Benjamin Seidenberg

Candidate Number: 001279-023

IB Computer Science Dossier

Table Of Contents:

Introduction:

All high school students are familiar with taking tests. They are a necessary evil of the educational system. Teachers need a reliable way to create, give and grade tests. While the traditional system of giving a paper test and then reading the responses to give a score is workable, I sought to develop a computerized testing system to automate much of the monotony of this task.

Stage A: Analysis

Analysis of the Problem:

When creating a test, there are many choices that have to be made. The nature of an automated computer system will force the answer to many of these choices by its design. This raises the following questions:

- What type of questions would the program support? The basic types of questions are:
    - Essay The student gives a long answer (usually requiring multiple paragraphs)
    - Short Answer The student answers the question in a few sentences
      Multiple Choice A question and a few answer choices are given, and the student picks the correct one
    - Matching Two lists are given, and the student must match the corresponding choices in each list
    - True/False A statement is given, and the student must decide if it is true or false

- What mechanisms will there be to prevent cheating?
- How easy is the program to use for students?
- How easy is the program to use for teachers?
- How will exams be scored?
- Will the student be given feedback on their progress?

After some thought, I arrived at these goals:

- The program would support multiple choice questions. Essay and short answer styled questions are simply too hard to automatically grade. True/False questions can easily be supported as multiple choice questions. This decision lead to several new questions:
    - How many possible answer choices are allowed? While it is possible to allow an arbitrary number of answer choices, this behavior is more difficult to program. Thus, I settled on 4 answer choices, which matches what many of my previous teachers have used on their tests.
    - Should the answer choices be shown randomly? This question has arguments for both sides. While randomizing the order of the answers makes it more difficult to cheat and provides a personalized test for each student, it can throw off questions that have answers such as “All of the above” or “Both C and D”. I decided that the benefits of randomization outweighed the drawbacks, and that the documentation could explain the necessity of making answers ambiguous with regard to order.

  - Would the testing program show the answers after the student guessed? If not, would it at least show students their grade? I decided to make that an option for each test separately.

- I decided that to prevent cheating, all tests would require passwords to access, and the program would prevent a student from taking a test twice. In addition, every student and teacher would be required to login with a unique user ID and password. Also, if the program is aborted in the middle of a test, the grade of 0 is assigned, and the student will not be able to retake the test. If possible, the data files will be protected by the operating system in order to prevent the user from reading them, but this feature is not available in all operating systems.
- To make the program as easy to use for the user as possible, I decided to make the process as straight forward as possible. I decided to give the student as few choices as possible. They would log in, pick a test, enter the test password and take the test. Thus, their chance of confusion would be low. I decided to make it extremely simple, the test ID would be given to them by their teacher, to make it difficult to take the wrong test by mistake.
- Because the teacher needs to do more than the student, the teacher interface will be more complex, an unfortunate necessity. However, in order to maintain ease of use, the menu system will be task-based, and lead the user through their choices.

  I decided that exams will be scored on a simple 100 pt scale (number right / total * 100). The score would be stored as an integer in the program.

  I decided that the teacher would be able to set an option on whether students would be able to see the correct answer after they completed a problem, only their

grade at the end, or no information at all. This way, tests that are not for major grades can give students instant feedback, while other tests can be secured.

Criteria for Success:

Goals

As determined by the analysis, the program needs to be able to:

- Allow students to take tests
- Track Student Grades
- Allow teachers to:
    - Create and manage tests
    - Manage students
    - View Student Grades

Requirements:

The software must be:

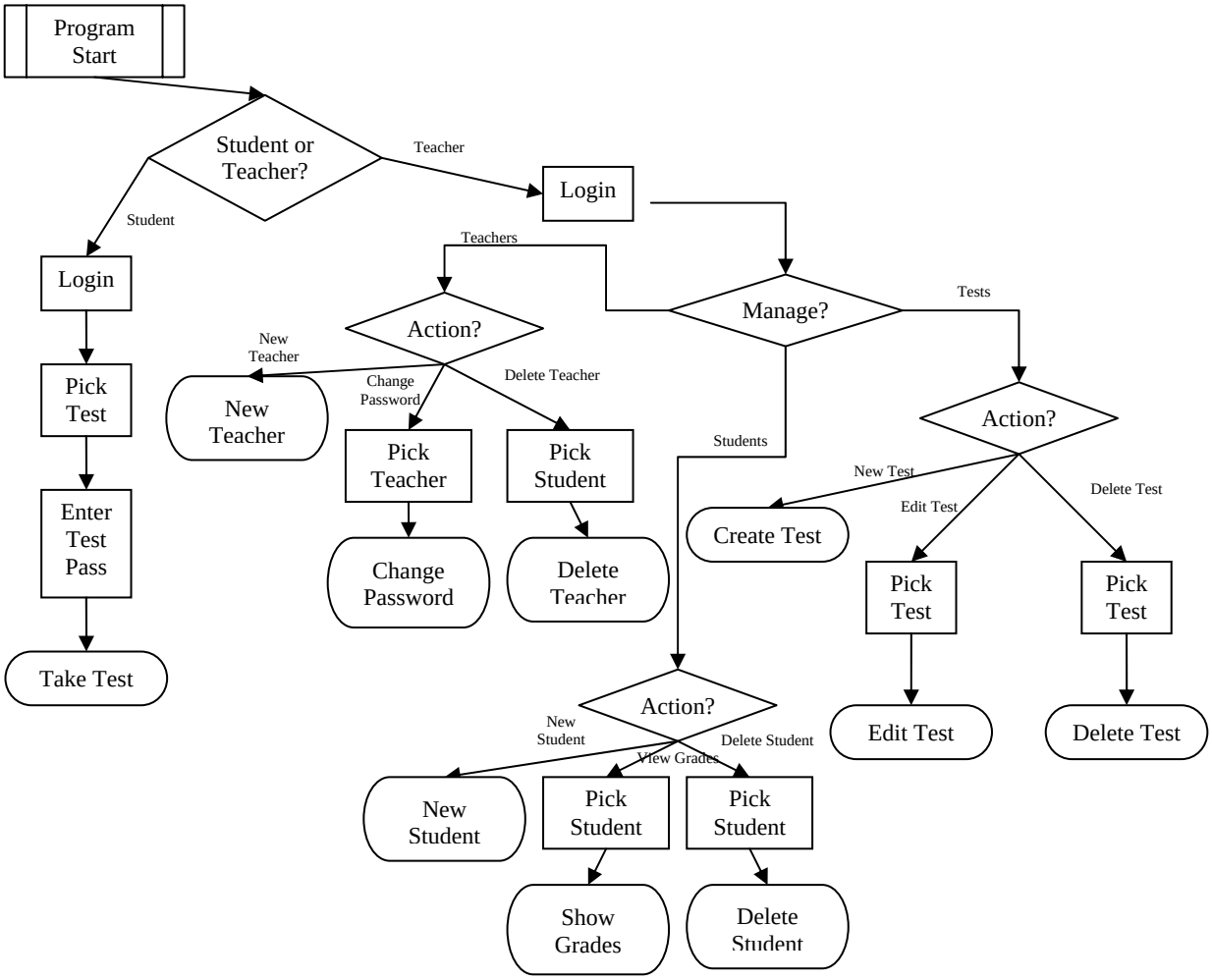
- Easy to use, especially for students who will not read most documentation
- Able to handle errors and arbitrary input
- Reasonably secure from cheating
    - This will be based on an assumption that students can not read the programs data files.
- Able to run on hardware available for schools without needing to purchase specialized hardware/software.
    - By using java, the program binary is cross-platform compatible and can run on these systems:

        - Macintosh
        - PC/Windows
        - PC/Linux
        - PC/BSD Variant
        - Mainframe/Unix and dumb terminals
    - Java interpreters are available for free.

Design Prototype:

Program Outline:

Diagram 1: Program flow in main interface



Stage B: Detailed Design

Data Types:

External:

There are several things that need to be stored for this program between runs. These must be stored in external files. I chose to use random access files to store binary records for

- Teacher database
- Student database
- Tests Each test was stored in a separate file. It had a header, then a bunch of question records
- A master database of all tests, storing the ID number and the file name of the test file

The layout of each file is shown below.

Diagram 2: Student Data Record

4 bytes int idNum	**36 bytes** byte[] password (Casted back to string later, for 36 character password)								
4 bytes test #1 id	**4 bytes** test #1 score	**4 bytes** test #2 id	**4 bytes** test #2 score	**4 bytes** test #3 id	**4 bytes** test #3 score	**4 bytes** test #4 id	**4 bytes** test #4 score	**4 bytes** test #5 id	**4 bytes** test #5 score
4 bytes test #6 id	**4 bytes** test #6 score	**4 bytes** test #7 id	**4 bytes** test #7 score	**4 bytes** test #8 id	**4 bytes** test #8 score	**4 bytes** test #9 id	**4 bytes** test #9 score	**4 bytes** test #10 id	**4 bytes** test #10 score

Diagram 3: Teacher Data Record

4 bytes int idNum	**36 bytes** byte[] password (Casted back to char later, for 36 character password)

Diagram 4: Question Data Record

<table>
<tr><td colspan="2">200 bytes
byte[] qtext
(Casted back to String later, for 200 character question)</td><td>1 byte
char
correctAnswer</td></tr>
<tr><td>100 bytes
byte[] choiceA
(Casted back to String later,
for 100 character answer)</td><td colspan="2">100 bytes
byte[] choiceB
(Casted back to String later,
for 100 character answer)</td></tr>
<tr><td>100 bytes
byte[] choiceC
(Casted back to String later,
for 100 character answer)</td><td colspan="2">100 bytes
byte[] choiceD
(Casted back to String later,
for 100 character answer)</td></tr>
</table>

Diagram 5: Test File

<table>
<tr><td>4 Bytes
int
numQuestions</td><td>4 Bytes
int
testId</td><td>1 byte
boolean
showGrade</td><td>1 byte
boolean
showAnswers</td></tr>
<tr><td colspan="4">36 bytes
byte[] testPass
(Casted back to String later, for 36 character password)</td></tr>
<tr><td colspan="4">601 * numQuestion bytes
Question[] questions
<table><tr><td>601 bytes
question 1</td><td>601 bytes
question 2</td><td>601 bytes
question 3</td><td>…..</td></tr></table></td></tr>
</table>

Internal:

Once the program was running, the data had to be stored internally. This is done in various ways for the various data types:

- Student: The database is accessed as needed, so that only one student is kept at a time. A student class represents this student, storing the password as a String and id number as an int. The grades on the past 10 tests are stored as a 10x2 array of integers. An example student would look like this:
    - ID: 323
    - Password: "thisisapassword" (Max 30 chars)
    - Grades:
        - Test 21: 100
        - Test 351: 100
        - Test 202: 90
        - Test 89: 87
        - Test 174: 68
        - Test 213: 100
        - Test 869: 74
        - Test 283: 96
        - Test 781: 84
        - Test 672: 93

- Teacher: Like students, teachers will mainly be kept in the random access file database, and will be read into a teacher class as the need arises. The teacher will store their id number as an int and their password as a string (max 30 chars). An example teacher would look like this (the default teacher):
    - ID Number: 42
    - Password: "lifeandall" (Max 30 chars)

- Test Database: This is a random access file that finds tests on demand. It stores the id number of the test as an integer and the file name of the test file as a string (max 256 chars). It returns the file name of the test, to be used by the Test class constructor. An example entry looks like this
    - Test ID: 23
    - Test Filename: “testname.tst” (Max 256 chars)
- Test: A test is stored in a random access file with a header and then each question. When it is time to take a test, an instance of the Test class is created and loads the information from the file. The test class contains a doubly linked list containing all the questions, the number of questions (stored as an int), a boolean variable of whether students get to see their scores, a boolean variable of whether students get to see their answers, the test id (stored as an int), and the test password (stored as a String). Thus, an example test might look like this:
    - Test ID: 23
    - Test Password: “foobar”
    - Number of questions: 3
    - Show students their grades: yes
    - Show students the correct answers: yes
    - [List of questions] (For example question, see next section)
- Questions: Questions are stored in the Test file, but are read into an instance of the Question class when the test is loaded, then put into a linked list. A question has a string for the question itself (200 char max), and 4 strings for each answer choice (100 chars each). The question will also have the options for showing

students their grades and the correct answers, set from the parent test. A sample question will look like this:

- Question Text: “Which planet is closest to the sun?”
- Choice A: “Venus”
- Choice B: “Earth”
- Choice C: “Mercury”
- Choice D: “Jupiter”
- Correct Answer: 'c'
- Show students their grades: yes (option stored in the test header)
- Show students the correct answers: yes (option stored in the test header)

Algorithms:

The program utilizes two main types of structures that take an algorithmic approach: a doubly linked list and random access files.

Linked List:

- **Adding:** This is done by either prepending or appending. In both cases, there is a precondition that the list exists. A check is done for an empty list, then the node is added an links are updated like such:

```
// Appends to end of list
public void append(Object x) {
      if (head == null || tail == null) {
            head = new DLLNode(x);
            head.prev = null;
            head.next = null;
            tail = head;
            return;
      }
      tail.next = new DLLNode(x);
      tail.next.prev = tail;
      tail.next.next = null;
      tail = tail.next;
}
// Prepends to beginning of the list
public void prepend(Object x) {
      if (head == null || tail == null) {
            head = new DLLNode(x);
            head.prev = null;
            head.next = null;
            tail = head;
            return;
      }
      head.prev = new DLLNode(x);
      head.prev.prev = null;
      head.prev.next = head;
      head = head.prev;
}
```

- **Searching:** The whereAt method will return a pointer to the node containing some object by doing a linear search. It requires the object to be in the list as a precondition. It is private since it returns a pointer to a node. It will return the pointer or throw a NoSuchElementException.

```
// Finds a node in the list
private DLLNode whereAt(Object x) {
      DLLNode pos = head;
      while (pos!=null) {
            if (pos.data.equals(x))
                  return pos;
            pos = pos.next;
      }
      return null;
}
```

- **Deleting:** This is done by swapping links to remove all references to a node in the list. Once this is done, the memory will be freed by the garbage collector.

  Preconditions: The method is passed in a pointer to the node; the list and the node exist. Postcondition: the node has been removed from the list.

```
// Deletes a node, given a pointer
private void delete(DLLNode x) {
      if (x == null)
            throw new NoSuchElementException("Can't delete
                  nonexistant node");
      x.next.prev=x.prev;
      x.prev.next=x.next;
      x = null;
}
```

Random Access Files:

- **Adding:** This is done by seeking to the end of the file and then writing the data.

