

```

1 import java.io.BufferedWriter;
2 import java.io.FileOutputStream;
3 import java.io.IOException;
4 import java.io.OutputStreamWriter;
5 import java.io.Writer;
6 import java.text.ParseException;
7 import java.util.InputMismatchException;
8 import java.util.Scanner;
9
10 /*****
11  * Dalton Nofs
12  * Login ID: nofs5491
13  * CS-102, Summer 2017
14  * Programming Assignment 4
15  * Assignment class: main entry point for assignment 3
16  *****/
17 public class Prog4
18 {
19     /*****
20     * Method: main()
21     * Purpose: main entry for the program
22     *
23     * Parameters: String[]:      string array of prog args
24     * Returns: void:             N/A
25     *****/
26     public static void main (String[] args)
27     {
28         ConsolePrint localPrinter = new ConsolePrint(); // local printer for general printing
29         CourseSearch courseSearcher = new CourseSearch(); // Database searcher for course's
30         GpaCalc gpaCalculator = new GpaCalc(); // Calculator for gpa via database
31         Scanner console = new Scanner(System.in); // Scanner for parsing the console
32         int userInput = 0; // User input for selecting option
33         Database courseData = new Database(); // Create a database to import to
34         // String for printing the greeting message
35         String greetingString = "Welcome to the CS-102 Transcript Program" +
36                                 "\nCurrent available commands:" +
37                                 "\n  1 --> Search for a course number" +
38                                 "\n  2 --> Search course titles" +
39                                 "\n  3 --> Print all records" +
40                                 "\n  4 --> Compute GPA" +
41                                 "\n  5 --> Add Course" +
42                                 "\n  6 --> Remove Course" +
43                                 "\n  7 --> Edit Course" +
44                                 "\n  8 --> Store Database" +
45                                 "\n  9 --> Reload Database" +
46                                 "\n  0 --> Exit" ;
47
48         final int srchCrNum = 1; // Course by number
49         final int srchTitle = 2; // Course by title
50         final int prntData = 3; // Print all records
51         final int cmptGpa = 4; // Computer GPAs
52         final int addCrns = 5; // Add a course
53         final int rmCrns = 6; // Remove a course
54         final int editCrns = 7; // Edit a course
55         final int strData = 8; // Store Database
56         final int reLdData = 9; // Reload Database
57         final int exit = 0; // Exit
58         try
59         {
60             // attempt to load the database with file location
61             // given at runtime
62             courseData.loadDatabase(args);
63         }
64         catch (ArrayIndexOutOfBoundsException exc)
65         {
66             System.out.println("The database is full, " + exc.getMessage() + " course(s) were loaded!");
67         }
68         catch (IllegalArgumentException exc)
69         {
70             System.out.println(exc.getMessage());
71         }
72
73         System.out.println("Database has been loaded!\n");
74
75         while(true)
76         {
77             System.out.println(greetingString); // Print welcome message

