Benjamin Seidenberg
Candidate Number: 001279-023

IB Computer Science Dossier

Table Of Contents:

Introduction: ................................................................. 2

Stage A: Analysis .......................................................... 2
  Criteria for Success: ....................................................... 5
  Goals: ............................................................................. 5
  Requirements: .............................................................. 5
  Design Prototype: .......................................................... 7
  Program Outline: ........................................................... 7

Stage B: Detailed Design ................................................. 8
  Data Types: ........................................................................ 8
    External: ........................................................................ 8
    Internal: ......................................................................... 12
  Algorithms: The program utilizes two main types of structures that take an algorithmic approach: a doubly linked list and random access files. .............................................. 14
    Linked List: ...................................................................... 14
    Random Access Files: ..................................................... 16

Stage C: The Program ...................................................... 21
  Usability: ......................................................................... 21
  Error Handling: .............................................................. 22
    User Errors: .................................................................... 22
    System Errors: .............................................................. 23
  Evidence of Success: ....................................................... 25

Stage D: Documentation ............................................... 26
  Annotated Hardcopy of Output: ......................................... 26
  Evaluation of Solution: ................................................... 37
  There are two documents, one for teachers and one for students. ........................................... 39
    Teacher ........................................................................... 39
    Student: ......................................................................... 44

Appendices: Source Code .................................................
  Each file is individually numbered
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:
  o 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.
  o 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 straightforward 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:
  o Create and manage tests
  o Manage students
  o 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
  o 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.
  o 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

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>36 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>int idNum</td>
<td>byte[] password</td>
</tr>
</tbody>
</table>

(Casted back to string later, for 36 character password)

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>test #1 id</td>
<td>test #2 id</td>
<td>test #3 id</td>
<td>test #4 id</td>
<td>test #5 id</td>
<td>test #6 id</td>
<td>test #7 id</td>
<td>test #8 id</td>
<td>test #9 id</td>
</tr>
<tr>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
<td>score</td>
</tr>
</tbody>
</table>

## Diagram 3: Teacher Data Record

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>36 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>int idNum</td>
<td>byte[] password</td>
</tr>
</tbody>
</table>

(Casted back to char later, for 36 character password)
Diagram 4: Question Data Record

<table>
<thead>
<tr>
<th>200 bytes</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] qtext</td>
<td>char correctAnswer</td>
</tr>
<tr>
<td>(Casted back to String later, for 200 character question)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100 bytes</th>
<th>100 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] choiceA</td>
<td>byte[] choiceB</td>
</tr>
<tr>
<td>(Casted back to String later, for 100 character answer)</td>
<td>(Casted back to String later, for 100 character answer)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>100 bytes</th>
<th>100 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte[] choiceC</td>
<td>byte[] choiceD</td>
</tr>
<tr>
<td>(Casted back to String later, for 100 character answer)</td>
<td>(Casted back to String later, for 100 character answer)</td>
</tr>
</tbody>
</table>
Diagram 5: Test File

<table>
<thead>
<tr>
<th>4 Bytes</th>
<th>4 Bytes</th>
<th>1 byte</th>
<th>1 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>int</td>
<td>boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>numQuestions</td>
<td>testId</td>
<td>showGrade</td>
<td>showAnswers</td>
</tr>
</tbody>
</table>

36 bytes

byte[] testPass

(Casted back to String later, for 36 character password)

601 * numQuestion bytes

Question[] questions

<table>
<thead>
<tr>
<th>601 bytes</th>
<th>601 bytes</th>
<th>601 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>question 1</td>
<td>question 2</td>
<td>question 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.....</td>
</tr>
</tbody>
</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 and links are updated like such:
• **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`.

```java
// 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.

```java
private void delete(DLLNode x) {
    if (x == null)
        throw new NoSuchElementException("Can't delete nonexistent 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

```java
// 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.