  Precondition: File can be written to. Postcondition: Entry is in

  file

```
// Adds a teacher to the database
// It is the responsibility of the calling function
// to ensure that the information is correct
public void addTeacher(int id, String pass) throws IOException {
      RandomAccessFile fb = new RandomAccessFile(fname, "rw");
      fb.seek(fb.length()); // Go to end of file
      fb.writeInt(id);
      StringBuffer sb = new StringBuffer(pass);
      sb.setLength(TEACHPASS_BYTES);
      fb.writeBytes(sb.toString());
      fb.close();
}
```

- **Deleting a file:** The record is overwritten with the last item in the file, and then the file is truncated. This will cause less I/O usage and make the operation

MUCH faster, being O(1) rather than O(n) like moving every record around would be.

```
public void delete(int teacher) throws NoSuchTeacherException,IOException {
      // Open file buffer
      RandomAccessFile fb = new RandomAccessFile(fname, "rw");
      // Look for teacher
      int numItems = (int)(fb.length()/TEACHER_SIZE); // How many items
      for (int i = 0; i < numItems; i++) { // iterate through
            fb.seek(i * TEACHER_SIZE); // go to beggining of the record
            if (fb.readInt() == teacher) { // the id is first
                  fb.seek((numItems - 1) * TEACHER_SIZE); // Go to last
                  int newId = fb.readInt(); // read id
                  byte[] ba = new byte[TEACHPASS_BYTES];
                  fb.readFully(ba); // read pass
                  fb.seek(i * TEACHER_SIZE); // go back to where we were
                  fb.writeInt(newId); //write id
                  fb.write(ba); // write pass
                  fb.setLength ((numItems -1) * TEACHER_SIZE); // truncate file
                  return; // done
            }
      }
      fb.close();
      throw new NoSuchTeacherException("Teacher number" + teacher + " not found!");
}
```

- **Searching:** A linear search is done in the same manner as the linked list. This is shown in the example above. The for() loop does a sequential search.

Modular Organization:

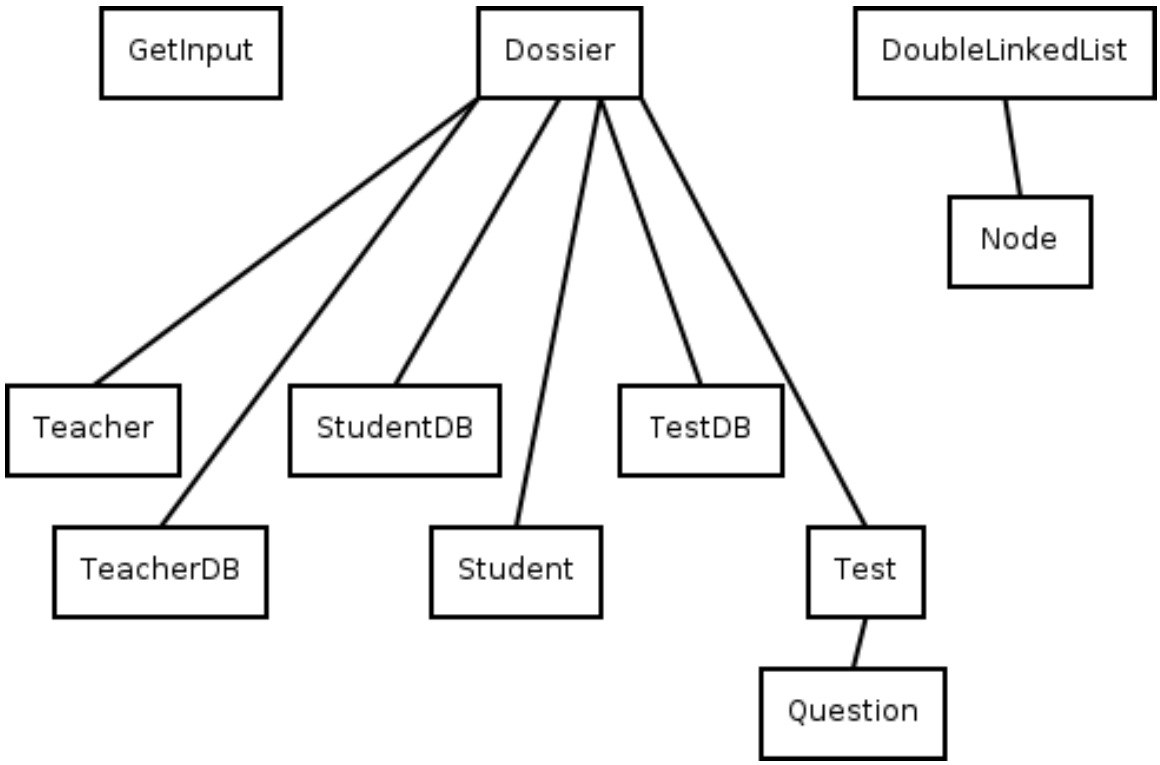
The program will be made up of several modules, each a separate Java class that represents either a real world object or offers a specific type of utility. The classes that will be used are as follows:

- Dossier: This class can be thought of as the “main class”. It is never instantiated, and contains all the code to interact with the user, handle logins, etc. Program entry is into this class, and it will branch out into other classes as needed.
- Teacher: This class represents a teacher in the real world. It is instantiated as needed from the TeacherDB class. It will store the teacher’s login ID and password, and has the necessary assessors and modifiers to manipulate this data.
- TeacherDB: This class represents a database of teachers. It uses a RandomAccessFile to store the information on disk, and has the methods required to read, write, search for, modify and delete teachers.
- Student: This class represents a student in the real world. It is instantiated as needed from the StudentDB class. It will store the student’s login ID and password and the grades in the last 10 tests the student took. It has the necessary assessors and modifiers to manipulate this data.
- StudentDB: This class represents a database of students. It uses a RandomAccessFile to store the information on disk, and has the methods required to read, write, search for, modify and delete teachers.
- Test: This class represents a test. It is instantiated with a filename from the TestDB class, and has the I/O methods necessary to read and write itself to disk. It will contain the various test options and a linked list of all the questions. It also

has the methods to edit the test, create a new test, and take the test, returning the score.

- TestDB: This class represents a database of tests, indexed by their id number and returning their filename. It uses a RandomAccessFile to store the information on disk, and has the methods required to read, write, search for, and delete tests.
- Question: This class represents a question on a test. It stores the question's information and then asks it, returning whether it was correct. It is given the options on whether or not students can see their grades and the correct answers by the constructor, which is called by the Test it is a part of. The Question class has the various assessors and modifiers needed to manipulate its data.
- DoubleLinkedList: This class is an implementation of a doubly linked list. It is used by the Test class to store questions, however all of its methods support a generic Object class so it can be used to store any type of data. It has the various methods to append, prepend, remove, find, get, and manipulate objects in the list.
- Node: This is a private subclass of DoubleLinkedList. It represents a node in the list, and stores one object and a pointer to the nodes ahead of and behind it.
- GetInput: This class is used to get input from the user. All of its methods are static and it is never instantiated. It provides basic error checking to ensure data of the requested type is returned.

Diagram 6: Modular organization



Stage C: The Program

Usability:

Design Goal	Documentation
The student is guided through the login and test taking process	See student login and test screenshots.
Teachers can add students.	See student management and add student screenshot.
Teachers can delete students.	See student management and delete student screenshots.
Teachers can view student grades.	See student management and add student screenshots.
Teachers can add teachers.	See teacher management and add teacher screenshots.
Teachers can delete teachers.	See teacher management and delete teacher screenshots.
Teachers can change their passwords.	See teacher management and change password screenshots.
Teachers can create new tests.	See test management and test creation screenshots.
Teachers can edit tests.	See test management and test editing screenshots.
Teachers can remove tests.	See test management and test deletion screenshots.
All prompts are explanatory, and explicitly tell the user what to do.	See all screenshots.
Errors are handled gracefully.	See next section

Bad input is handled gracefully.	See next Section.

Error Handling:

There are two types of errors likely to occur in this program, user errors and system errors. Each is handled differently.

User Errors:

This type of error mainly consists of putting in the wrong type of data or the wrong data itself. An example of the first would be entering a string instead of an integer; an example of the second would be entering the wrong ID number.

The first type of error is handled by the GetInput class. After reading the input, it will catch the Exceptions thrown by whatever class parses that type of data. If this happens, it will alert the user the input was bad, and recursively call itself again (the recursion is to minimize the amount of code needed). This can be seen in this example which gets an integer:

```
// returns an integer from the keyboard
public static int getInt() {
   x = 0;
   BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));
      try {
            x=Integer.parseInt(kb.readLine());
      }
      catch (IOException e) {
             System.out.println("Error " + e + " in the getInt() function");
      }
      catch (NumberFormatException e) {
            System.out.println("\nWarning! That was not an Integer!");
            System.out.print("Enter an Integer: ");
            x=getInt();
      }
      return x;
}
```

This type of error checking is found in GetInput.java lines 32-36, 52-55, and 69-73.

The second type of user error is entering an out-of-bounds value, for example in a menu. For example, if a menu has choices 1 through 4, the user might enter 5 by mistake (or -236). This is handled by looping until a proper value is entered. Consider this example from the main menu:

```
[...]
while(true) { // Will abort with System.exit()
   // Prompt for Student/Teacher login or quit
   System.out.println("1.) Login Student");
   System.out.println("2.) Login Teacher");
   System.out.println("3.) Quit");
   System.out.print("Enter your choice: ");
   int choice = GetInput.getInt();
   while (!(choice > 0 && choice <4)) {
      System.out.println("That was not a valid choice!");
      System.out.println("Enter your choice: ");
      choice = GetInput.getInt();
   }
   switch(choice) { // No need for 'default' since only possible values are 1-3
[...]
```

This type of error checking can be seen in Dossier.java lines 41-45, 111-115, 146-150, 157-161, 180-184, 203-207, 241-245, 252-256, 274-278, 302-306, 338-342, 349-352, 381-385, 403-407, 440-445, 463-467, 480-485, and 527-531, as well as in Test.java on lines 169-173, 207-213, 245-249, 272-275, 280-283, 293-296, 302-305, 344-348, 354-357, 364-367, and 377-381.

System Errors:

Like user errors, there are two kinds of system errors, very similar to each other. There are I/O errors and other such things that are completely outside the scope of our program. They are caused by the java libraries, and throw IOExceptions. The lower level modules (TestDB, StudentDB, TeacherDB) simply pass these up for the higher level code

(Dossier and Test) to handle. The program simply exits with an error status after these, since they are a fault of the underlying system. For example:

```
try {
   teach = teachers.findTeacher(id); // Find teacher
   [...OTHER CODE HERE...]
}
catch(IOException e) {
   System.out.println("An I/O Error Occured.");
   System.out.println("The exact error follows: ");
   System.out.println("\t" + e.getMessage());
   System.exit(1);
}
```

This is repeated throughout Test.java and Dossier.java

The other type of error is when an item that is requested is not found in a database. In other uses of these modules, this could be a normal occurance, but in this program, these functions only occurs after reading something out of the database, which means that if this exception is thrown, someone tampered with the databases. We then exit like this:

```
try{
   std.replaceScore(testid, score);
}
catch (TestNotFoundException e) { // Should never happen
   System.out.println("A serious error has occured.");
   System.out.println("The student database has been corrupted or tampered with");
   System.out.println("Contact your teacher for more help");
   System.exit(1);
}
```

This is repeated throughout Dossier.java

Evidence of Success:

Goal	Documentation
The student is guided through the login and test taking process	See student login and test screenshots.
Teachers can add students.	See student management and add student screenshot.
Teachers can delete students.	See student management and delete student screenshots.
Teachers can view student grades.	See student management and add student screenshots.
Teachers can add teachers.	See teacher management and add teacher screenshots.
Teachers can delete teachers.	See teacher management and delete teacher screenshots.
Teachers can change their passwords.	See teacher management and change password screenshots.
Teachers can create new tests.	See test management and test creation screenshots.
Teachers can edit tests.	See test management and test editing screenshots.
Teachers can remove tests.	See test management and test deletion screenshots.
All prompts are explanatory, and explicitly tell the user what to do.	See all screenshots.

Stage D: Documentation

Annotated Hardcopy of Output:

This output was generated by running the program in a terminal and copying/pasting the output into this file. For clarity's sake, user input is in **bold** and I have added comments in the form of [... Location …] to show at what stage the program was in before and after the screenshot happened.

Screenshot 1: Starting the Program / Main Menu

```
astronut@astronut:~/web/Dossier$ java Dossier
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

Comments: The menu is very clear, and allows the user to select the proper option.

Screenshot 2: Logging in a Teacher / Teacher Main Menu

```
[... From Main Menu ...]
Enter your choice: 2
Enter your teacher ID number: 42
Enter your password: lifeandall
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice:
```

Comments: The prompts tell the user exactly what they need to do.

Screenshot 3: Test Management Options

```
[... From Teacher Menu ...]
Enter your choice: 1
1.) Add New Test
2.) Show All Tests
3.) Edit a Test
4.) Remove a Test
5.) Return to Teacher Menu
Enter your choice:
```

Comments: The menu is very clear, and allows the user to select their desired action

Screenshot 4: Adding a new test

```
[ ... From Test Management Options ...]
Enter your choice: 1
Enter the ID for the new test: 78
Enter test password (up to 36 characters): sampletestpassword
How many questions: 4
Are students allowed to see their grades?
Enter 1 for yes, 0 for no: 1
Are students allowed to see the correct answers?
Enter 1 for yes, 0 for no: 1
Enter the question (200 character max): What is the first derivative of 2x^2
Enter answer choice A (100 character max): 4x
Enter answer choice B (100 character max): 8x^2
Enter answer choice C (100 character max): x
Enter answer choice D (100 character max): 12
Enter the correct choice (a, b, c, or d): a
Enter the question (200 character max): What is the largest planet?
Enter answer choice A (100 character max): Earth
Enter answer choice B (100 character max): Jupiter
Enter answer choice C (100 character max): Saturn
Enter answer choice D (100 character max): Neptune
Enter the correct choice (a, b, c, or d): b
Enter the question (200 character max): In what year was the United States'
Decleration of Indepdence signed?
Enter answer choice A (100 character max): 1884
Enter answer choice B (100 character max): 1642
Enter answer choice C (100 character max): 1400
Enter answer choice D (100 character max): 1776
Enter the correct choice (a, b, c, or d): d
Enter the question (200 character max): In what region of the world is Israel
located?
Enter answer choice A (100 character max): The Americas
Enter answer choice B (100 character max): The Indian Sub-continent
Enter answer choice C (100 character max): The Middle East
Enter answer choice D (100 character max): Africa
Enter the correct choice (a, b, c, or d): c
Is the test correct?
Enter 1 for yes, 0 for no: 1
Enter filename to save test to
Filename: sampletest.tst
Warning. Writing to file sampletest.tst will overwrite it. Continue? (Y/N): y
[...Program Returns to Test Management Menu...]
```

Comments: The process is clear, and the program guides the user through the necessary steps.

Screenshot 5: Return from test management options to teacher menu, go to student management menu and add a new student.

