```java
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import java.util.InputMismatchException;
import java.util.Scanner;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

/*************************************************************
* Dalton Nofs                                                *
* Login ID: nofs5491                                         *
* CS-102, Summer 2017                                        *
* Programming Assignment 5                                   *
* Assignment class:  main entry point for assignment 3       *
*************************************************************/
public class Prog5
{
    static Database courseData = new Database(); // Create a database to import to

    /*************************************************************
    * Method: main()                                            *
    * Purpose: main entry for the program                         *
    *                                                              *
    * Parameters: String[]:             string array of prog args     *
    * Returns: void:                    N/A                       *
    *************************************************************/
    public static void main (String[] args)
    {
        UserInterface ui = new UserInterface(); // init the ui
        ui.addMainWindow();         // config for main window config

        try
        {
            // attempt to load the database with file location
            //   given at runtime
            courseData.loadDatabase(args);
        }
        catch(ArrayIndexOutOfBoundsException exc)
        {
            UserInterface.sendWarning("The database is full, " + exc.getMessage() +
                    " course(s) were loaded!", "ERROR");
        }
        catch(IllegalArgumentException exc)
        {
            // there was an error show it to user
            UserInterface.sendWarning(exc.getMessage(),"ERROR");
        }
    }

    /*************************************************************
    * Method: userAddClass()                                    *
    * Purpose: add a user class to database                     *
    *                                                             *
    * Parameters: Database: targetbase                           *
    * Returns: void:                    N/A                       *
    *************************************************************/
    static void userAddCourse(Database targetDatabase)
    {
        String userInput = JOptionPane.showInputDialog(
                            "Enter your class in the following format:\n"  +
                           "201003/CE-320/4/Microcomputers I/B+/N\n"      +
                           "yyyytt/corsnum/credit/title/grade/excluded\n" +
                           "yyyy is year tt is term (01,02,03,04)     ",
                           "New Course");
        Course tempCourse = new Course(); // course to be added
        String dateString = ""; // Temp string for separating the year and semester
        if(userInput == "")
        {
            UserInterface.sendWarning("You didn't enter anything", "Really?");
            return;
```

```
78          }
79          Scanner pieces = new Scanner(userInput); // Scanner for spliting string
80
81
82          // setTermTaken, and setExcludeFlag will throw a parse error
83          //     if data sent is not in the correct format
84          try
85          {
86              pieces.useDelimiter("/");
87              dateString = pieces.next();
88              if(dateString.length() != 6)
89              {
90                  throw new ParseException("Year/Term is wrong length",0);
91              }
92              tempCourse.setYearTaken(dateString.substring(0, 4)); // Set year to string char's 0-4 (year)
93              tempCourse.setTermTaken(dateString.substring(4, 6)); // Set term taken to the 2 digit semester code
94              tempCourse.setCourseNumber(pieces.next());             // Set the course number
95              tempCourse.setCreditCount(pieces.nextInt());         // Set the number of credits the class is worth
96              tempCourse.setCourseTitle(pieces.next());             // Set the course title
97              tempCourse.setCourseGrade(pieces.next().toUpperCase()); // Set the course grade
98              tempCourse.setExcludeFlag(pieces.next());             // Set the exclude flag
99          }
100         catch(ParseException exc)
101         {
102             UserInterface.sendWarning(exc.getMessage() + " Your input is ignored!\n", "Error");
103             return;
104         }
105         catch(InputMismatchException exc)
106         {
107             UserInterface.sendWarning(exc.getMessage()+"your input is ignored\n", "Error");
108             return;
109         }
110         targetDatabase.addCourse(tempCourse);
111     }
112
113     /*************************************************************
114     * Method: userStore()                                        *
115     * Purpose: store database to file                           *
116     *                                                             *
117     * Parameters: Database:          targetbase                 *
118     * Returns: void:                    N/A                      *
119     *************************************************************/
120     static void userStore(Database targetDatabase)
121     {
122         String userInput = JOptionPane.showInputDialog("Enter a file name to save to: ");
123         try
124         {
125             targetDatabase.storeDatabase(userInput);
126         }
127         catch(IllegalArgumentException exc)
128         {
129             UserInterface.sendMessage(
130                     "\nAre you trying to break me? There is nothing to store!\n",
131                     "ERROR");
132         }
133     }
134
135     /*************************************************************
136     * Method: userReload()                                       *
137     * Purpose: reload a database from file                        *
138     *                                                             *
139     * Parameters: Database:          targetbase                  *