```java
text
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 beginning 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[TEACHER_PASS_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

GetInput

Dossier

DoubleLinkedList

Node

Teacher

StudentDB

TestDB

TeacherDB

Student

Test

Question
## Stage C: The Program

### Usability:

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

```java
// 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("Warning! 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:

```java
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
        [...]
```


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:

```java
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 occurrence, 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:

```java
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:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The student is guided through the login and test taking process</td>
<td>See student login and test screenshots.</td>
</tr>
<tr>
<td>Teachers can add students.</td>
<td>See student management and add student screenshot.</td>
</tr>
<tr>
<td>Teachers can delete students.</td>
<td>See student management and delete student screenshots.</td>
</tr>
<tr>
<td>Teachers can view student grades.</td>
<td>See student management and add student screenshots.</td>
</tr>
<tr>
<td>Teachers can add teachers.</td>
<td>See teacher management and add teacher screenshots.</td>
</tr>
<tr>
<td>Teachers can delete teachers.</td>
<td>See teacher management and delete teacher screenshots.</td>
</tr>
<tr>
<td>Teachers can change their passwords.</td>
<td>See teacher management and change password screenshots.</td>
</tr>
<tr>
<td>Teachers can create new tests.</td>
<td>See test management and test creation screenshots.</td>
</tr>
<tr>
<td>Teachers can edit tests.</td>
<td>See test management and test editing screenshots.</td>
</tr>
<tr>
<td>Teachers can remove tests.</td>
<td>See test management and test deletion screenshots.</td>
</tr>
<tr>
<td>All prompts are explanatory, and explicitly tell the user what to do.</td>
<td>See all screenshots.</td>
</tr>
</tbody>
</table>
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@astroanut:/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' Declaration of Independence 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.

```plaintext
[... 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

```plaintext
[... From Adding Student...]
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 1
---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' Declaration of Independence 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
0               0
1.) Add Student
2.) Show Student Grades
3.) Delete Student
4.) Return to Teacher Menu
Enter your choice: 2
```

**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.
[... 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' Declaration of Independence 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 Occurred.
The exact error follows:
  students.db (No such file or directory)
astronut@astronut:/home/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:

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

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 to 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
  - 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:

---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**

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**

<table>
<thead>
<tr>
<th>1.) Add Student</th>
<th>2.) Show Student Grades</th>
<th>3.) Delete Student</th>
<th>4.) Return to Teacher Menu</th>
</tr>
</thead>
</table>