```
[... From Test Creation ...]
1.) Add New Test
2.) Show All Tests
3.) Edit a Test
4.) Remove a Test
5.) Return to Teacher Menu
Enter your choice: 5
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 2
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 1
Enter new student ID: 32
Enter student password: samplestudentpass
[... Student management menu ...]
```

Comments: The process is clear, and the program guides the user through the necessary steps.

Screenshot 6: Logout as teacher (in student management) and return to main menu

```
[... From Adding Student...]
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 4
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 4
Logging out...done
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

Comments: The process is self explanatory

Screenshot 7: Logging in as student and taking the test.

```
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice: 1
Enter your student ID number: 32
Enter your password: samplestudentpass
Enter the number of the test you wish to take: 78
Enter the test password (given to you by your teacher): sampletestpassword
Now starting test. Do not quit the program while in the middle of the test. If
you do, you will recieve a zero on the test. If this happens in error, please
contact your teacher.
What is the first derivative of 2x^2

a.) 8x^2
b.) 12
c.) x
d.) 4x

Enter your answer (a-d): d
You are correct.
What is the largest planet?

a.) Earth
b.) Jupiter
c.) Saturn
d.) Neptune

Enter your answer (a-d): a
You are incorrect.
The correct answer was: Jupiter
In what year was the United States' Decleration of Indepdence signed?

a.) 1884
b.) 1776
c.) 1400
d.) 1642

Enter your answer (a-d): b
You are correct.
In what region of the world is Israel located?

a.) Africa
b.) The Middle East
c.) The Americas
d.) The Indian Sub-continent

Enter your answer (a-d): b
You are correct.
Test Complete!
You made a score of: 75
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

Comments: The process is self explanatory and easily helps the student through it. A problem was incorrect and the program shows the correct answer. Note that the questions are given in random order.

Screenshot 8: Viewing grades

```
[... From Main Menu...]
Enter your choice: 2
Enter your teacher ID number: 42
Enter your password: lifeandall
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 2
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 2
1.) 32
Which student do you want to see grades for?
Enter student number: 1
Test ID#:       Grade:
78              75
0               0
0               0
0               0
0               0
0               0
0               0
0               0
0               0
0               0
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice:
```

Comments: The teacher is given a list of students, and can see their grades on the past 10 tests they took.

Screenshot 9: Deleting a student

```
[... Already in Student Management Menu...]
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 3
1.) 32
Which student do you want to delete?
Enter student number: 1
[ ... Program returns to Student Management Menu ...]
```

Comments: The process is extremely easy.

Screenshot 10: Adding another teacher

```
[... In teacher's menu ...]
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 3
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice: 1
Enter new teacher ID: 68
Enter teacher password: thisistheteacherspass
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice:
```

Comments: The process is extremely easy.

Screenshot 11: Changing the teacher's password

```
[... In teacher management menu ...]
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice: 2
Enter the ID of the teacher whose password you want to change:  68
Enter new password: thenewpass
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice:
```

Comments: The process is extremely easy.

Screenshot 12: Deleting a teacher

```
[... In teacher management menu ...]
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice: 3
Enter the ID of the teacher whom you want to delete: 68
[ ... Returns to teacher management menu ... ]
```

Comments: The process is extremely easy.

Screenshot 13: Editing a test

```
[... In teacher menu ...]
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 1
1.) Add New Test
2.) Show All Tests
3.) Edit a Test
4.) Remove a Test
5.) Return to Teacher Menu
Enter your choice: 3
Enter the ID for the test to edit: 78
Test Details:
Test ID: 78
Test (p)assword: sampletestpassword
Show (a)nswers: yes
Show (g)rades: yes
1.) What is the first derivative of 2x^2
2.) What is the largest planet?
3.) In what year was the United States' Decleration of Indepdence signed?
4.) In what region of the world is Israel located?
What do you want to edit? Type "o" for test options or "q" to edit a question
Edit : q
Which question do you want to edit?
Enter question number (1-4): 2
Question: What is the largest planet?
a.) Earth
b.) Jupiter
c.) Saturn
d.) Neptune
Correct Answer: b
Modify what?
        Question (T)ext
        Answer (A)
        Answer (B)
        Answer (C)
        Answer (D)
        The (R)ight Answer
or (Q)uit editing
Enter Your Choice: c
Enter New Answer Choice (Max 100 characters):
> Mars
Question: What is the largest planet?
a.) Earth
b.) Jupiter
c.) Mars
d.) Neptune
Correct Answer: b
Modify what?
        Question (T)ext
        Answer (A)
        Answer (B)
        Answer (C)
        Answer (D)
        The (R)ight Answer
or (Q)uit editing
Enter Your Choice: q
Warning. Writing to file sampletest.tst will overwrite it. Continue? (Y/N): y
```

Screenshot 14: Deleting a test

```
[... In test management menu ...]
1.) Add New Test
2.) Show All Tests
3.) Edit a Test
4.) Remove a Test
5.) Return to Teacher Menu
Enter your choice: 4
Enter the ID for the test to delete: 78
Test Deleted!
1.) Add New Test
2.) Show All Tests
3.) Edit a Test
4.) Remove a Test
5.) Return to Teacher Menu
Enter your choice:
```

Comments: The process is extremely easy. The prompts tell the teacher exactly what is needed. This will remove the test both from the database and from disk.

Screenshot 15: Detecting User Error (Bad Input)

```
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice: this is a string, not an int

Warning! That was not an Integer!
Enter an Integer: 0.2

Warning! That was not an Integer!
Enter an Integer: 1
Enter your student ID number:
[ ... ]
```

Comments: This will go on indefinitely until the user puts in the correct type of data.

Screenshot 16: Detecting User Error (No Student)

```
[ ... ]
Enter your student ID number: 31
Student number 31 not found!
Enter your student ID number: 52
Student number 52 not found!
Enter your student ID number: 62
Student number 62 not found!
Enter your student ID number: 12
Student not found! Returning to main menu.
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

Comments: After a few tries, the program will give up.

Screenshot 17: Detecting User Error (Choosing invalid option)

```
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 5
Invalid Choice!
Enter your choice: 6
Invalid Choice!
Enter your choice: 2
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice:
```

Comments: This is widely repeated throughout the program.

Screenshot 18: Detecting System Error

For this demonstration, I deleted a file mid-action to demonstrate that the program can handle I/O Errors.

```
[ ... Student Management Menu ... ]
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 2
1.) 65
2.) 423
Which student do you want to see grades for?
[ At this point, the file students.db was deleted ]
Enter student number: 1
An I/O Error Occured.
The exact error follows:
        students.db (No such file or directory)
astronut@astronut:~/web/Dossier$
```

Comments: The program quits with an explanatory message because there is no way to handle a system error.

Evaluation of Solution:

The first question that comes to mind when evaluating the dossier is: "Does it work?" The answer to this question is a resounding "Yes!" As evidenced by the screenshots above in the "Annotated Output" section, the program is fully functional and carries out its appointed task. The next question is: "Does the program fulfill its design goals?" The answer to this is also yes, as evidenced by the table below:

Goal	Documentation
The student is guided through the login and test taking process	See student login and test screenshots.
Teachers can add students.	See student management and add student screenshot.
Teachers can delete students.	See student management and delete student screenshots.
Teachers can view student grades.	See student management and add student screenshots.
Teachers can add teachers.	See teacher management and add teacher screenshots.
Teachers can delete teachers.	See teacher management and delete teacher screenshots.
Teachers can change their passwords.	See teacher management and change password screenshots.
Teachers can create new tests.	See test management and test creation screenshots.
Teachers can edit tests.	See test management and test editing screenshots.
Teachers can remove tests.	See test management and test deletion screenshots.

All prompts are explanatory, and explicitly tell the user what to do.	See all screenshots.

Thus, we can see that all design goals are met. However, there are several limitations to the program:

- It has a very rigid structure. Only one specific type of question (multiple choice) is possible
- All strings involved in the program have a fixed maximum length
- The text only interface is antiquated.

There are several possible enhancements:

- A graphical user interface. The current code would be very easy to tie into an alternate interface due to its modular nature. In fact, many similar programs could be built on top of the infrastructure developed for this program.
- The lengths of the questions, answers and passwords are all arbitrary and easy to change. In order to make this as easy as possible, they are all set by declaring final ints for the classes. To increase the max values is only a matter of changing one number at the top of the file.

Based on the success shown in the table above, I feel that the initial design was quite appropriate, and served me well during the writing of the dossier.

User Documentation:

There are two documents, one for teachers and one for students.

Teacher

- **Installation:** Installation of this program requires that all files be placed in a directory accessible to students. If possible, allow students to run the program in a way that they do not have access do the data files (Setting the program setuid in UNIX and UNIX like operating systems, or using ACL's in the NT line of Windows). The following files need to be in that directory:
    - Dossier.class
    - DoubleLinkedList$DLLNode.class
    - DoubleLinkedList.class
    - GetInput.class
    - NoSuchTeacherException.class
    - NoSuchTestException.class
    - Question.class
    - TestNotFoundException.class
    - StudentDB.class
    - StudentNotFoundException.class
    - Teacher.class
    - TeacherDB.class
    - Test.class
    - Student.class
    - TestDB.class
    - TestTest.class
    - students.db
    - tests.db
    - teachers.db

    Then prepare a shortcut for students to use to launch the program (on most systems "java Dossier". A shortcut can be used in windows and a simple shell script in UNIX/UNIX like operating systems. The default teacher is "42" and the password is "lifeandall". **Important: If you delete teachers.db, no one will be able to log in.** Add the necessary teachers and students as described below, and distribute the student instructions to all students.
- **Basic Tasks**
    - **Logging in:** Select "Login Teacher" at the main menu and enter your user name and password like in the following example:

Logging In

```
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice: 2
Enter your teacher ID number: 42
Enter your password: lifeandall
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice:
```

- **Add a teacher:** This is done through the “Teacher Management” menu, accessed via the teacher main menu. Select the “Add a teacher” option and follow the example below:

Adding another teacher

```
[... In teacher’s menu ...]
---Teacher Options---
        1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 3
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice: 1
Enter new teacher ID: 68
Enter teacher password: thisistheteacherspass
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice:
```

Adding a student: This is done through the "Student Management" menu, accessed via the teacher main menu. Select the "Add a Student" option and follow the example below:

Adding a Student:

```
---Teacher Options---
1.) Manage Tests
2.) Manage Students
3.) Manage Teachers
4.) Log out and return to main menu
Enter your choice: 2
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 1
Enter new student ID: 32
Enter student password: samplestudentpass
```

- **Deleting a Student:** This is done through the "Student Management" menu, accessed via the teacher main menu. Select the "Delete a Student" option and follow the example below:

Deleting a Student

```
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 3
1.) 32
Which student do you want to delete?
Enter student number: 1
[ ... Program returns to Student Management Menu
```

- **Viewing a Student's grades:** This is done through the "Student Management" menu, accessed via the teacher main menu. Select the "Shw Student Grades" option and follow the example below to see the grades a student made on their last 10 tests:

Viewing Grades

```
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 2
1.) 32
Which student do you want to see grades for?
Enter student number: 1
Test ID#:       Grade:
78              75
0               0
0               0
0               0
0               0
0               0
0               0
0               0
0               0
0               0
```

- **Changing the teacher's password:** This is done through the teacher management menu. Follow the example below:

```
[... In teacher management menu ...]
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice: 2
Enter the ID of the teacher whose password you want to change:  68
Enter new password: thenewpass
1.) Add Teacher
2.) Change Teacher Password
3.) Delete Teacher
4.) Return to Teacher Menu
Enter your choice:
```

- **Adding a test:** This is done through the “Test Management” menu, accessed via the teacher main menu. Select the “Add a Test” option and follow the example below:

```
[ ... From Test Management Options ...]
Enter your choice: 1
Enter the ID for the new test: 78
Enter test password (up to 36 characters): sampletestpassword
How many questions: 4
Are students allowed to see their grades?
Enter 1 for yes, 0 for no: 1
Are students allowed to see the correct answers?
Enter 1 for yes, 0 for no: 1
Enter the question (200 character max): What is the first derivative of 2x^2
Enter answer choice A (100 character max): 4x
Enter answer choice B (100 character max): 8x^2
Enter answer choice C (100 character max): x
Enter answer choice D (100 character max): 12
Enter the correct choice (a, b, c, or d): a
Enter the question (200 character max): What is the largest planet?
Enter answer choice A (100 character max): Earth
Enter answer choice B (100 character max): Jupiter
Enter answer choice C (100 character max): Saturn
Enter answer choice D (100 character max): Neptune
Enter the correct choice (a, b, c, or d): b
Enter the question (200 character max): In what year was the United States'
Decleration of Indepdence signed?
Enter answer choice A (100 character max): 1884
Enter answer choice B (100 character max): 1642
Enter answer choice C (100 character max): 1400
Enter answer choice D (100 character max): 1776
Enter the correct choice (a, b, c, or d): d
Enter the question (200 character max): In what region of the world is Israel
located?
Enter answer choice A (100 character max): The Americas
Enter answer choice B (100 character max): The Indian Sub-continent
Enter answer choice C (100 character max): The Middle East
Enter answer choice D (100 character max): Africa
Enter the correct choice (a, b, c, or d): c
Is the test correct?
Enter 1 for yes, 0 for no: 1
Enter filename to save test to
Filename: sampletest.tst
Warning. Writing to file sampletest.tst will overwrite it. Continue? (Y/N): y
[...Program Returns to Test Management Menu...]
```

The remaining actions are rarely taken, but are self explanatory. Follow the prompts through the program!

Student:

Start the program according to the instructions your teacher gave you.