140     * Returns: void:                    N/A                      *
141     *************************************************************/
142     static void userReload(Database targetDatabase)
143     {
144         String userInput = JOptionPane.showInputDialog(
145                 "Enter a file to load, within the local directory: ");
146         // Cast to string array to pass to load database
147         String userInputArray[] = {userInput};
148         targetDatabase.removeAll(); // destroy old data
149         try
150         {
151             // attempt to load the database with file location
152             //   given at runtime
153             targetDatabase.loadDatabase(userInputArray);
154         }
155         catch(ArrayIndexOutOfBoundsException exc)
```

```
156         {
157             UserInterface.sendWarning("The database is full, " + exc.getMessage() +
158                     " course(s) were loaded!", "ERROR");
159         }
160         catch(IllegalArgumentException exc)
161         {
162             UserInterface.sendWarning(exc.getMessage(), "ERROR");
163         }
164         UserInterface.sendMessage("Database reloaded!", "INFO");
165     }
166 
167     /*************************************************************
168     * Method: printCourse()                                      *
169     * Purpose: compile a single course into a buffer             *
170     *                                                            *
171     * NOTE: this was making all my methods too long so...        *
172     *                                                            *
173     * Parameters: Database: LinkedList: Index:                   *
174     *                 targetDatabase, returnResults, index       *
175     * Returns: String:              the info in the index        *
176     *************************************************************/
177     private static String printCourse(LinkedList<Course> returnResults, int index)
178     {
179         String buffer =
180             returnResults.get(index).getCourseNumber()     + ": " +
181             returnResults.get(index).getCourseTitle()      + " (" +
182             returnResults.get(index).getCreditCount()      + "). "+
183             returnResults.get(index).getTermTaken()        + " "  +
184             returnResults.get(index).getYearTaken()         + " "  +
185             returnResults.get(index).getCourseGrade()      + "\n" ;
186         return buffer;
187     }
188 
189     /*************************************************************
190     * Method: searchCourseNumInt()                               *
191     * Purpose: search for num interface for UI                   *
192     * Parameters: String             Course num                  *
193     * Returns: void:                    N/A                      *
194     *************************************************************/
195     public static void searchCourseNumInt(String userInput)
196     {
197         try
198         {
199             // this prints internally so just junk the return we dont need it
200             new CourseSearch().findByNumber(userInput, courseData);
201         }
202         catch (NoSuchFieldException e)
203         {
204             UserInterface.sendWarning("Course does not exist!", "ERROR");
205         }
206     }
207 
208     /*************************************************************
209     * Method: searchCourseTitleInt()                             *
210     * Purpose: search interface for title for UI                 *
211     * Parameters: String             Course title                *
212     * Returns: void:                    N/A                      *
213     *************************************************************/
214     public static void searchCourseTitleInt(String userInput)
215     {
216         try
217         {
218             // this prints internally so just junk the return we dont need it
219             new CourseSearch().findByTitle(userInput, courseData);
220         }
221         catch (NoSuchFieldException e)
222         {
223             UserInterface.sendWarning("Course does not exist!", "ERROR");
224         }
225     }
226 
227     /*************************************************************
228     * Method: printDatabaseInt()                                 *
229     * Purpose: printDatabase interface for UI                    *
230     * Parameters: void:                 N/A                      *
231     * Returns: void:                    N/A                      *
232     *************************************************************/
233     public static void printDatabaseInt()
```

```
234     {
235         try
236         {
237             // this does the actual printing
238             new ConsolePrint().printDatabase(courseData);
239         }
240         catch(IllegalArgumentException exc)
241         {
242             // some exception happened show it to user
243             UserInterface.sendWarning(exc.getMessage(),"ERROR");
244         }
245     }
246 
247     /*************************************************************
248     * Method: calculateGPAInt()                                  *
249     * Purpose: gpa calc interface for UI                         *
250     * Parameters: void:                 N/A                        *
251     * Returns: void:                    N/A                        *
252     *************************************************************/
253     public static void calculateGPAInt()
254     {
255         try
256         {
257             String GPA = String.format("%.2f\n\n",new GpaCalc().calcGpa(courseData));
258             UserInterface.sendMessage("GPA is: " + GPA, "GPA Calculation");
259         }
260         catch(IllegalArgumentException exc)
261         {
262             // some exception happened show it to user
263             UserInterface.sendWarning(exc.getMessage(),"ERROR");
264         }
265     }
266 
267     /*************************************************************
268     * Method: userEditCourse()                                  *
269     * Purpose: edit a user class to database , frontend public   *