Enter your choice: **2**

Which student do you want to see grades for?
Enter student number: **1**

<table>
<thead>
<tr>
<th>Test ID#</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>78</td>
<td>75</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**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**
```
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’ Declaration of Independence 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 receive 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' Declaration of Independence 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:
/*
 * 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 database
        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.print("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();
            }
        }
    }
}
break;
    case 3:
        System.exit(0);
        break;
    }
}

// Teacher Code
public static void teacherLogin() {
    // Login part 1 - Find teacher
    System.out.print("Enter your teacher ID number: ");
    int id = GetInput.getInt();
    int c = 1;
    Teacher teach = null;
    try {
        teach = teachers.findTeacher(id); // Find teacher
        while (teach == null && c++ <= 3) { // give teachers 3 chances, c incremented in loop conditional
            System.out.println("Teacher number " + id + " not found!");
            System.out.print("Enter your teacher ID number: ");
            id = GetInput.getInt();
            teach = teachers.findTeacher(id);
        }
    }
    catch (IOException e) {
        System.out.println("An I/O Error Occurred.");
        System.out.println("The exact error follows: ");
        System.out.println("\t" + e.getMessage());
        System.exit(1);
    }
    if (teach == null) { // Teacher not found, go back to main menu
        System.out.println("Teacher not found! Returning to main menu.");
        return;
    }
    // Login part 2 - Verify password
    System.out.print("Enter your password: ");
    String passwd = GetInput.getString();
    c = 1;
    while (!passwd.equals(teach.getPass()) && c++ <= 3) { // again, 3 tries, and c is incremented in loop conditional
        System.out.println("Password Incorrect: ");
        System.out.print("Enter your password: ");
        passwd = GetInput.getString();
    }
    if (!passwd.equals(teach.getPass()))) { // Teacher missed password three tries
        System.out.println("Login Failed! Returning to main menu.");
        return; // Go back to main menu
    }
    // First menu-loop
    int choice = 0;
    while (true) { // break loop with a return
        
    }
// Function to manage students.
// For use by the teacher
public static void manageStudents() {
    // Manage student loop
    while (true) { // break with return
        System.out.println("1.) Add Student");
        System.out.println("2.) Show Student Grades");
        System.out.println("3.) Delete Student");
        System.out.println("4.) Return to Teacher Menu");
        System.out.print("Enter your choice: ");
        int choice = GetInput.getInt();
        while (! (choice > 0 && choice < 5)) { // Bad input
            System.out.println("Invalid Response!");
            System.out.print("Enter your choice: ");
            choice = GetInput.getInt();
        }
        switch (choice) {
            case 1:
                // Add Student
                try {
                    System.out.println("Enter new student ID: ");
                    int stdID = GetInput.getInt();
                    while (students.findByID(stdID) !=

- 1) {  
  ID " + stdID + " already exists. Choose another";
  System.out.println("Enter new student ID: ");
  stdID = GetInput.getInt();
  }
  System.out.print("Enter student password: ");
  String pass = GetInput.getString();
  students.addNewStudent(new Student(stdID, pass));
  }  
  catch (IOException e) {
    System.out.println("An I/O Error Occurred.");
    System.out.println("The exact error follows: ");
    System.out.println("\t" + e.getMessage());
    System.exit(1);
    break;
  case 2:
    try {
      // Pick Student show all students, with the number <index> + 1
      System.out.println("Which student do you want to see grades for? ");
      System.out.print("Enter student number: ");
      int sindex = GetInput.getInt();
      while (!sindex >= 0 && sindex <= students.count()) {
        System.out.println("That's not a valid choice!");
      student number: ");
      } Student std = students.read(sindex - 1); // Offset for human readability
      System.out.println("Show Grades");
      std.showGrades();
      }  
      catch (IOException e) {
        System.out.println("An I/O Error Occurred.");
        System.out.println("The exact error follows: ");
        System.out.println("\t" + e.getMessage());
        System.exit(1);
        break;
      }
case 3:
    try {
        // Pick Student
        students.showAll(); // This will
        System.out.println("Which student do
        System.out.print("Enter student
        int sindex = GetInput.getInt();
        while !(sindex > 0 && sindex <=
        System.out.println("That's
        System.out.print("Enter
        sindex = GetInput.getInt();
        Student std = students.read(sindex
        // Delete student
        students.delete(std);
        } catch (StudentNotFoundException e) {
            System.out.println("A serious error
            System.out.println("The student
            System.exit(1);
        } catch (IOException e) {
            System.out.println("An I/O Error