You will end up at the main menu, which looks like this:

```
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

Type '**1**' to log in as a student, and enter your user name and password at the prompts. The program will ask you for the test id number and password. These will be given to you by your teacher and are different for each test. After you enter this information, the test will start.

Important! If you quit the test in the middle, you will receive a zero for it and will not get another try. If this happens, contact your teacher for help.

When the test starts, you will be asked several multiple choice questions. Each one will have four possible answer choices. Read the question and all four possible choices. Enter the letter of the correct answer. If your teacher has allowed it, the program may show whether you are right or wrong as well as the correct answer. At the end of the test, the program will show your grade (if the teacher allowed that). This process is demonstrated in the following picture:

```
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice: 1
Enter your student ID number: 32
Enter your password: samplestudentpass
Enter the number of the test you wish to take: 78
Enter the test password (given to you by your teacher): sampletestpassword
Now starting test. Do not quit the program while in the middle of the test. If
you do, you will recieve a zero on the test. If this happens in error, please
contact your teacher.
What is the first derivative of 2x^2

a.) 8x^2
b.) 12
c.) x
d.) 4x

Enter your answer (a-d): d
You are correct.
What is the largest planet?

a.) Earth
b.) Jupiter
c.) Saturn
d.) Neptune

Enter your answer (a-d): a
You are incorrect.
The correct answer was: Jupiter
In what year was the United States' Decleration of Indepdence signed?

a.) 1884
b.) 1776
c.) 1400
d.) 1642

Enter your answer (a-d): b
You are correct.
In what region of the world is Israel located?

a.) Africa
b.) The Middle East
c.) The Americas
d.) The Indian Sub-continent

Enter your answer (a-d): b
You are correct.
Test Complete!
You made a score of: 75
1.) Login Student
2.) Login Teacher
3.) Quit
Enter your choice:
```

```java
/*
 * Dossier.java
 *
 * The main class of my Dossier.
 * Provides entry code, etc
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */

import java.io.IOException;
import java.io.FileNotFoundException;

public class Dossier {
        public static final String TEST_DB_NAME = "tests.db"; // Where the test database is stored
        public static final String STUDENT_DB_NAME = "students.db"; // Where the student database is stored
        public static final String TEACHER_DB_NAME = "teachers.db"; // Where the student database is stored

        // Variables to be used throughout this class.
        // This class will never be instantiated so they are static
        static TestDB tests;
        static StudentDB students;
        static TeacherDB teachers;

        public static void main(String[] args) {
                // Load test database
                tests = new TestDB(TEST_DB_NAME);
                // Load teacher database
                teachers = new TeacherDB(TEACHER_DB_NAME);
                // Load student databse
                 students = new StudentDB(STUDENT_DB_NAME);
                // Main program loop
                while(true) { // Will abort with System.exit()
                        // Prompt for Student/Teacher login or quit
                        System.out.println("1.) Login Student");
                        System.out.println("2.) Login Teacher");
                        System.out.println("3.) Quit");
                        System.out.print("Enter your choice: ");
                        int choice = GetInput.getInt();
                        while (!(choice > 0 && choice <4)) {
                                System.out.println("That was not a valid choice!");
                                System.out.println("Enter your choice: ");
                                choice = GetInput.getInt();
                        }
                        // Branch here
                        switch(choice) { // No need for 'default' since only possible values are 1-3
                                case 1:
                                        studentLogin();
                                break;
                                case 2:
                                        teacherLogin();
```

```
53                                      break;
54                                      case 3:
55                                              System.exit(0);
56                                      break;
57                              }
58                      }
59              }
60
61
62              // Teacher Code
63              public static void teacherLogin() {
64                      // Login part 1 - Find teacher
65                      System.out.print("Enter your teacher ID number: ");
66                      int id = GetInput.getInt();
67                      int c = 1;
68                      Teacher teach = null;
69                      try {
70                              teach = teachers.findTeacher(id); // Find teacher
71                              while (teach == null && c++ <=3) { // give teachers 3
    chances, c incremented in loop conditional
72                                      System.out.println("Teacher number " + id + " not
    found!");
73                                      System.out.print("Enter your teacher ID number: ");
74                                      id = GetInput.getInt();
75                                      teach = teachers.findTeacher(id);
76                              }
77                      }
78                      catch(IOException e) {
79                              System.out.println("An I/O Error Occured.");
80                              System.out.println("The exact error follows: ");
81                              System.out.println("\t" + e.getMessage());
82                              System.exit(1);
83                      }
84                      if (teach == null) { // Teacher not found, go back to main menu
85                              System.out.println("Teacher not found! Returning to main
    menu.");
86                              return;
87                      }
88                      // Login part 2 - Verify password
89                      System.out.print("Enter your password: ");
90                      String passwd = GetInput.getString();
91                      c = 1;
92                      while (!passwd.equals(teach.getPass()) && c++ <= 3) { // again, 3
    tries, and c is incrememnted in loop conditional
93                              System.out.println("Password Incorrect: ");
94                              System.out.print("Enter your password: ");
95                              passwd = GetInput.getString();
96                      }
97                      if (!(passwd.equals(teach.getPass()))) { // Teacher missed password
    three tries
98                              System.out.println("Login Failed! Returning to main menu.");
99                              return;         // Go back to main menu
100                     }
101                     // First menu-loop
102                     int choice = 0;
103                     while (true) { // break loop with a return
```

```
104                         System.out.println("---Teacher Options---");
105                         System.out.println("1.) Manage Tests");
106                         System.out.println("2.) Manage Students");
107                         System.out.println("3.) Manage Teachers");
108                         System.out.println("4.) Log out and return to main menu");
109                         System.out.print("Enter your choice: ");
110                         choice = GetInput.getInt();
111                         while (!(choice >0 && choice < 5)) { // Check for choice out
    of range
112                                 System.out.println("Invalid Choice!");
113                                 System.out.print("Enter your choice: ");
114                                 choice = GetInput.getInt();
115                         }
116                         // Dispatch to helper functions
117                         switch(choice) {
118                                 case 1:
119                                         manageTests();
120                                 break;
121                                 case 2:
122                                         manageStudents();
123                                 break;
124                                 case 3:
125                                         manageTeachers();
126                                 break;
127                                 case 4:
128                                         System.out.println("Logging out...done");
129                                         return;
130                         }
131                 }
132         }
133
134
135         // Function to manage students.
136         // For use by the teacher
137         public static void manageStudents() {
138                 // Manage student loop
139                 while (true) { // break with return
140                         System.out.println("1.) Add Student");
141                         System.out.println("2.) Show Student Grades");
142                         System.out.println("3.) Delete Student");
143                         System.out.println("4.) Return to Teacher Menu");
144                         System.out.print("Enter your choice: ");
145                         int choice = GetInput.getInt();
146                         while (!(choice > 0 && choice < 5)) { // Bad input
147                                 System.out.println("Invalid Response!");
148                                 System.out.print("Enter your choice: ");
149                                 choice = GetInput.getInt();
150                         }
151                         switch (choice) {
152                                 case 1:
153                                         // Add Student
154                                         try {
155                                                 System.out.print("Enter new student
    ID: ");
156                                                 int stdID = GetInput.getInt();
157                                                 while (students.findByID(stdID) !=
```

```
    -1) {
158                                                     System.out.println("Student
    ID " + stdID + " already exists. Choose another");
159                                                     System.out.print("Enter new
    student ID: ");
160                                                     stdID = GetInput.getInt();
161                                             }
162                                             System.out.print("Enter student
    password: ");
163                                             String pass = GetInput.getString();
164                                             students.addNewStudent(new
    Student(stdID, pass));
165                                     }
166                                     catch (IOException e) {
167                                             System.out.println("An I/O Error
    Occured.");
168                                             System.out.println("The exact error
    follows: ");
169                                             System.out.println("\t" +
    e.getMessage());
170                                             System.exit(1);
171                                     }
172                             break;
173                             case 2:
174                                     try {
175                                             // Pick Student
176                                             students.showAll(); // This will
    show all students, with the number <index> + 1
177                                             System.out.println("Which student do
    you want to see grades for?");
178                                             System.out.print("Enter student
    number: ");
179                                             int sindex = GetInput.getInt();
180                                             while (!(sindex > 0 && sindex <=
    students.count())) {
181                                                     System.out.println("That's
    not a valid choice!");
182                                                     System.out.print("Enter
    student number: ");
183                                                     sindex = GetInput.getInt();
184                                             }
185                                             Student std = students.read(sindex
    -1 ); // Offset for human readability
186                                             // Show Grades
187                                             std.showGrades();
188                                     }
189                                     catch (IOException e) {
190                                             System.out.println("An I/O Error
    Occured.");
191                                             System.out.println("The exact error
    follows: ");
192                                             System.out.println("\t" +
    e.getMessage());
193                                             System.exit(1);
194                                     }
195                             break;
```

```
196                                 case 3:
197                                         try {
198                                                 // Pick Student
199                                                 students.showAll(); // This will
    show all students, with the number <index> + 1
200                                                 System.out.println("Which student do
    you want to delete?");
201                                                 System.out.print("Enter student
    number: ");
202                                                 int sindex = GetInput.getInt();
203                                                 while (!(sindex > 0 && sindex <=
    students.count())) {
204                                                         System.out.println("That's
    not a valid choice!");
205                                                         System.out.print("Enter
    student number: ");
206                                                         sindex = GetInput.getInt();
207                                                 }
208                                                 Student std = students.read(sindex
    -1 ); // Offset for human readability
209                                                 // Delete student
210                                                 students.delete(std);
211                                         }
212                                         catch (StudentNotFoundException e) {
213                                                 System.out.println("A serious error
    has occured.");
214                                                 System.out.println("The student
    database has been corrupted or tampered with");
215                                                 System.exit(1);
216                                         }
217                                         catch (IOException e) {
218                                                 System.out.println("An I/O Error
    Occured.");
219                                                 System.out.println("The exact error
    follows: ");
220                                                 System.out.println("\t" +
    e.getMessage());
221                                                 System.exit(1);
222                                         }
223                                 break;
224                                 case 4:
225                                 return;
226                         }
227                 }
228         }
229
230         // Function to manage teachers
231         // For use by the teacher
232         public static void manageTeachers() {
233                 // Manage teachers loop
234                 while (true) { // break with return
235                         System.out.println("1.) Add Teacher");
236                         System.out.println("2.) Change Teacher Password");
237                         System.out.println("3.) Delete Teacher");
238                         System.out.println("4.) Return to Teacher Menu");
239                         System.out.print("Enter your choice: ");
```

```
240                         int choice = GetInput.getInt();
241                         while (!(choice > 0 && choice < 5)) { // Bad input
242                                 System.out.println("Invalid Response!");
243                                 System.out.print("Enter your choice: ");
244                                 choice = GetInput.getInt();
245                         }
246                         switch (choice) {
247                                 case 1:
248                                         // Add teacher
249                                         try {
250                                                 System.out.print("Enter new teacher
    ID: ");
251                                                 int teachID = GetInput.getInt();
252                                                 while (teachers.findTeacher(teachID)
    != null) {
253                                                         System.out.println("Teacher
    ID " + teachID + " already exists. Choose another");
254                                                         System.out.print("Enter new
    teacher ID: ");
255                                                         teachID = GetInput.getInt();
256                                                 }
257                                                 System.out.print("Enter teacher
    password: ");
258                                                 String pass = GetInput.getString();
259                                                 teachers.addTeacher(teachID, pass);
260                                         }
261                                         catch (IOException e) {
262                                                 System.out.println("An I/O Error
    Occured.");
263                                                 System.out.println("The exact error
    follows: ");
264                                                 System.out.println("\t" +
    e.getMessage());
265                                                 System.exit(1);
266                                         }
267                                 break;
268                                 case 2:
269                                         // Change a password
270                                         try {
271                                                 System.out.print("Enter the ID of
    the teacher whose password you want to change:  ");
272                                                 // We don't show IDs to discurage
    changing other teachers' passwords
273                                                 int teachID = GetInput.getInt();
274                                                 while (teachers.findTeacher(teachID)
    == null) {
275                                                         System.out.println("Invalid
    ID!");
276                                                         System.out.print("Enter the
    ID of the teacher whose password you want to change: ");
277                                                         teachID = GetInput.getInt();
278                                                 }
279                                                 Teacher t =
    teachers.findTeacher(teachID);
280                                                 System.out.print("Enter new
    password: ");
```

```
281                                             t.setPass(GetInput.getString());
282                                             teachers.changePass(t);
283                                     }
284                                     catch (IOException e) {
285                                             System.out.println("An I/O Error
    Occured.");
286                                             System.out.println("The exact error
    follows: ");
287                                             System.out.println("\t" +
    e.getMessage());
288                                             System.exit(1);
289                                     }
290                                     catch (NoSuchTeacherException e) { // should
    never happen
291                                             System.out.println("A serious error
    has occured.");
292                                             System.out.println("The teacher
    database has been corrupted or tampered with");
293                                             System.exit(1);
294                                     }
295                                 break;
296                                 case 3:
297                                     // Delete a teacher
298                                     try {
299                                             System.out.print("Enter the ID of
    the teacher whom you want to delete: ");
300                                             // We don't show IDs to discurage
    changing other teachers' passwords
301                                             int teachID = GetInput.getInt();
302                                             while (teachers.findTeacher(teachID)
    == null) {
303                                                     System.out.println("Invalid
    ID!");
304                                                     System.out.print("Enter the
    ID of the teacher whom you want to delete: ");
305                                                     teachID = GetInput.getInt();
306                                             }
307                                             teachers.delete(teachID);
308                                     }
309                                     catch (IOException e) {
310                                             System.out.println("An I/O Error
    Occured.");
311                                             System.out.println("The exact error
    follows: ");
312                                             System.out.println("\t" +
    e.getMessage());
313                                             System.exit(1);
314                                     }
315                                     catch (NoSuchTeacherException e) { // should
    never happen
316                                             System.out.println("A serious error
    has occured.");
317                                             System.out.println("The teacher
    database has been corrupted or tampered with");
318                                             System.exit(1);
319                                     }
```