```

```
78     try
79     {
80         userInput = console.nextInt();
81         console.nextLine(); // Clear the scanner's buffer by going to the return char
82                             // still viable if a newline char is pasted into terminal
83     }
84     catch(InputMismatchException exc)
85     {
86         // Just change the user input to use the default catch
87         userInput = 0;
88     }
89
90     switch (userInput)
91     {
92         case srchCrSNum: System.out.println("What is the Number of the course?: ");
93             try
94             {
95                 courseSearcher.findByNumber(console.next(), courseData);
96             }
97             catch(NoSuchFieldException exc)
98             {
99                 System.out.println("Course was not found!\n");
100             }
101             break;
102
103         case srchTitle: System.out.println("What is the title of the course?: ");
104             try
105             {
106                 courseSearcher.findByTitle(console.next(), courseData);
107             }
108             catch(NoSuchFieldException exc)
109             {
110                 System.out.println("Course was not found!\n");
111             }
112             break;
113
114         case prntData: System.out.println("Printing all records\n");
115             try
116             {
117                 localPrinter.printDatabase(courseData);
118             }
119             catch(IllegalArgumentException exc)
120             {
121                 System.out.println(exc.getMessage());
122             }
123             break;
124
125         case cmptGpa: System.out.print("Students GPA: ");
126             try
127             {
128                 System.out.format("%.2f\n\n", gpaCalculator.calcGpa(courseData));
129             }
130             catch(IllegalArgumentException exc)
131             {
132                 System.out.println(exc.getMessage());
133             }
134             break;
135
136         case addCrS: userAddCourse(courseData, console);
137             break;
138
139         case rmCrS: userRemoveCourse(courseData, console, courseSearcher);
140             break;
141
142         case editCrS: userEditCourse(courseData, console, courseSearcher);
143             break;
144
145         case strData: System.out.println("Storing database!");
146             userStore(courseData, console);
147             break;
148
149         case reLdData: userReload(courseData, console);
150             break;
151
152         case exit: System.out.println("Exiting");
153             System.exit(0); // exit successfully
154
155         // Catch all (commands that are not 1-4,9)
```

```

156         default: System.out.println("The command you entered is not recognized.\n");
157         break;
158     }
159 }
160 }
161
162 /*****
163  * Method: userAddClass()
164  * Purpose: add a user class to database
165  *
166  * Parameters: Database: Scanner: targetbase, console scanner
167  * Returns: void: N/A
168  *****/
169 private static void userAddCourse(Database targetDatabase, Scanner console)
170 {
171     System.out.println("Enter your class in the following format:\n" +
172         "201003/CE-320/4/Microcomputers I/B+/N\n" +
173         "yyyytt/corsnum/credit/title/grade/excluded\n" +
174         "yyyy is year tt is term (01,02,03,04) " );
175     String userInput = ""; // User input string to be feed to addCourse
176     Course tempCourse = new Course(); // course to be added
177     String dateString = ""; // Temp string for separating the year and semester
178     userInput = console.nextLine(); // get users input
179     if(userInput == "")
180     {
181         System.out.println("You didn't enter anything");
182         return;
183     }
184     Scanner pieces = new Scanner(userInput); // Scanner for splitting string
185
186     // setTermTaken, and setExcludeFlag will throw a parse error
187     // if data sent is not in the correct format
188     try
189     {
190         pieces.useDelimiter("/");
191         dateString = pieces.next();
192         if(dateString.length() != 6)
193         {
194             throw new ParseException("Year/Term is wrong length",0);
195         }
196         tempCourse.setYearTaken(dateString.substring(0, 4)); // Set year to string char's 0-4 (year)
197         tempCourse.setTermTaken(dateString.substring(4, 6)); // Set term taken to the 2 digit semester code
198         tempCourse.setCourseNumber(pieces.next()); // Set the course number
199         tempCourse.setCreditCount(pieces.nextInt()); // Set the number of credits the class is worth
200         tempCourse.setCourseTitle(pieces.next()); // Set the course title
201         tempCourse.setCourseGrade(pieces.next().toUpperCase()); // Set the course grade
202         tempCourse.setExcludeFlag(pieces.next()); // Set the exclude flag
203     }
204     catch(ParseException exc)
205     {
206         System.out.println(exc.getMessage() + " Your input is ignored!\n");
207         return;
208     }
209     catch(InputMismatchException exc)
210     {
211         System.out.println(exc.getMessage()+"your input is ignored\n");
212         return;
213     }
214     targetDatabase.addCourse(tempCourse);
215     System.out.println("\n"); // extra space for prettiness
216 }
217
218 /*****
219  * Method: userRemoveCourse()
220  * Purpose: remove a user class to database
221  *
222  * Parameters: Database: Scanner: targetbase, console scanner
223  * Returns: void: N/A
224  *****/
225
226 private static void userRemoveCourse(Database targetDatabase, Scanner console,
227     CourseSearch courseSearcher)
228 {
229     int promptCtr = 0; // Counter for seeing how many time the user was prompted
230     System.out.println("Please enter the course number you would like to remove:");
231     String userInput = console.next(); // users course input
232     console.nextLine(); // Clear the scanner's buffer by going to the return char
233     // still viable if a newline char is pasted into terminal

```

```

234     LinkedList<Course> returnResults; // Results from the course search
235     int deletedCounter = 0; // counter for number of courses deleted
236     try
237     {
238         System.out.print("Course search, ");
239         returnResults = courseSearcher.findByNumber(userInput, targetDatabase);
240     }
241     catch (NoSuchFieldException exc)
242     {
243         System.out.println("No results were found!\n");
244         return;
245     }
246     System.out.println("Please enter the term you would like to remove it from " +
247         "(yyyytt): ");
248     String termInput = console.next(); // capture the term to delete from
249     console.nextLine(); // Clear the scanner's buffer by going to the return char
250         // still viable if a newline char is pasted into terminal
251     for(int index=0;index<returnResults.size();)
252     {
253         // Print the course as long as it matches year and term
254         if(termInput.equalsIgnoreCase(returnResults.get(index).getTermTakenRaw()))
255         {
256             printCourse(returnResults, index);
257             System.out.println("Would you like to delete(y/n):");
258             userInput = console.next();
259             if(userInput.equalsIgnoreCase("y"))
260             {
261                 for(int index2=0;index2<targetDatabase.getDatabaseSize();index2++)
262                 {
263                     if(targetDatabase.get(index2).getTerm().equals(
264                         returnResults.get(index).getTermTakenRaw()))
265                     {
266                         // remove course from lower list
267                         targetDatabase.remove(index2, returnResults.get(index));
268                         deletedCounter++;
269                         break; // exit for loop as term search is done
270                     }
271                 }
272             }
273             else{/* do nothing */}
274             promptCtr++;
275         }
276         index += 2; // because of storage in results add 2 instead of 1
277     }
278     if(promptCtr>0)
279         System.out.println("You deleted " + deletedCounter + " course(s)!\n");
280     else
281         System.out.println("There were no courses with the specified term!\n");
282 }
283
284 /*****
285 * Method: userRemoveCourse()
286 * Purpose: remove a user class to database
287 *
288 * Parameters: Database: Scanner: targetbase, console,
289 * Returns: void: N/A
290 *****/
291 private static void userEditCourse(Database targetDatabase, Scanner console, CourseSearch courseSearcher)
292 {
293     int promptCtr = 0; // Counter for seeing how many time the user was prompted
294     System.out.println("Please enter the course number you would like to edit:");
295     String userInput = console.next(); // users course input
296     console.nextLine(); // Clear the scanner's buffer by going to the return char
297         // still viable if a newline char is pasted into terminal
298     LinkedList<Course> returnResults = null; // Results from the course search
299     int editedCounter = 0; // For the number of edited courses
300     try
301     {
302         System.out.print("Course search, ");
303         returnResults = courseSearcher.findByNumber(userInput, targetDatabase);
304     }
305     catch (NoSuchFieldException exc)
306     {
307         System.out.println("No results were found!\n");
308         return;
309     }
310     System.out.println("Please enter the term you would like to edit it in " +
311         "(yyyytt): ");