            System.out.println("The exact error
            System.out.println("\t" +
            System.exit(1);
        }
    } break;
    case 4:
        return;

    }

    // Function to manage teachers
    // For use by the teacher
    public static void manageTeachers() {
        // Manage teachers loop
        while (true) { // break with return
            System.out.println("1.) Add Teacher");
            System.out.println("2.) Change Teacher Password");
            System.out.println("3.) Delete Teacher");
            System.out.println("4.) Return to Teacher Menu");
            System.out.print("Enter your choice: ");
int choice = GetInput.getInt();
while (!(choice > 0 && choice < 5)) { // Bad input
    System.out.println("Invalid Response!");
    System.out.print("Enter your choice: ");
    choice = GetInput.getInt();
}
switch (choice) {
    case 1:
        // Add teacher
        try {
            System.out.print("Enter new teacher ID: ");
            int teachID = GetInput.getInt();
            while (teachers.findTeacher(teachID) != null) {
                System.out.println("Teacher ID " + teachID + " already exists. Choose another");
                System.out.print("Enter new teacher ID: ");
                teachID = GetInput.getInt();
            }
            System.out.print("Enter teacher password: ");
            String pass = GetInput.getString();
            teachers.addTeacher(teachID, pass);
        }
        catch (IOException e) {
            System.out.println("An I/O Error Occurred.");
            System.out.println("The exact error follows: ");
            e.getMessage();
        }
        break;
    case 2:
        // Change a password
        try {
            System.out.print("Enter the ID of the teacher whose password you want to change: ");
            // We don't show IDs to discourage changing other teachers' passwords
            int teachID = GetInput.getInt();
            while (teachers.findTeacher(teachID) == null) {
                System.out.println("Invalid ID!");
                System.out.print("Enter the ID of the teacher whose password you want to change: ");
                teachID = GetInput.getInt();
            }
            teachers.findTeacher(teachID);
            System.out.print("Enter new password: ");
        }
t.setPass(getInput.getString());
teachers.changePass(t);
}

} catch (IOException e) {
    System.out.println("An I/O Error
    System.out.println("The exact error
    System.out.println("\t +
    System.exit(1);
}

} catch (NoSuchTeacherException e) { // should
    System.out.println("A serious error
    System.out.println("The teacher
    System.exit(1);

break;
    case 3: // Delete a teacher
        try {
            System.out.print("Enter the ID of
            // We don't show IDs to discourage
            int teachID = getInput.getInt();
            while (teachers.findTeacher(teachID)
                System.out.println("Invalid
                System.out.println("Enter the
                teachID = getInput.getInt();
            } teachers.delete(teachID);
        }

} catch (IOException e) {
    System.out.println("An I/O Error
    System.out.println("The exact error
    System.out.println("\t +
    System.exit(1);
}

} catch (NoSuchTeacherException e) { // should
    System.out.println("A serious error
    System.out.println("The teacher
    System.exit(1);

}
320    break;
321    case 4: // Return to teacher menu
322    return;
323    }
324    }
325    }
326    }
327    }
328    }
329    }
330    }
331    }
332    }
333    }
334    }
335    }
336    }
337    }
338    }
339    }
340    }
341    }
342    }
343    }
344    case 1:
345    // Add a test
346    try{
347        System.out.print("Enter the ID for
348        the new test: ");
349        int tid = GetInput.getInt();
350        while (!tests.findTest(tid).equals("")) { // exists
351            System.out.println("Test number " + tid + " already exists!");
352            System.out.print("Enter the ID for the new test: ");
353            tid = GetInput.getInt();
354        }
355        Test t = new Test(tid);
356        tests.addTest(tid,t.buildTest()); //
357        createTest() returns filename
358    } catch(IOException e) {
359        System.out.println("An I/O Error
360        Occurred.");
361        System.out.println("The exact error
362        follows: ");
363        e.getMessage();
364        System.exit(1);
365    }
366    break;
367    case 2:
368    // Show all tests
369    try {
370        tests.showAll();
```java
368 } catch (IOException e) {
369     System.out.println("An I/O Error
370     System.out.println("The exact error
371     System.out.println("\t" +
372     System.exit(1);
373 }
374 break;
375 case 3: // Edit a test
376     try {
377         System.out.print("Enter the ID for
378         int tid = GetInput.getInt();
379         while
380             System.out.println("Test
381             System.out.print("Enter the
382             tid = GetInput.getInt();
383         String fname = tests.findTest(tid);
384         Test toEdit = new Test(fname); //
385         toEdit.editTest();
386         toEdit.writeToFile(fname);
387     }
388 }
389 catch (IOException e) {
390     System.out.println("An I/O Error
391     System.out.println("The exact error
392     System.out.println("\t" +
393     System.exit(1);
394 }
395 break;
396 case 4: // Delete a test
397     try {
398         System.out.print("Enter the ID for
399         int tid = GetInput.getInt();
400         while
401             System.out.println("Test