```
320                                 break;
321                                 case 4: // Return to teacher menu
322                                 return;
323                         }
324                 }
325         }
326
327         // Function to manage tests
328         // For use by the teacher
329         public static void manageTests() {
330                 while(true) { // return with return statement
331                         System.out.println("1.) Add New Test");
332                         System.out.println("2.) Show All Tests");
333                         System.out.println("3.) Edit a Test");
334                         System.out.println("4.) Remove a Test");
335                         System.out.println("5.) Return to Teacher Menu");
336                         System.out.print("Enter your choice: ");
337                         int choice = GetInput.getInt();
338                         while (!(choice > 0 && choice < 6)) { // Bad input
339                                 System.out.println("Invalid Response!");
340                                 System.out.print("Enter your choice: ");
341                                 choice = GetInput.getInt();
342                         }
343                         switch(choice) {
344                                 case 1:
345                                         // Add a test
346                                         try{
347                                                 System.out.print("Enter the ID for
    the new test: ");
348                                                 int tid = GetInput.getInt();
349                                                 while
    (!tests.findTest(tid).equals("")) { // exists
350                                                         System.out.println("Test
    number " + tid + " already exists!");
351                                                         System.out.print("Enter the
    ID for the new test: ");
352                                                         tid = GetInput.getInt();
353                                                 }
354                                                 Test t = new Test(tid);
355                                                 tests.addTest(tid,t.buildTest()); //
    createTest() returns filename
356                                         }
357                                         catch(IOException e) {
358                                                 System.out.println("An I/O Error
    Occured.");
359                                                 System.out.println("The exact error
    follows: ");
360                                                 System.out.println("\t" +
    e.getMessage());
361                                                 System.exit(1);
362                                         }
363                                 break;
364                                 case 2:
365                                         // Show all tests
366                                         try {
367                                                 tests.showAll();
```

```
368                                             }
369                                             catch(IOException e) {
370                                                     System.out.println("An I/O Error
    Occured.");
371                                                     System.out.println("The exact error
    follows: ");
372                                                     System.out.println("\t" +
    e.getMessage());
373                                                     System.exit(1);
374                                             }
375                                     break;
376                                     case 3:
377                                             // Edit a test
378                                             try {
379                                                     System.out.print("Enter the ID for
    the test to edit: ");
380                                                     int tid = GetInput.getInt();
381                                                     while
    (tests.findTest(tid).equals("")) { // exists
382                                                             System.out.println("Test
    number " + tid + " doesn't exist!");
383                                                             System.out.print("Enter the
    ID for the test to edit: ");
384                                                             tid = GetInput.getInt();
385                                                     }
386                                                     String fname = tests.findTest(tid);
387                                                     Test toEdit = new Test(fname); //
    Read from file
388                                                     toEdit.editTest();
389                                                     toEdit.writeToFile(fname);
390                                             }
391                                             catch(IOException e) {
392                                                     System.out.println("An I/O Error
    Occured.");
393                                                     System.out.println("The exact error
    follows: ");
394                                                     System.out.println("\t" +
    e.getMessage());
395                                                     System.exit(1);
396                                             }
397                                     break;
398                                     case 4:
399                                             // Delete a test
400                                             try {
401                                                     System.out.print("Enter the ID for
    the test to delete: ");
402                                                     int tid = GetInput.getInt();
403                                                     while
    (tests.findTest(tid).equals("")) { // exists
404                                                             System.out.println("Test
    number " + tid + " doesn't exist!");
405                                                             System.out.print("Enter the
    ID for the test to delete: ");
406                                                             tid = GetInput.getInt();
407                                                     }
408                                                     String fname = tests.findTest(tid);
```

```
409                                                     tests.removeTest(tid); // remove
    from database
410                                                     new java.io.File(fname).delete(); //
    delete on disk
411                                                     System.out.println("Test Deleted!");
412                                             }
413                                             catch(IOException e) {
414                                                     System.out.println("An I/O Error
    Occured.");
415                                                     System.out.println("The exact error
    follows: ");
416                                                     System.out.println("\t" +
    e.getMessage());
417                                                     System.exit(1);
418                                             }
419                                             catch(NoSuchTestException e) {  // Should
    never happen since we tested earlier
420                                                     System.out.println("A serious error
    has occured.");
421                                                     System.out.println("The test
    database has been corrupted or tampered with");
422                                                     System.exit(1);
423                                             }
424                                     break;
425                                     case 5:
426                                     return; //Return to menu
427                             }
428                     }
429             }
430
431             // Student Code
432             private static void studentLogin() {
433                     // Login  Part 1: Load student
434                     System.out.print("Enter your student ID number: ");
435                     int id = GetInput.getInt();
436                     int c = 1;
437                     Student std = null;
438                     try {
439                             int index = students.findByID(id); // Find position in file
440                             while (index == -1 && c++ <=3) { // give students 3 chances,
    c incremented in loop conditional
441                                     System.out.println("Student number " + id + " not
    found!");
442                                     System.out.print("Enter your student ID number: ");
443                                     id = GetInput.getInt();
444                                     index = students.findByID(id);
445                             }
446                             if (index == -1) { // Student not found, go back to main
    menu
447                                     System.out.println("Student not found! Returning to
    main menu.");
448                                     return;
449                             }
450                             std = students.read(index);
451                     }
452                     catch(IOException e) {
```

```
453                         System.out.println("An I/O Error Occured.");
454                         System.out.println("Please contact your teacher for further
    assistance.");
455                         System.out.println("The exact error follows: ");
456                         System.out.println("\t" + e.getMessage());
457                         System.exit(1); // Exit with error code; we don't know the
    cause of the error and it's unsafe to continue
458                 }
459                 // Login Part 2 - Check Password
460                 System.out.print("Enter your password: ");
461                 String passwd = GetInput.getString();
462                 c = 1;
463                 while (!passwd.equals(std.getPass()) && c++ <= 3) { // again, 3
    tries, and c is incrememnted in loop conditional
464                         System.out.println("Password Incorrect: ");
465                         System.out.print("Enter your password: ");
466                         passwd = GetInput.getString();
467                 }
468                 if (!(passwd.equals(std.getPass()))) { // Student missed password
    three tries
469                         System.out.println("Login Failed! Returning to main menu.");
470                         return;         // Go back to main menu
471                 }
472
473                 // Pick test
474                 System.out.print("Enter the number of the test you wish to take: ");
475                 int testid = GetInput.getInt();
476                 c = 1;
477                 String tfname = "";
478                 try {
479                         tfname = tests.findTest(testid); // Find the filename of
    that test
480                         while(tfname.equals("") && c++ <= 3) { // again, 3 tries,
    and c is incrememnted in loop conditional
481                                 System.out.println("Test number " + testid + " + not
    found!");
482                                 System.out.print("Enter the number of the test you
    wish to take (given to you by your teacher): ");
483                                 testid = GetInput.getInt();
484                                 tfname = tests.findTest(testid); // Find the
    filename of that test
485                         }
486                 }
487                 catch (IOException e) {
488                         System.out.println("An I/O Error Occured.");
489                         System.out.println("Please contact your teacher for further
    assistance.");
490                         System.out.println("The exact error follows: ");
491                         System.out.println("\t" + e.getMessage());
492                         System.exit(1); // Exit with error code; we don't know the
    cause of the error and it's unsafe to continue
493                 }
494                 if (tfname.equals("")) { // Test not found, go back to main menu
495                         System.out.println("Test not found! Returning to main
    menu.");
496                         return;
```

```
497                 }
498
499                 if (std.scoreOn(testid) != -1) {
500                         System.out.println("You have already taken this test!");
501                         return;
502                 }
503
504                 // Load the test
505                 // Make sure to catch IO exception
506                 Test currentTest = null;
507                 try {
508                         currentTest = new Test(tfname);
509                 }
510                 catch(FileNotFoundException e) { // the most likely reason for
    failure is a missing test
511                         System.out.println("Error! The file " + tfname + " was not
    found.");
512                         System.out.println("Please contact your teacher for further
    assistance.");
513                         return; // Go back to main menu
514                 }
515                 catch(IOException e) { // Only one catch() will run, so this
    generically handles all other IO errors
516                         System.out.println("An I/O Error Occured.");
517                         System.out.println("Please contact your teacher for further
    assistance.");
518                         System.out.println("The exact error follows: ");
519                         System.out.println("\t" + e.getMessage());
520                         System.exit(1); // Exit with error code; we don't know the
    cause of the error and it's unsafe to continue
521                 }
522
523                 // Verify the test password
524                 System.out.print("Enter the test password (given to you by your
    teacher): ");
525                 passwd = GetInput.getString();
526                 c = 1;
527                 while (!passwd.equals(currentTest.getPasswd()) && c++ <= 3) { //
    Familiar loop
528                         System.out.println("Password Incorrect: ");
529                         System.out.print("Enter your password: ");
530                         passwd = GetInput.getString();
531                 }
532                 if (!(passwd.equals(currentTest.getPasswd()))) { // Student missed
    password three tries
533                         System.out.println("Test Authentication Failed! Returning to
    main menu.");
534                         return;         // Go back to main menu
535                 }
536
537                 // Set score of 0
538                 std.addScore(testid, 0);
539                 try {
540                         students.save(std);
541                 }
542                 catch (IOException e) {
```

```
543                         System.out.println("An I/O Error Occured.");
544                         System.out.println("Please contact your teacher for further
    assistance.");
545                         System.out.println("The exact error follows: ");
546                         System.out.println("\t" + e.getMessage());
547                         System.exit(1); // Exit with error code; we don't know the
    cause of the error and it's unsafe to continue
548                 }
549                 catch (StudentNotFoundException e) { // This should never happen -
    Student was read out of database
550                         System.out.println("A serious error has occured.");
551                         System.out.println("The student database has been corrupted
    or tampered with");
552                         System.out.println("Contact your teacher for more help");
553                         System.exit(1);
554                 }
555
556                 // Take test
557                 System.out.println("Now starting test. Do not quit the program while
    in the middle of the test. If you do, you will recieve a zero on the test. If this
    happens in error, please contact your teacher.");
558                 int score = currentTest.takeTest();
559
560                 // Set proper score
561                 try{
562                         std.replaceScore(testid, score);
563                 }
564                 catch (TestNotFoundException e) { // Should never happen
565                         System.out.println("A serious error has occured.");
566                         System.out.println("The student database has been corrupted
    or tampered with");
567                         System.out.println("Contact your teacher for more help");
568                         System.exit(1);
569                 }
570                 try {
571                         students.save(std);
572                 }
573                 catch(IOException e) {
574                         System.out.println("An I/O Error Occured.");
575                         System.out.println("Please contact your teacher for further
    assistance.");
576                         System.out.println("The exact error follows: ");
577                         System.out.println("\t" + e.getMessage());
578                         System.exit(1); // Exit with error code; we don't know the
    cause of the error and it's unsafe to continue
579                 }
580                 catch (StudentNotFoundException e) { // Should never happen
581                         System.out.println("A serious error has occured.");
582                         System.out.println("The student database has been corrupted
    or tampered with");
583                         System.out.println("Contact your teacher for more help");
584                         System.exit(1);
585                 }
586
587                 // Return to main menu
588         }
```

```
589 }
590
```

```java
/*
/
/ Doubly Linked List
/
/ DoubleLinkedList Class
/
/ The implementation of a doubly linked
/ list of Objects. Provides most methods
/ needed.
/
/ (C) Benjamin Seidenberg 2005, 2006
/
*/

import java.util.NoSuchElementException;
import java.util.Iterator;
import java.util.ArrayList;

public class DoubleLinkedList {

        ////////////////////
        // Private Variables
        ////////////////////

        private DLLNode head;           // Beginning of list
        private DLLNode tail;           // End of list

        ////////////////////
        // Public Functions
        ////////////////////

        // Constructor
        public DoubleLinkedList() {
                head = null;
                tail = null;
        }

        // Appends to end of list
        public void append(Object x) {
                if (head == null || tail == null) {
                        head = new DLLNode(x);
                        head.prev = null;
                        head.next = null;
                        tail = head;
                        return;
                }

                tail.next = new DLLNode(x);
                tail.next.prev = tail;
                tail.next.next = null;
                tail = tail.next;
        }

        // Prepends to beginning of the list
        public void prepend(Object x) {
                if (head == null || tail == null) {
```

```
57                      head = new DLLNode(x);
58                      head.prev = null;
59                      head.next = null;
60                      tail = head;
61                      return;
62              }
63
64              head.prev = new DLLNode(x);
65              head.prev.prev = null;
66              head.prev.next = head;
67              head = head.prev;
68      }
69
70      // Dumps list to screen (Default is fwd)
71      public void dump() {
72              dumpFwd();
73      }
74
75      // Dumps list to screen (From head to tail)
76      public void dumpFwd() {
77              DLLNode pos = head;
78              while(pos != null) {
79                      System.out.println(pos.data);
80                      pos = pos.next;
81              }
82      }
83
84      // Dumps list to screen (From tail to head)
85      public void dumpBack() {
86              DLLNode pos = tail;
87              while(pos != null) {
88                      System.out.println(pos.data);
89                      pos = pos.prev;
90              }
91      }
92
93      // Returns if the list is empty
94      public boolean isEmpty() {
95              return (head == null || tail == null);
96      }
97
98      // Returns number of nodes in the list
99      public int size() {
100             int count = 0;
101             DLLNode pos = head;
102             while (pos != null) {
103                     count++;
104                     pos = pos.next;
105             }
106             return count;
107     }
108
109     // Returns whether an object is in the list
110     public boolean isIn(Object x) {
111             return (whereAt(x) != null);
112     }
```

```
113
114         // Deletes the node containing the passed in object
115         public void delete(Object o) throws NoSuchElementException {
116                 DLLNode x = whereAt(o);
117                 if (x == null)
118                         throw new NoSuchElementException("Item not found in the
    list");
119                 delete(x);
120         }
121
122         // Returns an iterator going forwards
123         public Iterator fwdIterator() {
124                 DLLNode pos = head;
125                 ArrayList l = new ArrayList(size());
126                 while (pos != null) {
127                         l.add(pos.data);
128                         pos = pos.next;
129                 }
130                 return l.iterator();
131         }
132
133         // Returns an interator going backwards
134         public Iterator backIterator() {
135                 DLLNode pos = tail;
136                 ArrayList l = new ArrayList(size());
137                 while (pos != null) {
138                         l.add(pos.data);
139                         pos = pos.prev;
140                 }
141                 return l.iterator();
142         }
143
144         // Returns an iterator (forwards)
145         public Iterator iterator() {
146                 return fwdIterator();
147         }
148
149         // Clears the list (removes all objects)
150         public void clear() {
151                 head = null;
152                 tail = null;
153         }
154
155         // Gets the nth object (from front), zero indexed
156         public Object get(int n) throws NoSuchElementException {
157                 if (n < 0 || n >= size())
158                         throw new NoSuchElementException("Index out of range");
159                 DLLNode pos = head;
160                 for (int i = 0; i < n; i++)
161                         pos = pos.next;
162                 return pos.data;
163         }
164
165         ////////////////////////////////////////////////////////////
166         // Private Functions -  Private because nothing out of class
167         //                                              should have access to a node
```