270     *                                                            *
271     * Parameters: Database: Scanner: targetbase, console,        *
272     * Returns: void:                    N/A                      *
273     *************************************************************/
274     static void userEditCourse(Database targetDatabase)
275     {
276         createEditSearchWindow();
277     }
278 
279     /*************************************************************
280     * Method: userEditCourseWork()                              *
281     * Purpose: edit the course, the background work              *
282     *                                                            *
283     * Parameters: Search: JFrame:    search, searchWindow        *
284     * Returns: void:                    N/A                      *
285     *************************************************************/
286     private static void userEditCourseWork(String search)
287     {
288         String[] split = search.split("/"); // split the entry into 2
289         String userInput; // get first part
290         String termInput; // get second part
291         try
292         {
293             userInput = split[0]; // get first part
294             termInput = split[1]; // get second part
295         }
296         catch( ArrayIndexOutOfBoundsException exc)
297         {
298             UserInterface.sendWarning("You didn't enter enough information", "ERROR");
299             UserInterface.getMainWindow().setVisible(true);
300             return;
301         }
302         int promptCtr = 0; // Counter for seeing how many time the user was prompted
303         LinkedList<Course> returnResults = null; // Results from the course search
304         int editedCounter = 0; // For the number of edited courses
305         Database targetDatabase = Prog5.courseData; // get the database
306         try
307         {
308             returnResults = new CourseSearch().findByNumber(userInput, targetDatabase);
309         }
310         catch (NoSuchFieldException exc)
311         {
```

```
312             UserInterface.sendMessage("No results were found!", "INFO");
313             UserInterface.getMainWindow().setVisible(true);
314             return;
315         }
316         for(int index=0;index<returnResults.size();)
317         {
318             // Print the course as long as it matches year and term
319             if(termInput.equalsIgnoreCase(returnResults.get(index).getTermTakenRaw()))
320             {
321                 if(JOptionPane.showConfirmDialog(null, "Would you like to Edit: " +
322                         printCourse(returnResults, index)) == 0)
323                 {
324                     for(int index2=0;index2<targetDatabase.getDatabaseSize();index2++)
325                     {
326                         if(targetDatabase.get(index2).getTerm().equals(
327                                 returnResults.get(index).getTermTakenRaw()))
328                         {
329                             // remove course from lower list
330                             targetDatabase.remove(index2, returnResults.get(index));
331                             userAddCourse(targetDatabase);
332                             editedCounter++;
333                         }
334                     }
335                 }
336                 else{/* do nothing */}
337                 promptCtr++;
338             }
339             index += 2; // because of storage in results add 2 instead of 1
340         }
341         if(promptCtr>0)
342             UserInterface.sendMessage("You edited " + editedCounter +
343                     " course(s)!\n","INFO");
344         else
345             UserInterface.sendWarning(
346                     "There were no courses with the specified term!\n", "ERROR");
347         UserInterface.getMainWindow().setVisible(true);
348     }
349
350     /************************************************************
351     * Method: createEditSearchWindow()                          *
352     * Purpose: window for entry of course info                  *
353     * Parameters: void:                N/A                       *
354     * Returns: void:                   N/A                       *
355     ************************************************************/
356     private static void createEditSearchWindow()
357     {
358         // New window for search info
359         JFrame searchWindow = new JFrame("Edit SearchWindow");
360
361         searchWindow.setSize(400,100); // set the size
362         GridLayout aGrid = new GridLayout(3,2,0,0); // config the layout
363         searchWindow.setLayout(aGrid); // set window layout
364
365         // create buttons
366         JButton crsSearch     = new JButton("Search");
367         JButton close = new JButton("Close");
368
369         // create lables
370         JLabel crsNumLbl = new JLabel("Course Number:");
371         JLabel crsSemesterLbl = new JLabel("Course Semester (yyyytt):");
372
373         // create
374         JTextField crsNum = new JTextField();
375         JTextField crsSemester = new JTextField();
376
377         // add objects to the window's
378         searchWindow.add(crsNumLbl);
379         searchWindow.add(crsNum);
380         searchWindow.add(crsSemesterLbl);
381         searchWindow.add(crsSemester);
382         searchWindow.add(crsSearch);
383         searchWindow.add(close);
384
385         crsSearch.addActionListener(new ActionListener()
386         {
387               public void actionPerformed(ActionEvent e)
388               {
389                   userEditCourseWork(crsNum.getText() + "/" + crsSemester.getText());
```

```
390                 searchWindow.dispose();
391             }
392         });
393 
394         close.addActionListener(new ActionListener()
395         {
396               public void actionPerformed(ActionEvent e)
397               {
398                   searchWindow.dispose();
399                   UserInterface.getMainWindow().setVisible(true);
400               }
401         });
402 
403         searchWindow.setVisible(true);
404     }
405 
406     /************************************************************
407     * Method: userRemoveCourse()                                *