402             System.out.print("Enter the
403             tid = GetInput.getInt();
404         String fname = tests.findTest(tid);
405         ")
406         System.out.println("\t" +
407         System.exit(1);
408         ")
```
tests.removeTest(tid); // remove
new java.io.File(fname).delete(); //
}
catch (IOException e) {
    System.out.println("An I/O Error
    
    ");
}
catch (NoSuchTestException e) { // Should
    System.out.println("A serious error
    
    ");
}
break;
case 5:
    return; //Return to menu
}
}

// Student Code
private static void studentLogin() {
    // Login  Part 1: Load student
    System.out.print("Enter your student ID number: ");
    int id = GetInput.getInt();
    int c = 1;
    Student std = null;
    try {
        int index = students.findIndex(id); // Find position in file
        while (index == -1 & & c++ <= 3) { // give students 3 chances,
c incremented in loop conditional
            System.out.println("Student number " + id + " not
            
            ");
    System.out.print("Enter your student ID number: ");
    id = GetInput.getInt();
    index = students.findIndex(id);
    }
    if (index == -1) { // Student not found, go back to main
        System.out.println("Student not found! Returning to
        
        ");
        return;
    }
    std = students.read(index);
}
catch (IOException e) {

System.out.println("An I/O Error Occured.");
System.out.println("Please contact your teacher for further assistance.");
System.out.println("The exact error follows: ");
System.out.println("\" + e.getMessage());
System.exit(1); // Exit with error code; we don't know the cause of the error and it's unsafe to continue
}
  // Login Part 2 - Check Password
  System.out.print("Enter your password: ");
  String passwd = GetInput.getString();
  c = 1;
  while (!passwd.equals(std.getPass()) && c++ <= 3) { // again, 3 tries, and c is incremented in loop conditional
    System.out.println("Password Incorrect: ");
    System.out.print("Enter your password: ");
    passwd = GetInput.getString();
  }
  if (!passwd.equals(std.getPass()))) { // Student missed password three tries
    System.out.println("Login Failed! Returning to main menu.");
    return; // Go back to main menu
  }

  // Pick test
  System.out.print("Enter the number of the test you wish to take: ");
  int testid = GetInput.getInt();
  c = 1;
  String tfname = "";
  try {
    tfname = tests.findTest(testid); // Find the filename of that test
    while(tfname.equals("") && c++ <= 3) { // again, 3 tries, and c is incremented in loop conditional
      System.out.println("Test number " + testid + " + not found!");
      System.out.print("Enter the number of the test you wish to take (given to you by your teacher): ");
      testid = GetInput.getInt();
      tfname = tests.findTest(testid); // Find the filename of that test
    }
  }
  catch (IOException e) {
    System.out.println("An I/O Error Occured.");
    System.out.println("Please contact your teacher for further assistance.");
    System.out.println("The exact error follows: ");
    System.out.println("\" + e.getMessage());
    System.exit(1); // Exit with error code; we don't know the cause of the error and it's unsafe to continue
    if (tfname.equals("")) { // Test not found, go back to main menu
      System.out.println("Test not found! Returning to main menu.");
      return;
    }
if (std.scoreOn(testid) != -1) {
    System.out.println("You have already taken this test!");
    return;
}

// Load the test
// Make sure to catch IO exception
    Test currentTest = null;
    try {
        currentTest = new Test(tfname);
    } catch (FileNotFoundException e) { // the most likely reason for failure is a missing test
        System.out.println("Error! The file " + tfname + " was not found.");
        System.out.println("Please contact your teacher for further assistance.");
        return; // Go back to main menu
    }

    catch (IOException e) { // Only one catch() will run, so this generically handles all other IO errors
        System.out.println("An I/O Error Occurred.");
        System.out.println("Please contact your teacher for further assistance.");
        System.out.println("The exact error follows: ");
        System.out.println("\t" + e.getMessage());
        System.exit(1); // Exit with error code; we don't know the cause of the error and it's unsafe to continue
    }

    // Verify the test password
    System.out.print("Enter the test password (given to you by your teacher): ");
    passwd = GetInput.getString();
    c = 1;
    while (!passwd.equals(currentTest.getPassword()) && c++ <= 3) { // Familiar loop
        System.out.println("Password Incorrect: ");
        System.out.print("Enter your password: ");
        passwd = GetInput.getString();
        if (!passwd.equals(currentTest.getPassword())) { // Student missed password three tries
            System.out.println("Test Authentication Failed! Returning to main menu.");
            return; // Go back to main menu
        }
    }

    // Set score of 0
    std.addScore(testid, 0);
    try {
        students.save(std);
    } catch (IOException e) {
System.out.println("An I/O Error Occured.");
System.out.println("Please contact your teacher for further assistance.");
System.out.println("The exact error follows: ");
System.out.println("\t" + e.getMessage());
System.exit(1); // Exit with error code; we don't know the cause of the error and it's unsafe to continue

catch (StudentNotFoundException e) { // This should never happen - Student was read out of database