```
168         ////////////////////////////////////////////////////////////////
169
170         // Finds a node in the list
171         private DLLNode whereAt(Object x) {
172                 DLLNode pos = head;
173                 while (pos!=null) {
174                         if (pos.data.equals(x))
175                                 return pos;
176                         pos = pos.next;
177                 }
178                 return null;
179         }
180
181         // Deletes a node, given a pointer
182         private void delete(DLLNode x) {
183                 if (x == null)
184                         throw new NoSuchElementException("Can't delete nonexistant
    node");
185                 x.next.prev=x.prev;
186                 x.prev.next=x.next;
187                 x = null;
188         }
189
190         // Swaps a node with the next one in the list
191         private void swapWithNext(DLLNode pos) {
192                 if (pos.prev != null)
193                         pos.prev.next = pos.next;
194                 if (pos.next.next != null)
195                         pos.next.next.prev = pos;
196                 pos.next.prev = pos.prev;
197                 DLLNode tmp = pos.next.next;
198                 pos.next.next = pos;
199                 pos.prev = pos.next;
200                 pos.next = tmp;
201                 tmp  = null;
202
203         }
204
205         ////////////////////////
206         // Node data structure
207         ////////////////////////
208
209         private class DLLNode implements Comparable {
210                 public Object data;             // Data stored
211                 public DLLNode prev;    // Points to previous node
212                 public DLLNode next;    // Points to next node
213
214                 // Default constructor
215                 public DLLNode() {
216                         data = null;
217                         next = null;
218                         prev = null;
219                 }
220
221                 // Standard constructor
222                 public DLLNode(Object x) {
```

```
223                         data = x;
224                         next = null;
225                         prev = null;
226                 }
227
228                 public int compareTo(Object o) {
229                         return ((Comparable)data).compareTo(o);
230                 }
231
232                 public int compareTo(DLLNode o) {
233                         return ((Comparable)data).compareTo(o.data);
234                 }
235         }
236 }
237
238 /*EOF*/
```

```
1  ////////////////////////////////////
2  //
3  // GetInput Class
4  //
5  // Various functions to input basic types
6  // from the user.
7  //
8  // Uses recursion for error checking to
9  // minimize code length.
10 //
11 // (C) 2004, 2005, 2006 Benjamin Seidenberg
12 //
13 ////////////////////////////////////
14
15 // Import only what we need from java.io
16 import java.io.BufferedReader;
17 import java.io.InputStreamReader;
18 import java.io.IOException;
19
20 public class GetInput {
21
22         // returns an integer from the keyboard
23         public static int getInt() {
24                 int x = 0;
25                 BufferedReader kb = new BufferedReader(new
   InputStreamReader(System.in));
26                 try {
27                         x=Integer.parseInt(kb.readLine());
28                 }
29                 catch (IOException e) {
30                         System.out.println("Error " + e + " in the getInt()
   function");
31                 }
32                 catch (NumberFormatException e) {
33                         System.out.println("\nWarning! That was not an Integer!");
34                         System.out.print("Enter an Integer: ");
35                         x=getInt();
36                 }
37                 return x;
38         }
39
40         // returns a charecter from the keyboard
41         public static char getChar() {
42                 String tmp;
43                 BufferedReader kb = new BufferedReader(new
   InputStreamReader(System.in));
44                 try {
45                         tmp=kb.readLine();
46                 }
47                 catch (IOException e) {
48                         System.out.println("Error " + e + " in the getInt()
   function");
49                         return 0;
50                 }
51                 try {return tmp.charAt(0);}
52                 catch(IndexOutOfBoundsException e) {
```

```
53                          System.out.println("\nWarning! Nothing was inputted!");
54                          System.out.print("Enter a character: ");
55                          return getChar();
56                  }
57          }
58
59          // returns an double from the keyboard
60          public static double getDouble() {
61                  double x = 0;
62                  BufferedReader kb = new BufferedReader(new
    InputStreamReader(System.in));
63                  try {
64                          x=Double.parseDouble(kb.readLine());
65                  }
66                  catch (IOException e) {
67                          System.out.println("Error " + e + " in the getDouble()
    function");
68                  }
69                  catch (NumberFormatException e) {
70                          System.out.println("\nWarning! That was not an Double!");
71                          System.out.print("Enter a Double: ");
72                          x=getDouble();
73                  }
74                  return x;
75          }
76
77          // returns an string from the keyboard
78          public static String getString() {
79                  String x = "";
80                  BufferedReader kb = new BufferedReader(new
    InputStreamReader(System.in));
81                  try {
82                          x=kb.readLine();
83                  }
84                  catch (IOException e) {
85                          System.out.println("Error " + e + " in the getString()
    function");
86                  }
87                  return x;
88          }
89  }
```

```java
/*
 * Student.java
 *
 * Instance of a student
 * Closely related to StudentDB
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */

public class Student {
	private int idNum;                    // The student's ID number
	private String password;        // The student's password
	private int[][] scores;         // A 10x2 array of the student's scores
                                        // on the past 10 tests

	// Constructor
	public Student(int id, String pass) {
		idNum = id;
		password = pass;
		scores = new int[10][2];
	}

	// Full Constructor
	public Student(int id, String pass, int[][] grades) {
		idNum = id;
		password = pass;
		scores = grades;
	}

	// Returns Student ID
	public int getId() {
		return idNum;
	}

	// Returns Student password
	public String getPass() {
		return password;
	}

	// Returns all the scores
	public int[][] getScores() {
		return scores;
	}

	// Returns the score the student got on
	// the test 'testID' or -1 if that test
	// was not found
	public int scoreOn(int testID) {
		int i = 0;
		// This loop both increments and compares
		while (i < scores.length && scores[i][0] != testID)
			i++;
		// If i < scores.length, the item is in scores
		// If not, return - 1
```

```
 57                 return (i < scores.length) ? scores[i][1] : -1;
 58         }
 59
 60         // Adds a score to the student
 61         // If the student has a full record,
 62         // we will need to delete the oldest score and
 63         // move all scores down.
 64         public void addScore(int testNum, int score) {
 65                 // Find the first unused space, if any
 66                 int i = 0;
 67                 // This loop both increments and compares
 68                 while (i < scores.length && scores[i][0] !=0)
 69                         i++;
 70                 if (i < scores.length) { // we found an empty spot
 71                         // save the data
 72                         scores[i][0] = testNum;
 73                         scores[i][1] = score;
 74                 }
 75                 else {
 76                         // move everything down
 77                         for (int j = 0; j < scores.length - 1; j++) {
 78                                 scores[j][0] = scores[j+1][0];
 79                                 scores[j][1] = scores[j+1][1];
 80                         }
 81                         // put new test in the last position
 82                         scores[scores.length-1][0] = testNum;
 83                         scores[scores.length-1][1] = score;
 84                 }
 85         }
 86
 87         // Replaces a score for the student
 88         // This is because we set an inital score of zero
 89         public void replaceScore(int testNum, int score) throws
    TestNotFoundException {
 90                 for (int i = 0; i < scores.length; i++) {
 91                         if (scores[i][0] == testNum) { // Is it the right score?
 92                                 scores[i][1] =  score; // Set new score
 93                                 return; // end
 94                         }
 95                 }
 96                 // Score not found
 97                 throw new TestNotFoundException("Test " + testNum + " not found!");
 98         }
 99
100         // Shows the grades for a student
101         public void showGrades() {
102                 System.out.println("Test ID#:\tGrade:");
103                 for (int i = 0; i < scores.length; i++) {
104                         System.out.println(scores[i][0] + "\t\t" + scores[i][1]);
105                 }
106         }
107 }
108
109 class TestNotFoundException extends Exception {
110         public TestNotFoundException(String s) {
111                 super(s);
```

```
112         }
113 }
114
115 /* EOF */
```

```java
/*
 * StudentDB.java
 *
 * This is the database of students.
 * It uses in place modification of a
 * random access file.
 *
 * For functions refering to a specific place
 * in the file, they are ordered like an array.
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */

import java.io.RandomAccessFile;
import java.io.IOException;

public class StudentDB {
        // Size constants
        public static final int IDNUM_BYTES = 4;        // ID Number
        public static final int PASSWD_BYTES = 36;      // Password
        public static final int TESTID_BYTES = 4;       // Test ID (in array)
        public static final int TESTSCORE_BYTES = 4;    // Test score (in array)
        public static final int NUM_TESTS = 10          // Number of tests

        public static final int STUDENT_SIZE = IDNUM_BYTES + PASSWD_BYTES
//total size
                                        + ((TESTID_BYTES + TESTSCORE_BYTES) *
NUM_TESTS);


        // Member variables
        private String fname;

        // Constructor
        public StudentDB(String filename) {
                fname = filename;
        }

        // How many students are in the file
        public int count() throws IOException {
                RandomAccessFile fb = new RandomAccessFile(fname, "r");
                return (int)(fb.length() / STUDENT_SIZE);
        }

        // Reads a student from the nth position in the file
        public Student read(int n) throws IOException {
                // Open the file buffer
                RandomAccessFile fb = new RandomAccessFile(fname, "r");
                // Go to right place in file
                fb.seek(n * STUDENT_SIZE);
                int id = fb.readInt();
                byte[] ba = new byte[36]; // new byte array for reading
                fb.readFully(ba); // read in password
                String pass = new String(ba).trim();
```

```
55                  // Read in grades
56                  int[][] grades = new int[NUM_TESTS][2];
57                  for (int i = 0; i < NUM_TESTS; i++) {
58                          grades[i][0] = fb.readInt(); // Test ID
59                          grades[i][1] = fb.readInt(); // Score
60                  }
61                  fb.close();
62                  return new Student(id, pass, grades);
63          }
64
65          // Same as above, but with an already open file descriptor
66          public Student read(int n, RandomAccessFile fb) throws IOException {
67                  fb.seek(n * STUDENT_SIZE);
68                  int id = fb.readInt();
69                  byte[] ba = new byte[36]; // new byte array for reading
70                  fb.readFully(ba); // read in password
71                  String pass = new String(ba).trim();
72                  // Read in grades
73                  int[][] grades = new int[NUM_TESTS][2];
74                  for (int i = 0; i < NUM_TESTS; i++) {
75                          grades[i][0] = fb.readInt(); // Test ID
76                          grades[i][1] = fb.readInt(); // Score
77                  }
78                  return new Student(id, pass, grades);
79          }
80
81          // Writes a student to the nth position in the file
82          public void write(int n, Student s) throws IOException {
83                  // Open file buffer
84                  RandomAccessFile fb = new RandomAccessFile(fname, "rw");
85                  // Seek to right position
86                  fb.seek(n * STUDENT_SIZE);
87                  // Write out data
88                  fb.writeInt(s.getId()); // Student ID
89                  // Write out student pass
90                  StringBuffer sb = new StringBuffer (s.getPass());
91                  sb.setLength(36);
92                  fb.writeBytes(sb.toString());
93                  // Write out grades
94                  int[][] grades = s.getScores();
95                  for (int i = 0; i < grades.length; i++) {
96                          fb.writeInt(grades[i][0]); // Test ID
97                          fb.writeInt(grades[i][1]); // Score
98                  }
99                  fb.close();
100         }
101
102         // Same as above, but with an already open file descriptor
103         public void write(int n, Student s, RandomAccessFile fb) throws IOException
    {
104                 // Seek to right position
105                 fb.seek(n * STUDENT_SIZE);
106                 // Write out data
107                 fb.writeInt(s.getId()); // Student ID
108                 // Write out student pass
109                 StringBuffer sb = new StringBuffer (s.getPass());
```

```
110                 sb.setLength(36);
111                 fb.writeBytes(sb.toString());
112                 // Write out grades
113                 int[][] grades = s.getScores();
114                 for (int i = 0; i < grades.length; i++) {
115                         fb.writeInt(grades[i][0]); // Test ID
116                         fb.writeInt(grades[i][1]); // Score
117                 }
118         }
119
120         // Overwrites a student by finding the matching record with
121         // the same student ID
122         public void save(Student s) throws StudentNotFoundException, IOException {
123                 int n = findByID(s.getId());
124                 if (n == -1)
125                         throw new StudentNotFoundException("Couldn't find student
    number " + s.getId());
126                 else write(n, s);
127         }
128
129         // Find a student based on their student id
130         // returns the integer position in file or -1 if not found
131         public int findByID(int id) throws IOException {
132                 RandomAccessFile fb = new RandomAccessFile(fname, "r");
133                 for (long i = 0; i < (fb.length() / STUDENT_SIZE); i++)
134                         if (read((int)i).getId() == id)
135                                 return (int)i;
136                 return -1;
137         }
138
139         // Adds a new student to the file
140         public void addNewStudent(Student s) throws IOException  {
141                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
142                 fb.seek(fb.length()); // go to end of file
143                                 // Write out data
144                 fb.writeInt(s.getId()); // Student ID
145                 // Write out student pass
146                 StringBuffer sb = new StringBuffer (s.getPass());
147                 sb.setLength(36);
148                 fb.writeBytes(sb.toString());
149                 // Write out grades
150                 int[][] grades = s.getScores();
151                 for (int i = 0; i < grades.length; i++) {
152                         fb.writeInt(grades[i][0]); // Test ID
153                         fb.writeInt(grades[i][1]); // Score
154                 }
155                 fb.close();
156         }
157
158         // Shows all the students in a file
159         // Will display in the format "n.) <ID>" where n is the index + 1
160         public void showAll() throws IOException {
161                 RandomAccessFile fb = new RandomAccessFile(fname, "r");
162                 for (int i = 0; i < count(); i++) {
163                         System.out.println((i+1) + ".) " + read(i,fb).getId());
164                 }
```

```
165         }
166
167         // Deletes a student by index
168         // Since the student file is unordered, we will move
169         // the last item into the empty space.
170         // This will cause less I/O  usage and make the operation MUCH faster
171         // ( O(1) rather than O(n))
172         public void delete(int n) throws IOException {
173                 // Open File Buffer
174                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
175                 // Get count of items
176                 int count = (int)(fb.length() / STUDENT_SIZE);
177                 // Get the last item
178                 Student std = read(count - 1, fb);
179                 // Replace nth item with it
180                 write(n, std, fb);
181                 // Truncate file
182                 fb.setLength((count - 1) * STUDENT_SIZE);
183         }
184
185         // Deletes a student in the file (wrapper)
186         public void delete(Student s) throws IOException, StudentNotFoundException {
187                 int index = findByID(s.getId());
188                 if (index == -1)
189                         throw new StudentNotFoundException("Student number "+
    s.getId() +" not found!");
190                 delete(index);
191         }
192  }
193
194 class StudentNotFoundException extends Exception {
195         public StudentNotFoundException(String s) {
196                 super(s);
197         }
198 }
199
200 /* EOF */
```

