();
573         int newTodayDailyStreak =
    todayDailyStreakResultSet.getInt(1) + 1;
574
575         if(mysqlExecute.mysqlUpdate("UPDATE
    UserList SET streak = " + newStreaks + " WHERE
    userid = " + userId) == -1 ||
576                 mysqlExecute.mysqlUpdate("UPDATE
    UserList SET todayDailyStreak = " +
    newTodayDailyStreak + " WHERE userId = " + userId
```

```
576 ) == -1)//update last time task was completed
577                 {
578                     throw new SQLException();
579             }
580 
581             updateStreakLevel(newStreaks, userId);
582 
583             System.out.println("\tOne more streak   "
    );
584         }
585 
586         public void updateStreakLevel(int newStreaks,
    int userId) throws SQLException {
587             MysqlExecute mysqlExecute = new
    MysqlExecute();
588             String newStreakLevel;
589             //optimize the code so that don't do the
    database operation if level is same; also set
    specific scope for levels
590             if(newStreaks <= 15){
591                 newStreakLevel = "Newbie";
592             } else if(newStreaks <= 55){
593                 newStreakLevel = "Apprentice";
594             } else if(newStreaks <= 125){
595                 newStreakLevel = "Seasoned";
596             } else if(newStreaks <= 250){
597                 newStreakLevel = "Cerebral";
598             } else if(newStreaks <= 550){
599                 newStreakLevel = "Master";
600             } else{
601                 newStreakLevel = "Legend";
602             }
603             if(mysqlExecute.mysqlUpdate("UPDATE
    UserList SET streakLevel = \"" + newStreakLevel +
    "\" WHERE userId = " + userId) == -1){
604                 throw new SQLException();
605             }
606         }
607 
608         public void changeUsername(int userId) throws
    NoSuchAlgorithmException, SQLException {
```

```
609         if(userId == 1){
610             System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
611             return;
612         }
613         while(true) {
614             System.out.println("Please enter your
    password for security check.");
615             StringBuilder saltString = new
    StringBuilder(Sha256Hash.getSaltString(username));
616             final String passwordEntered =
    Sha256Hash.doubleSHA256(keyboard.nextLine(),
    saltString);
617 
618             ResultSet password = mysqlExecute.
    mysqlQuery("SELECT password FROM UserList WHERE
    userId = " + userId);
619             if(password == null){
620                 throw new SQLException();
621             }
622             password.next();
623             if(!passwordEntered.equals(password.
    getString(1))){
624                 System.out.println("Security check
     failed; please enter valid password.");
625                 continue;
626             }
627             System.out.println("Please enter your
    new username:");
628             String newUsername = keyboard.nextLine
    ();
629 
630             System.out.println("Please confirm (re
    -type) your new username:");
631             String confirmNewUsername = keyboard.
    nextLine();
632 
633             if(newUsername.equals("") ||
    confirmNewUsername.equals("")){
634                 System.out.println("Username
```

```
634 cannot be a Blank");
635                     return;
636                 }
637 
638                 ResultSet currentUsername =
    mysqlExecute.mysqlQuery("SELECT username FROM
    UserList WHERE userId = " + userId);
639                 if(currentUsername == null){
640                     throw new SQLException();
641                 }
642                 currentUsername.next();
643                 if(newUsername.equals(currentUsername.
    getString(1))){
644                     System.out.println("Cannot change
    the username if the new username is equal to the
    current username.");
645                     continue;
646                 }
647                 if (newUsername.equals(
    confirmNewUsername)) {
648                     if(mysqlExecute.mysqlUpdate("
    UPDATE UserList SET username = \"" + newUsername
     + "\" WHERE userId = " + userId + ";") == -1){
649                         throw new SQLException();
650                     }
651                     System.out.println("Username
    successfully changed.");
652                     return;
653                 } else {
654                     System.out.println("New Username
    does not match; please Try Again.");
655                 }
656             }
657         }
658         public void changePassword(int userId) throws
    NoSuchAlgorithmException, SQLException {
659             if(userId == 1){
660                 System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
661                 return;
```