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);
}

// Take test
System.out.println("Now starting test. Do not quit the program while in the middle of the test. If you do, you will receive a zero on the test. If this happens in error, please contact your teacher.");
int score = currentTest.takeTest();

// Set proper score
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);
}
try {
  students.save(std);
}
catch (IOException e) {
System.out.println("An I/O Error Occured.");
System.out.println("Please contact your teacher for further assistance.");
System.out.println("The exact error follows: ");
System.out.println("\t" + e.getMessage());
System.exit(1); // Exit with error code; we don't know the cause of the error and it's unsafe to continue
}
catch (StudentNotFoundException 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);
}

// Return to main menu
}
589 }
590
/*
 * Doubly Linked List
 * DoubleLinkedList Class
 * The implementation of a doubly linked
 * list of Objects. Provides most methods
 * needed.
 */

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

public class DoubleLinkedList {

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

    // 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) {

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;

// Dumps list to screen (Default is fwd)
public void dump() {
  dumpFwd();
}

// Dumps list to screen (From head to tail)
public void dumpFwd() {
  DLLNode pos = head;
  while(pos != null) {
    System.out.println(pos.data);
    pos = pos.next;
  }
}

// Dumps list to screen (From tail to head)
public void dumpBack() {
  DLLNode pos = tail;
  while(pos != null) {
    System.out.println(pos.data);
    pos = pos.prev;
  }
}

// Returns if the list is empty
public boolean isEmpty() {
  return (head == null || tail == null);
}

// Returns number of nodes in the list
public int size() {
  int count = 0;
  DLLNode pos = head;
  while (pos != null) {
    count++;
    pos = pos.next;
  }
  return count;
}

// Returns whether an object is in the list
public boolean isIn(Object x) {
  return (whereAt(x) != null);
// Deletes the node containing the passed in object
public void delete(Object o) throws NoSuchElementException {
    DLLNode x = whereAt(o);
    if (x == null)
        throw new NoSuchElementException("Item not found in the list");
    delete(x);
}

// Returns an iterator going forwards
public Iterator fwdIterator() {
    DLLNode pos = head;
    ArrayList l = new ArrayList(size());
    while (pos != null) {
        l.add(pos.data);
        pos = pos.next;
    }
    return l.iterator();
}

// Returns an iterator going backwards
public Iterator backIterator() {
    DLLNode pos = tail;
    ArrayList l = new ArrayList(size());
    while (pos != null) {
        l.add(pos.data);
        pos = pos.prev;
    }
    return l.iterator();
}

// Returns an iterator (forwards)
public Iterator iterator() {
    return fwdIterator();
}

// Clears the list (removes all objects)
public void clear() {
    head = null;
    tail = null;
}

// Gets the nth object (from front), zero indexed
public Object get(int n) throws NoSuchElementException {
    if (n < 0 || n >= size())
        throw new NoSuchElementException("Index out of range");
    DLLNode pos = head;
    for (int i = 0; i < n; i++)
        pos = pos.next;
    return pos.data;
}

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

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

// Swaps a node with the next one in the list
private void swapWithNext(D ListNode pos) {
    if (pos.prev != null)
        pos.prev.next = pos.next;
    if (pos.next.next != null)
        pos.next.next.prev = pos;
    pos.next.prev = pos.prev;
    D ListNode tmp = pos.next.next;
    pos.next.next = pos;
    pos.prev = pos.next;
    pos.next = tmp;
    tmp = null;
}

// Node data structure
private class D ListNode implements Comparable {
    public Object data; // Data stored
    public D ListNode prev; // Points to previous node
    public D ListNode next; // Points to next node

    // Default constructor
    public D ListNode() {
        data = null;
        next = null;
        prev = null;
    }

    // Standard constructor
    public D ListNode(Object x) {
data = x;
next = null;
prev = null;
}

public int compareTo(Object o) {
    return ((Comparable)data).compareTo(o);
}

public int compareTo(DLLNode o) {
    return ((Comparable)data).compareTo(o.data);
}
// GetInput Class
// Various functions to input basic types
// from the user.
// Uses recursion for error checking to
// minimize code length.
// (C) 2004, 2005, 2006 Benjamin Seidenberg

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class GetInput {

    // returns an integer from the keyboard
    public static int getInt() {
        int 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("Warning! That was not an Integer! ");
            System.out.println("Enter an Integer: ");
            x=getInt();
        } return x;
    }

    // returns a character from the keyboard
    public static char getChar() {
        String tmp;
        BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));
        try {
            tmp=kb.readLine();
        } catch (IOException e) {
            System.out.println("Error " + e + " in the getInt() function");
        } return 0;
    }

    try {return tmp.charAt(0);}
    catch(IndexOutOfBoundsException e) {
    }
System.out.println("\nWarning! Nothing was inputted!");
System.out.print("Enter a character: ");
return getChar();