```java
/*
 * Teacher.java
 *
 * Instance of a teacher
 * Closely related to TeacherDB
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */

public class Teacher {
        private int idNum;
        private String password;

        // Constructor
        public Teacher(int id, String pass) {
                idNum = id;
                password = pass;
        }

        // Get Password
        public String getPass() {
                return password;
        }

        // Set the password
        public void setPass(String pass) {
                password = pass;
        }

        // Get ID number
        public int getId() {
                return idNum;
        }
}

/* EOF */
```

```java
/*
 * TeacherDB.java
 *
 * This is the database of teachers
 * It uses in place modification of
 * random access file
 *
 * For functions refering to a specific place
 * in the file, they are ordered like an array
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */


import java.io.RandomAccessFile;
import java.io.IOException;

public class TeacherDB {
        // Data file sizes
        public static final int TEACHERID_BYTES = 4;
        public static final int TEACHPASS_BYTES = 36;
        public static final int TEACHER_SIZE = TEACHERID_BYTES + TEACHPASS_BYTES;

        // Member variables
        private String fname;

        // Constructor
        public TeacherDB(String filename) {
                fname = filename;
        }

        // Finds a teacher by ID
        // Returns a teacher if found, null if not
        public Teacher findTeacher(int id) throws IOException {
                // Open file buffer
                RandomAccessFile fb = new RandomAccessFile(fname, "r");
                int numItems = (int)(fb.length()/TEACHER_SIZE); // How many items
                for (int i = 0; i < numItems; i++) { // iterate through
                        fb.seek(i * TEACHER_SIZE); // go to beggining of the record
                        if (fb.readInt() == id) { // the id is first
                                byte[] ba = new byte[TEACHPASS_BYTES];
                                fb.readFully(ba);
                                fb.close();
                                return new Teacher (id, new String(ba).trim());
                        }
                }
                fb.close();
                return null; // item not found
        }

        // Adds a teacher to the database
        // It is the responsibility of the calling function
        // to ensure that the information is correct
        public void addTeacher(int id, String pass) throws IOException {
```

```
 57                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
 58                 fb.seek(fb.length()); // Go to end of file
 59                 fb.writeInt(id);
 60                 StringBuffer sb = new StringBuffer(pass);
 61                 sb.setLength(TEACHPASS_BYTES);
 62                 fb.writeBytes(sb.toString());
 63                 fb.close();
 64         }
 65
 66
 67         // Updates a teacher
 68         // Passed in a teacher, resets the password for
 69         // that teacher in file
 70         // Throws NoSuchTeacherException if not found
 71         public void changePass(Teacher t) throws NoSuchTeacherException,IOException
    {
 72                 // Open file buffer
 73                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
 74                 // Look for teacher
 75                 int numItems = (int)(fb.length()/TEACHER_SIZE); // How many items
 76                 for (int i = 0; i < numItems; i++) { // iterate through
 77                         fb.seek(i * TEACHER_SIZE); // go to beggining of the record
 78                         if (fb.readInt() == t.getId()) { // the id is first
 79                                 StringBuffer sb = new StringBuffer(t.getPass());
 80                                 sb.setLength(TEACHPASS_BYTES);
 81                                 fb.writeBytes(sb.toString());
 82                                 fb.close();
 83                                 return;
 84                         }
 85                 }
 86                 fb.close();
 87                 throw new NoSuchTeacherException("Teacher number" + t.getId() + "
    not found!");
 88         }
 89
 90         // Deletes a teacher that is passed in
 91         // Since the teacher file is unordered, we will move
 92         // the last item into the empty space.
 93         // This will cause less I/O  usage and make the operation MUCH faster
 94         // (O(1) rather than O(n))
 95         public void delete(int teacher) throws NoSuchTeacherException,IOException {
 96                 // Open file buffer
 97                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
 98                 // Look for teacher
 99                 int numItems = (int)(fb.length()/TEACHER_SIZE); // How many items
100                 for (int i = 0; i < numItems; i++) { // iterate through
101                         fb.seek(i * TEACHER_SIZE); // go to beggining of the record
102                         if (fb.readInt() == teacher) { // the id is first
103                                 fb.seek((numItems - 1) * TEACHER_SIZE); // Go to
    last
104                                 int newId = fb.readInt(); // read id
105                                 byte[] ba = new byte[TEACHPASS_BYTES];
106                                 fb.readFully(ba); // read pass
107                                 fb.seek(i * TEACHER_SIZE); // go back to where we
    were
108                                 fb.writeInt(newId); //write id
```

```
109                                 fb.write(ba); // write pass
110                                 fb.setLength ((numItems -1) * TEACHER_SIZE); //
    truncate file
111                                 return; // done
112                         }
113                 }
114                 fb.close();
115                 throw new NoSuchTeacherException("Teacher number" + teacher + " not
    found!");
116         }
117 }
118
119 class NoSuchTeacherException extends Exception {
120         public NoSuchTeacherException(String s) {
121                 super(s);
122         }
123 }
124
125 /* EOF */
```

```java
/*
 * Test.java
 *
 * Instance of a test, IO code to read/write it from file,
 * and code to create, modify and take a test.
 *
 * (C) Benjamin Seidenberg, 2006
 * For use with my IB Computer Science Dossier
 *
 */

import java.io.RandomAccessFile;
import java.io.IOException;
import java.util.Iterator;

public class Test {

        ////////////////////////////////////////
        // Defines data file sizes -- see diagram
        ////////////////////////////////////////

        // Header parts
        public static final int NUM_QUESTION_BYTES = 4;
        public static final int TEST_ID_BYTES = 4;
        public static final int SHOW_GRADE_BYTES = 1;
        public static final int SHOW_ANSWER_BYTES = 1;
        public static final int TEST_PASS_BYTES = 36;
        public static final int HEADER_BYTES = NUM_QUESTION_BYTES
                                        + TEST_ID_BYTES + SHOW_GRADE_BYTES
                                        + SHOW_ANSWER_BYTES + TEST_PASS_BYTES;

        // Question parts
        public static final int QTEXT_BYTES = 200;
        public static final int ANS_BYTES = 1;
        public static final int CHOICE_BYTES = 100;

        public static final int QUESTION_BYTES = QTEXT_BYTES + ANS_BYTES +
(CHOICE_BYTES * 4);

        ///////////////////////
        // End data file sizes
        ///////////////////////

        /////////////////
        // Class members
        /////////////////

        private DoubleLinkedList questions;     // List of questions in the test
        private int numQuestions;               // Number of questions on the tests
        private boolean showGrade;              // Whether or not to show students
their grade
        private boolean showAnswers;            // Whether or not to show students
the correct answers
        private int testID;                     // Unique test identification number
        private String testPass;                // Password to access test

```

```
54          ///////////////
55          // End Members
56          ///////////////
57
58          ////////////////////
59          // Constructors
60          ////////////////////
61
62          // File Constructor
63          // Takes filename to read out of
64          // We throw IOException so it can be handled higher up in the code
65          public Test(String fname) throws IOException {
66                  readFrom(fname);
67          }
68
69          // Default constructor
70          public Test(int testid) {
71                  questions = new DoubleLinkedList();
72                  testID = testid;
73                  showGrade = false;
74                  showAnswers = false;
75                  testPass = "";
76          }
77
78          ////////////////////
79          // End Constructors
80          ////////////////////
81
82          ////////////////////////////////////////////////////////
83          // Accessors - Descriptions are self-evident from names
84          ////////////////////////////////////////////////////////
85
86          public int getTestId() {
87                  return testID;
88          }
89
90          public String getPasswd() {
91                  return testPass;
92          }
93
94          public int getNumQuestions() {
95                  return numQuestions;
96          }
97
98          public boolean getShowAnswers() {
99                  return showAnswers;
100         }
101
102         public boolean getShowGrade() {
103                 return showGrade;
104         }
105
106         ////////////////////
107         // End of accessors
108         ////////////////////
109
```

```
110         ////////////////////////////////////////////////////
111         // Modifiers - Descriptions are self-evident from names
112         ////////////////////////////////////////////////////
113
114         public void setShowGrade(boolean sg) {
115                 showGrade = sg;
116         }
117
118         public void setShowAnswers(boolean sa) {
119                 showAnswers = sa;
120         }
121
122         /////////////////
123         // End Modifiers
124         /////////////////
125
126         ////////////////////////////////
127         // Work Functions - Main section
128         ////////////////////////////////
129
130         // Take the test
131         // returns the score the student got
132         public int takeTest() {
133                 int right = 0;
134                 Iterator it = questions.iterator();
135                 while (it.hasNext())
136                         if (((Question)it.next()).askQuestion())
137                                 right += 1;
138                 int score =  (int)((float)right / (float)numQuestions * 100);
139                 System.out.println("Test Complete!");
140                 if (showGrade)
141                         System.out.println("You made a score of: " + score);
142                 return score;
143         }
144
145         // Adds a question to the test
146         public void addQuestion(Question q) {
147                 numQuestions+=1;
148                 questions.append(q);
149         }
150
151         // Creates a question from input
152         // For use by the teacher
153
154         public Question createQuestion() {
155                 // Prompt for information
156                 System.out.print("Enter the question (200 character max): ");
157                 String qtext = GetInput.getString();
158                 System.out.print("Enter answer choice A (100 character max): ");
159                 String a = GetInput.getString();
160                 System.out.print("Enter answer choice B (100 character max): ");
161                 String b = GetInput.getString();
162                 System.out.print("Enter answer choice C (100 character max): ");
163                 String c = GetInput.getString();
164                 System.out.print("Enter answer choice D (100 character max): ");
165                 String d = GetInput.getString();
```

```
166                 System.out.print("Enter the correct choice (a, b, c, or d): ");
167                 char ca = Character.toLowerCase(GetInput.getChar());
168                 // Validate correct answer
169                 while ( ca < 97 || ca > 100) { // make sure ca is in range a -d
170                         System.out.println("Error! Answer should be a, b, c, or
    d!");
171                         System.out.print("Enter the correct choice (a, b, c, or d):
    ");
172                         ca = Character.toLowerCase(GetInput.getChar());
173                 }
174                 // Create and return the question
175                 return new Question(qtext, a, b, c, d, ca, showAnswers, showGrade);
176         }
177
178         // Displays a question and all the answer choices.
179         // For use by the teacher
180         public static void displayQuestion(Question q) {
181                 System.out.println("Question: " + q.getQText());
182                 System.out.println("a.) " + q.getChoiceA());
183                 System.out.println("b.) " + q.getChoiceB());
184                 System.out.println("c.) " + q.getChoiceC());
185                 System.out.println("d.) " + q.getChoiceD());
186                 System.out.println("Correct Answer: " + q.getCorrectAnswer());
187         }
188
189         // Allows modifications to a question
190         // For use by the teacher
191         public static void editQuestion(Question q) {
192                 char choice = '\0';
193                 while (choice != 'Q') {
194                         // Show the question
195                         displayQuestion(q);
196                         // Prompt for modification
197                         System.out.println("Modify what?");
198                         System.out.println("\tQuestion (T)ext");
199                         System.out.println("\tAnswer (A)");
200                         System.out.println("\tAnswer (B)");
201                         System.out.println("\tAnswer (C)");
202                         System.out.println("\tAnswer (D)");
203                         System.out.println("\tThe (R)ight Answer");
204                         System.out.println("or (Q)uit editing");
205                         System.out.print("Enter Your Choice: ");
206                         choice = Character.toUpperCase(GetInput.getChar());
207                         while(choice != 'A' && choice != 'B' && choice != 'C'
208                                         && choice != 'D' && choice != 'R' && choice
    != 'T'
209                                         && choice != 'Q') { // All possible choices
210                                 System.out.println("Your choice was invalid, please
    retry.");
211                                 System.out.print("Enter Your Choice: ");
212                                 choice = Character.toUpperCase(GetInput.getChar());
213                         }
214                         // Actually edit
215                         switch(choice) {
216                                 case 'A':
217                                         System.out.println("Enter New Answer Choice
```

```
(Max 100 characters): ");
                                        System.out.print("> ");
                                        q.setChoiceA(GetInput.getString());
                                break;
                                case 'B':
                                        System.out.println("Enter New Answer Choice
(Max 100 characters): ");
                                        System.out.print("> ");
                                        q.setChoiceB(GetInput.getString());
                                break;
                                case 'C':
                                        System.out.println("Enter New Answer Choice
(Max 100 characters): ");
                                        System.out.print("> ");
                                        q.setChoiceC(GetInput.getString());
                                break;
                                case 'D':
                                        System.out.println("Enter New Answer Choice
(Max 100 characters): ");
                                        System.out.print("> ");
                                        q.setChoiceD(GetInput.getString());
                                break;
                                case 'T':
                                        System.out.println("Enter New Question Text
(Max 200 characters): ");
                                        System.out.print("> ");
                                        q.setQText(GetInput.getString());
                                break;
                                case 'R':
                                        System.out.print("Enter new answer (a, b, c,
or d): ");
                                        char ca =
Character.toLowerCase(GetInput.getChar());
                                        // Validate correct answer
                                        while ( ca < 97 || ca > 100) { // make sure
ca is in range a -d
                                                System.out.println("Error! Answer
should be a, b, c, or d!");
                                                System.out.print("Enter the correct
choice (a, b, c, or d): ");
                                                ca =
Character.toLowerCase(GetInput.getChar());
                                        }
                                        q.setCorrectAnswer(ca);
                                break;
                                // No need to do anything for 'Q', as the loop will
end
                        } // End Switch
                } // End loop
        }

        // Creates a test from scratch
        // For use by the teacher
        public String buildTest() {
                // Prompt for basic data
                // Test Password
```