```
662         }
663         while (true) {
664             System.out.println("Please enter your
    current password for security check.");
665             StringBuilder saltString = new
    StringBuilder(Sha256Hash.getSaltString(username));
666             final String currentPasswordCheck =
    Sha256Hash.doubleSHA256(keyboard.nextLine(),
    saltString);
667             ResultSet currentPassword =
    mysqlExecute.mysqlQuery("SELECT password FROM
    UserList WHERE userId = " + userId);
668             if(currentPassword == null){
669                 throw new SQLException();
670             }
671             currentPassword.next();
672             if(!currentPasswordCheck.equals(
    currentPassword.getString(1))){
673                 System.out.println("Security check
     failed; please enter valid password.");
674                 return;
675             }
676             System.out.println("Please enter your
    new password:");
677             StringBuilder newSaltString =
    Sha256Hash.generateSaltString();
678             final String newPassword = Sha256Hash.
    doubleSHA256(keyboard.nextLine(), newSaltString);
679             System.out.println("Please confirm (
    retype) your new password:");
680             final String confirmNewPassword =
    Sha256Hash.doubleSHA256(keyboard.nextLine(),
    newSaltString);
681             if(newPassword.equals(currentPassword.
    getString(1))){
682                 System.out.println("Cannot change
    the password if the new password is equal to the
    current password.");
683                 continue;
684             }
685             if (newPassword.equals(
```

```
685 confirmNewPassword)) {
686                 System.out.println("Please enter
    your new password hint:");
687                 String passwordHint = keyboard.
    nextLine();
688                 if(mysqlExecute.mysqlUpdate("
    UPDATE UserList SET password = \"" + newPassword
     + "\" , salt = \"" + newSaltString + "\",
    passwordHint =  \"" + passwordHint + "\" WHERE
    userId = " + userId + ";") == -1){
689                     throw new SQLException();
690                 }
691                 System.out.println("Password
    successfully changed.");
692                 return;
693             } else {
694                 System.out.println("New Password
    does not match; Please Try Again.");
695             }
696         }
697     }
698     public synchronized void printTasks(int userId
    , String username) throws SQLException,
    InterruptedException {
699         if (userId == 1) {
700             System.out.println("You cannot modify/
    change anything about Administrator's account on
    the client version.");
701             return;
702         }
703 
704         ArrayList<Integer> interestScheduleIds =
    new ArrayList<>();
705         ResultSet scheduleList = mysqlExecute.
    mysqlQuery("SELECT taskId, manualScheduleName,
    manualScheduleDate, interestScheduleId FROM
    TaskList WHERE userId = " + userId);
706         if (scheduleList == null) {
707             throw new SQLException();
708         }
709         scheduleList.next();
```

```
710         while (scheduleList.getString(4) != null){
711             interestScheduleIds.add(scheduleList.
    getInt(4));
712             scheduleList.next();
713             if(scheduleList.isAfterLast()) {
714                 break;
715             }
716         }
717         scheduleList.previous();
718 
719 
720         System.out.println("Schedules added by "
     + username + ":");
721         while(scheduleList.next()){
722             System.out.println("\tTask ID " +
    scheduleList.getString(1) + ": " + scheduleList.
    getString(2) + " in " + scheduleList.getString(3
    ));
723         }
724 
725         System.out.println("Schedules added based
    on " + username + "'s interest:");
726         Thread printInterestSchedulesThread = new
    Thread(()-> { try { printInterestSchedules(
    interestScheduleIds); } catch (SQLException
    exception) { exception.printStackTrace(); } });
727         printInterestSchedulesThread.start();
728         printInterestSchedulesThread.join();
729     }
730 
731     private void printInterestSchedules(ArrayList<
    Integer> eventIds) throws SQLException {
732         for(int i = 0; i<eventIds.size(); i++) {
733             ResultSet interestSchedule =
    mysqlExecute.mysqlQuery("SELECT eventName,
    eventDate FROM InterestSchedule WHERE eventId = "
     + eventIds.get(i) + ";");
734             if (interestSchedule == null) {
735                 throw new SQLException();
736             }
737             interestSchedule.next();
```

```
738             System.out.println("\t" + (i+1) + ". "
     + interestSchedule.getString(1) + " in " +
    interestSchedule.getString(2));
739         }
740     }
741 }
```