```

```

312     String termInput = console.next(); // capture the term to delete from
313     console.nextLine(); // Clear the scanner's buffer by going to the return char
314                             // still viable if a newline char is pasted into terminal
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             printCourse(returnResults, index);
322             System.out.println("Would you like to Edit?(y/n):");
323             userInput = console.next();
324             if(userInput.equalsIgnoreCase("y"))
325             {
326                 for(int index2=0;index2<targetDatabase.getDatabaseSize();index2++)
327                 {
328                     if(targetDatabase.get(index2).getTerm().equals(
329                         returnResults.get(index).getTermTakenRaw()))
330                     {
331                         // remove course from lower list
332                         targetDatabase.remove(index2, returnResults.get(index));
333                         // create a new scanner as the old one causes problems
334                         userAddCourse(targetDatabase, new Scanner(System.in));
335                         editedCounter++;
336                     }
337                 }
338             }
339             else{/* do nothing */}
340             promptCtr++;
341         }
342         index += 2; // because of storage in results add 2 instead of 1
343     }
344     if(promptCtr>0)
345         System.out.println("You edited " + editedCounter + " course(s)!\n");
346     else
347         System.out.println("There were no courses with the specified term!\n");
348 }
349
350 /*****
351  * Method: userStore()
352  * Purpose: store database to file
353  *
354  * Parameters: Database: Scanner: targetbase, console scanner *
355  * Returns: void: N/A
356  *****/
357 private static void userStore(Database targetDatabase, Scanner console)
358 {
359     System.out.println("Enter a file name to save to: ");
360     String userInput = console.next(); // users course input
361     console.nextLine(); // Clear the scanner's buffer by going to the return char
362                             // still viable if a newline char is pasted into terminal
363     try
364     {
365         targetDatabase.storeDatabase(userInput);
366     }
367     catch(IllegalArgumentException exc)
368     {
369         System.out.println("\nAre you trying to break me? There is nothing to store!\n");
370     }
371 }
372
373 /*****
374  * Method: userReload()
375  * Purpose: reload a database from file
376  *
377  * Parameters: Database: Scanner: targetbase, console scanner *
378  * Returns: void: N/A
379  *****/
380 private static void userReload(Database targetDatabase, Scanner console)
381 {
382     System.out.println("Enter a file to load, within the local directory: ");
383     String userInput = console.next(); // users course input
384     console.nextLine(); // Clear the scanner's buffer by going to the return char
385                             // still viable if a newline char is pasted into terminal
386
387     // Cast to string array to pass to load database
388     String userInputArray[] = {userInput};
389     targetDatabase.removeAll(); // destroy old data

```

```

390     try
391     {
392         // attempt to load the database with file location
393         // given at runtime
394         targetDatabase.loadDatabase(userInputArray);
395     }
396     catch (ArrayIndexOutOfBoundsException exc)
397     {
398         System.out.println("The database is full, " + exc.getMessage() + " course(s) were loaded!");
399     }
400     catch (IllegalArgumentException exc)
401     {
402         System.out.println(exc.getMessage());
403     }
404     System.out.println("\nDatabase reloaded!\n");
405 }
406
407 /*****
408  * Method: printCourse()
409  * Purpose: print a singular course
410  *
411  * NOTE: this was making all my methods too long so...
412  *
413  * Parameters: Database: LinkedList<Course> returnResults, int index
414  * Returns: void: N/A
415  *****/
416 private static void printCourse(LinkedList<Course> returnResults, int index)
417 {
418     System.out.print (
419         returnResults.get(index).getCourseNumber() + ": " +
420         returnResults.get(index).getCourseTitle() + " (" +
421         returnResults.get(index).getCreditCount() + "). " +
422         returnResults.get(index).getTermTaken() + " " +
423         returnResults.get(index).getYearTaken() + " " +
424         returnResults.get(index).getCourseGrade() + "\n" );
425 }
426
427 }

```