408     * Purpose: remove a user class to database , frontend public *
409     *                                                            *
410     * Parameters: Database:          targetbase,                 *
411     * Returns: void:                   N/A                       *
412     ************************************************************/
413     static void userRemoveCourse(Database targetDatabase)
414     {
415         createRemoveSearchWindow();
416     }
417 
418     /************************************************************
419     * Method: userRemoveCourseWork()                            *
420     * Purpose: background work for the remove course            *
421     * Parameters: void:                N/A                      *
422     * Returns: void:                   N/A                      *
423     ************************************************************/
424     private static void userRemoveCourseWork(String search)
425     {
426         String[] split = search.split("/"); // split the entry into 2
427         String userInput; // get first part
428         String termInput; // get second part
429         try
430         {
431             userInput = split[0]; // get first part
432             termInput = split[1]; // get second part
433         }
434         catch( ArrayIndexOutOfBoundsException exc)
435         {
436             UserInterface.sendWarning("You didn't enter enough information", "ERROR");
437             UserInterface.getMainWindow().setVisible(true);
438             return;
439         }
440         int promptCtr = 0; // Counter for seeing how many time the user was prompted
441         LinkedList<Course> returnResults = null; // Results from the course search
442         int deletedCounter = 0; // For the number of edited courses
443         Database targetDatabase = Prog5.courseData; // get the database
444         try
445         {
446             returnResults = new CourseSearch().findByNumber(userInput, targetDatabase);
447         }
448         catch (NoSuchFieldException exc)
449         {
450             UserInterface.sendWarning("No results were found!", "ERROR");
451             UserInterface.getMainWindow().setVisible(true);
452             return;
453         }
454         for(int index=0;index<returnResults.size();)
455         {
456             // Print the course as long as it matches year and term
457             if(termInput.equalsIgnoreCase(returnResults.get(index).getTermTakenRaw()))
458             {
459                 if(JOptionPane.showConfirmDialog(null,
460                         "Would you like to delete: " +
461                         printCourse(returnResults, index)) == 0)
462                 {
463                     for(int index2=0;index2<targetDatabase.getDatabaseSize();index2++)
464                     {
465                         if(targetDatabase.get(index2).getTerm().equals(
466                                 returnResults.get(index).getTermTakenRaw()))
467                         {
```

```
468                             // remove course from lower list
469                             targetDatabase.remove(index2, returnResults.get(index));
470                             deletedCounter++;
471                             break; // exit for loop as term search is done
472                         }
473                     }
474                 }
475                 else{/* do nothing */}
476                 promptCtr++;
477             }
478             index += 2; // because of storage in results add 2 instead of 1
479         }
480         if(promptCtr>0)
481             UserInterface.sendMessage("You deleted " + deletedCounter +
482                     " course(s)!\n", "INFO");
483         else
484             UserInterface.sendMessage(
485                     "There were no courses with the specified term!\n", "INFO");
486         UserInterface.getMainWindow().setVisible(true);
487     }
488 
489     /*************************************************************
490     * Method: createRemoveSearchWindow()                         *
491     * Purpose: window for entry of course info                     *
492     * Parameters: void:                 N/A                         *
493     * Returns: void:                    N/A                         *
494     *************************************************************/
495     private static void createRemoveSearchWindow()
496     {
497         // New window for search info
498         JFrame searchWindow = new JFrame("Remove SearchWindow");
499 
500         searchWindow.setSize(400,100); // set the size
501         GridLayout aGrid = new GridLayout(3,2,0,0); // config the layout
502         searchWindow.setLayout(aGrid); // set window layout
503 
504         // create buttons
505         JButton crsSearch     = new JButton("Search");
506         JButton close = new JButton("Close");
507 
508         // create lables
509         JLabel crsNumLbl = new JLabel("Course Number:");
510         JLabel crsSemesterLbl = new JLabel("Course Semester (yyyytt):");
511 
512         // create
513         JTextField crsNum = new JTextField();
514         JTextField crsSemester = new JTextField();
515 
516         // add objects to the window's
517         searchWindow.add(crsNumLbl);
518         searchWindow.add(crsNum);
519         searchWindow.add(crsSemesterLbl);
520         searchWindow.add(crsSemester);
521         searchWindow.add(crsSearch);
522         searchWindow.add(close);
523 
524         crsSearch.addActionListener(new ActionListener()
525         {
526               public void actionPerformed(ActionEvent e)
527               {
528                   userRemoveCourseWork(crsNum.getText() + "/" + crsSemester.getText());
529                   searchWindow.dispose();
530               }
531         });
532 
533         close.addActionListener(new ActionListener()
534         {
535               public void actionPerformed(ActionEvent e)
536               {
537                   searchWindow.dispose();
538                   UserInterface.getMainWindow().setVisible(true);
539               }
540         });
541 
542         searchWindow.setVisible(true);
543     }
544 }
```