```
262                 System.out.print("Enter test password (up to 36 characters): ");
263                 testPass=GetInput.getString(); // We don't need to check length,
264                                                // it will just be truncated.
265                 // Number of questions
266                 System.out.print("How many questions: ");
267                 int nq = GetInput.getInt();
268                 // Test options
269                 System.out.println("Are students allowed to see their grades?");
270                 System.out.print("Enter 1 for yes, 0 for no: ");
271                 int toBool = GetInput.getInt();
272                 while (toBool != 1 && toBool != 0) {
273                         System.out.println("Invalid answer.");
274                         System.out.print("Enter 1 for yes, 0 for no: ");
275                 }
276                 showGrade = (toBool == 1);
277                 System.out.println("Are students allowed to see the correct
    answers?");
278                 System.out.print("Enter 1 for yes, 0 for no: ");
279                 toBool = GetInput.getInt();
280                 while (toBool != 1 && toBool != 0) {
281                         System.out.println("Invalid answer.");
282                         System.out.print("Enter 1 for yes, 0 for no: ");
283                 }
284                 showAnswers = (toBool == 1);
285
286                 // Get the questions
287                 for (int i = 0; i < nq; i++) {
288                         addQuestion(createQuestion());
289                 }
290                 System.out.println("Is the test correct?");
291                 System.out.print("Enter 1 for yes, 0 for no: ");
292                 toBool = GetInput.getInt();
293                 while (toBool != 1 && toBool != 0) {
294                         System.out.println("Invalid answer.");
295                         System.out.print("Enter 1 for yes, 0 for no: ");
296                 }
297                 while (toBool == 0) {
298                         editTest(); // allow the user to make modifications
299                         System.out.println("Is the test correct?");
300                         System.out.print("Enter 1 for yes, 0 for no: ");
301                         toBool = GetInput.getInt();
302                         while (toBool != 1 && toBool != 0) {
303                                 System.out.println("Invalid answer.");
304                                 System.out.print("Enter 1 for yes, 0 for no: ");
305                         }
306                 }
307                 System.out.println("Enter filename to save test to");
308                 System.out.print("Filename: ");
309                 String fname = GetInput.getString();
310                 try {
311                         writeToFile(fname);
312                 }
313                 catch (IOException e) {
314                         System.out.println("Error writing file!");
315                         System.out.println("Error: " + e.getMessage());
316                 }
```

```
317                 return fname;
318         }
319
320         // Edits a test
321         // For use by the teacher
322         // Note - Saving the test MUST be handled by the calling method
323         public void editTest() {
324                 showTest();
325                 System.out.println("What do you want to edit? Type \"o\" for test
    options or \"q\" to edit a question");
326                 System.out.print("Edit : ");
327                 char ch = Character.toLowerCase(GetInput.getChar());
328                 while (ch != 'o' && ch != 'q') {
329                         System.out.println("That was not a valid response.");
330                         System.out.println("What do you want to edit? Type \"o\" for
    test options or \"q\" to edit a question");
331                         System.out.print("Edit : ");
332                         ch = Character.toLowerCase(GetInput.getChar());
333                 }
334                 if (ch == 'o') {
335                         // Test Option code here
336                         System.out.println("Test Details:");
337                         System.out.println("Test ID: " + testID);
338                         System.out.println("Test (p)assword: " + testPass);
339                         System.out.println("Show (a)nswers: " + ((showAnswers) ?
    "yes" : "no"));
340                         System.out.println("Show (g)rades: " + ((showGrade) ? "yes"
    : "no"));
341                         System.out.println("What do you want to change?  (Enter a,
    g, or p)");
342                         System.out.print("Enter your choice: ");
343                         ch = Character.toLowerCase(GetInput.getChar());
344                         while (ch != 'a' && ch != 'g' && ch != 'p'){
345                                 System.out.println("Error! That was not an
    option!");
346                                 System.out.print("What do you want to change?
    (Enter a, g, or p)");
347                                 ch = Character.toLowerCase(GetInput.getChar());
348                         }
349                         switch (ch) {
350                                 case 'a':
351                                         System.out.println("Are students allowed to
    see the correct answers?");
352                                         System.out.println("Enter 1 for yes, 0 for
    no: ");
353                                         int toBool = GetInput.getInt();
354                                         while (toBool != 1 && toBool != 0) {
355                                                 System.out.println("Invalid
    answer.");
356                                                 System.out.println("Enter 1 for yes,
    0 for no: ");
357                                         }
358                                         showAnswers = (toBool == 1);
359                                 break;
360                                 case 'g':
361                                         System.out.println("Are students allowed to
```

```
see their grades?");
362                                         System.out.println("Enter 1 for yes, 0 for
no: ");
363                                         toBool = GetInput.getInt();
364                                         while (toBool != 1 && toBool != 0) {
365                                                 System.out.println("Invalid
answer.");
366                                                 System.out.println("Enter 1 for yes,
0 for no: ");
367                                         }
368                                         showGrade = (toBool == 1);
369                                 break;
370                         }
371
                }
372                 else {
373                         // Pick and edit a question
374                         System.out.println("Which question do you want to edit?");
375                         System.out.print("Enter question number (1-" + numQuestions
+ "): ");
376                         int questionNum = GetInput.getInt();
377                         while (!(questionNum > 0 && questionNum <= numQuestions)) {
378                                 System.out.println("Invalid Question Number!");
379                                 System.out.print("Enter question number (1-" +
numQuestions + "): ");
380                                 questionNum = GetInput.getInt();
381                         }
382                         editQuestion((Question)questions.get(questionNum - 1)); //
Zero-indexed
383                 }
384         }
385
386         // Shows a test
387         // Outputs the test meta-data and the questions, but not their answers
388         public void showTest() {
389                 System.out.println("Test Details:");
390                 System.out.println("Test ID: " + testID);
391                 System.out.println("Test (p)assword: " + testPass);
392                 System.out.println("Show (a)nswers: " + ((showAnswers) ? "yes" :
"no"));
393                 System.out.println("Show (g)rades: " + ((showGrade) ? "yes" :
"no"));
394                 int i = 1;
395                 Iterator it = questions.iterator();
396                 while (it.hasNext()) {
397                         System.out.println(i++ + ".) " +
((Question)it.next()).getQText());
398                 }
399         }
400
401
402         // End Work Functions
403
404         ////////////////
405         // IO Functions
406         ////////////////
```

```
407
408         // Reads out of a file into the test
409         // We throw IOException so it can be handled higher up in the code
410         public void readFrom(String fname) throws IOException {
411                 RandomAccessFile fb = new RandomAccessFile(fname, "r");
412                 numQuestions = 0; // this will be set by addQuestion
413                 questions = new DoubleLinkedList();
414                 int nq = fb.readInt(); // number of questions in the file
415                 testID = fb.readInt();
416                 showGrade = fb.readBoolean();
417                 showAnswers = fb.readBoolean();
418                 byte[] ba = new byte[TEST_PASS_BYTES]; // byte array to read into
419                 fb.readFully(ba); // read test pass
420                 testPass = new String(ba).trim(); // turn into string
421                 for (int i = 1; i <= nq; i++) {
422                         addQuestion(readQuestion(fb));
423                 }
424                 fb.close();
425         }
426
427
428         // Reads a question from the passed RandomAccessFile
429         // This assumes fb is open and at right offset
430         // We throw IOException so it can be handled higher up in the code
431         public Question readQuestion(RandomAccessFile fb) throws IOException {
432                 byte[] ba = new byte[QTEXT_BYTES]; // byte array to read into
433                 fb.readFully(ba); // Read in the question
434                 String qt = new String(ba).trim(); // The question
435                 ba = new byte[CHOICE_BYTES];
436                 char ans = (char) fb.read(); // Reads in one byte, the correct
    answer
437                 fb.readFully(ba); // read in choice a
438                 String a = new String(ba).trim();
439                 fb.readFully(ba); // read in choice b
440                 String b = new String(ba).trim();
441                 fb.readFully(ba); // read in choice c
442                 String c = new String(ba).trim();
443                 fb.readFully(ba); // read in choice d
444                 String d = new String(ba).trim();
445                 return new Question(qt, a, b, c, d, ans, showGrade, showAnswers);
446         }
447
448         // Writes out of the test into a file
449         // We throw IOException so it can be handled higher up in the code
450         public void writeToFile(String fname) throws IOException {
451                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
452                 // Make sure they want to overwrite the file. Throw exception if not
453                 if (new java.io.File(fname).exists()) {
454                         System.out.print("Warning. Writing to file " + fname + "
    will overwrite it. Continue? (Y/N): ");
455                         if(Character.toLowerCase(GetInput.getChar()) != 'y')
456                                 throw new IOException("User aborted");
457                 }
458                 fb.setLength(0); // Kill any existing data
459                 fb.writeInt(numQuestions); // Write the number of questions
460                 fb.writeInt(testID); // Write the testID
```

```
461                 fb.writeBoolean(showGrade); // Write showGrade
462                 fb.writeBoolean(showAnswers); // Write showAnswers
463                 StringBuffer sb; // Scratch StringBuffer to fix the length of the
    strings
464                 sb = new StringBuffer(testPass);
465                 sb.setLength(TEST_PASS_BYTES); // Make the proper size
466                 fb.writeBytes(sb.toString()); // Write to file
467                 Iterator it = questions.iterator();
468                 while (it.hasNext()) // Iterate through the questions
469                         writeQuestion(fb, (Question)it.next());
470                 fb.close();
471         }
472
473         // Writes a question to the passed RandomAccessFile
474         // This assumes fb is open and at right offset
475         // We throw IOException so it can be handled higher up in the code
476         public void writeQuestion(RandomAccessFile fb, Question q) throws
    IOException {
477                 StringBuffer sb;        // StringBuffer to manipulate the data
478                 byte[] ba;                      // Byte array
479                 // Get, set length and write out the question text
480                 sb = new StringBuffer(q.getQText());
481                 sb.setLength(QTEXT_BYTES);
482                 fb.writeBytes(sb.toString());
483
484                 // Write correct answer
485                 fb.write((byte)q.getCorrectAnswer());
486
487                 // Write choice A
488                 sb = new StringBuffer(q.getChoiceA());
489                 sb.setLength(CHOICE_BYTES);
490                 fb.writeBytes(sb.toString());
491
492                 // Write choice B
493                 sb = new StringBuffer(q.getChoiceB());
494                 sb.setLength(CHOICE_BYTES);
495                 fb.writeBytes(sb.toString());
496
497                 // Write choice C
498                 sb = new StringBuffer(q.getChoiceC());
499                 sb.setLength(CHOICE_BYTES);
500                 fb.writeBytes(sb.toString());
501
502                 // Write choice D
503                 sb = new StringBuffer(q.getChoiceD());
504                 sb.setLength(CHOICE_BYTES);
505                 fb.writeBytes(sb.toString());
506         }
507
508         // End IO Functions
509 }
510
511 /* EOF */
```

```java
/*
 * TestDB.java
 *
 * This is the database of tests
 * It uses in place modification of
 * random access file
 *
 * For functions refering to a specific place
 * in the file, they are ordered like an array
 *
 * (C) Benjamin Seidenberg 2006
 * For use with my IB Computer Science Dossier
 *
 */

import java.io.RandomAccessFile;
import java.io.IOException;

public class TestDB {
        // Data file sizes
        public static final int TESTID_BYTES = 4;
        public static final int TESTFNAME_BYTES = 256;
        public static final int TEST_SIZE = TESTID_BYTES + TESTFNAME_BYTES;

        // Member variables
        private String fname;

        // Constructor
        public TestDB(String filename) {
                fname = filename;
        }

        // Finds a test by ID
        // Returns a filename if found, "" if not
        public String findTest(int id) throws IOException {
                // Open file buffer
                RandomAccessFile fb = new RandomAccessFile(fname, "r");
                int numItems = (int)(fb.length()/TEST_SIZE); // How many items
                for (int i = 0; i < numItems; i++) { // iterate through
                        fb.seek(i * TEST_SIZE); // go to beggining of the record
                        if (fb.readInt() == id) { // the id is first
                                byte[] ba = new byte[TESTFNAME_BYTES];
                                fb.readFully(ba);
                                fb.close();
                                return new String(ba);
                        }
                }
                fb.close();
                return ""; // item not found
        }

        // Adds a test to the database
        // It is the responsibility of the calling function
        // to ensure that the information is correct
        public void addTest(int id, String filename) throws IOException {
                RandomAccessFile fb = new RandomAccessFile(fname, "rw");
```

```
 57                 fb.seek(fb.length()); // Go to end of file
 58                 fb.writeInt(id);
 59                 StringBuffer sb = new StringBuffer(filename);
 60                 sb.setLength(TESTFNAME_BYTES);
 61                 fb.writeBytes(sb.toString());
 62                 fb.close();
 63         }
 64
 65         // Shows all tests in the database
 66         public void showAll() throws IOException {
 67
 68                 // Open file buffer
 69                 RandomAccessFile fb = new RandomAccessFile(fname, "r");
 70                 int numItems = (int)(fb.length()/TEST_SIZE); // How many items
 71                 System.out.println("Test ID:\tFile Name:");
 72                 for (int i = 0; i < numItems; i++) { // iterate through
 73                         fb.seek(i * TEST_SIZE); // go to beggining of the record
 74                         System.out.print(fb.readInt()+ "\t\t"); // ID
 75                         byte[] ba = new byte[TESTFNAME_BYTES];
 76                         fb.readFully(ba);
 77                         System.out.println(new String(ba).trim());
 78                 }
 79                 fb.close();
 80         }
 81
 82         // Removes a test based on the id
 83         // Since the test file is unordered, we will move
 84         // the last item into the empty space.
 85         // This will cause less I/O  usage and make the operation MUCH faster
 86         // ( O(1) rather than O(n))
 87         public void removeTest(int testId) throws NoSuchTestException, IOException {
 88                 // Open file buffer
 89                 RandomAccessFile fb = new RandomAccessFile(fname, "rw");
 90                 int numItems = (int)(fb.length()/TEST_SIZE); // How many items
 91                 for (int i = 0; i < numItems; i++) { // iterate through
 92                         fb.seek(i * TEST_SIZE); // go to beggining of the record
 93                         if (fb.readInt() == testId) { // the id is first
 94                                 fb.seek((numItems-1) * TEST_SIZE); // got to last
    item
 95                                 int newId = fb.readInt();
 96                                 byte[] ba = new byte[TESTFNAME_BYTES];
 97                                 fb.readFully(ba); // read in file name
 98                                 fb.seek(i * TEST_SIZE);
 99                                 fb.writeInt(newId); // overwrite id
100                                 fb.write(ba); // overwrite filename
101                                 fb.setLength((numItems-1) * TEST_SIZE); // truncate
    file
102                                 return;
103                         }
104                 }
105                 fb.close();
106                 throw new NoSuchTestException("Test number " + testId + " not
    found"); // item not found
107         }
108 }
109
```

```
110 class NoSuchTestException extends Exception {
111         public NoSuchTestException(String s) {
112                 super(s);
113         }
114 }
115
116 /* EOF */
```