}
}

// returns a double from the keyboard
public static double getDouble() {
    double x = 0;
    BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));
    try {
        x=Double.parseDouble(kb.readLine());
    } 
    catch (IOException e) {
        System.out.println("Error " + e + " in the getDouble() function");
    } 
    catch (NumberFormatException e) {
        System.out.println("\nWarning! That was not a Double!");
        System.out.print("Enter a Double: ");
        x=getDouble();
    } 
    return x;
}

// returns a string from the keyboard
public static String getString() {
    String x = "";
    BufferedReader kb = new BufferedReader(new InputStreamReader(System.in));
    try {
        x=kb.readLine();
    } 
    catch (IOException e) {
        System.out.println("Error " + e + " in the getString() function");
    } 
    return x;
}
/*
 * 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
    }
}
return (i < scores.length) ? scores[i][1] : -1;
}

// Adds a score to the student
// If the student has a full record,
// we will need to delete the oldest score and
// move all scores down.
public void addScore(int testNum, int score) {
    // Find the first unused space, if any
    int i = 0;
    // This loop both increments and compares
    while (i < scores.length && scores[i][0] != 0)
        i++;
    if (i < scores.length) { // we found an empty spot
        // save the data
        scores[i][0] = testNum;
        scores[i][1] = score;
    }
    else {
        // move everything down
        for (int j = 0; j < scores.length - 1; j++) {
            scores[j][0] = scores[j+1][0];
            scores[j][1] = scores[j+1][1];
        }
        // put new test in the last position
        scores[scores.length-1][0] = testNum;
        scores[scores.length-1][1] = score;
    }
}

// Replaces a score for the student
// This is because we set an initial score of zero
public void replaceScore(int testNum, int score) throws TestNotFoundException {
    for (int i = 0; i < scores.length; i++) {
        if (scores[i][0] == testNum) { // Is it the right score?
            scores[i][1] = score; // Set new score
            return; // end
        }
    }
    // Score not found
    throw new TestNotFoundException("Test " + testNum + " not found!");
}

// Shows the grades for a student
public void showGrades() {
    System.out.println("Test ID:\tGrade:");
    for (int i = 0; i < scores.length; i++) {
        System.out.println(scores[i][0] + "\t" + scores[i][1]);
    }
}

class TestNotFoundException extends Exception {
    public TestNotFoundException(String s) {
        super(s);
112     }
113   }
114
115  /* EOF */
import java.io.RandomAccessFile;
import java.io.IOException;

public class StudentDB {

    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
        + ((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 {
        RandomAccessFile fb = new RandomAccessFile(fname, "r");
        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();
// Read in grades
int[][] grades = new int[NUM_TESTS][2];
for (int i = 0; i < NUM_TESTS; i++) {
    grades[i][0] = fb.readInt(); // Test ID
    grades[i][1] = fb.readInt(); // Score
}
fb.close();
return new Student(id, pass, grades);

// Same as above, but with an already open file descriptor
public Student read(int n, RandomAccessFile fb) throws IOException {
    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();
    // Read in grades
    int[][] grades = new int[NUM_TESTS][2];
    for (int i = 0; i < NUM_TESTS; i++) {
        grades[i][0] = fb.readInt(); // Test ID
        grades[i][1] = fb.readInt(); // Score
    }
    return new Student(id, pass, grades);
}

// Writes a student to the nth position in the file
public void write(int n, Student s) throws IOException {
    // Open file buffer
    RandomAccessFile fb = new RandomAccessFile(fname, "rw");
    // Seek to right position
    fb.seek(n * STUDENT_SIZE);
    // Write out data
    fb.writeInt(s.getId()); // Student ID
    // Write out student pass
    StringBuffer sb = new StringBuffer(s.getPass());
    sb.setLength(36);
    fb.writeBytes(sb.toString());
    // Write out grades
    int[][] grades = s.getGrades();
    for (int i = 0; i < grades.length; i++) {
        fb.writeInt(grades[i][0]); // Test ID
        fb.writeInt(grades[i][1]); // Score
    }
    fb.close();
}

// Same as above, but with an already open file descriptor
public void write(int n, Student s, RandomAccessFile fb) throws IOException {
    // Seek to right position
    fb.seek(n * STUDENT_SIZE);
    // Write out data
    fb.writeInt(s.getId()); // Student ID
    // Write out student pass
    StringBuffer sb = new StringBuffer(s.getPass());
sb.setLength(36);
fb.writeBytes(sb.toString());
// Write out grades
int[][] grades = s.getScores();
for (int i = 0; i < grades.length; i++) {
    fb.writeInt(grades[i][0]); // Test ID
    fb.writeInt(grades[i][1]); // Score
}
// Overwrites a student by finding the matching record with
// the same student ID
public void save(Student s) throws StudentNotFoundException, IOException {
    int n = findById(s.getId());
    if (n == -1)
        throw new StudentNotFoundException("Couldn't find student
number " + s.getId());
    else write(n, s);
}
// Find a student based on their student id
// returns the integer position in file or -1 if not found
public int findById(int id) throws IOException {
    RandomAccessFile fb = new RandomAccessFile(fname, "r");
    for (long i = 0; i < (fb.length() / STUDENT_SIZE); i++)
        if (read((int)i).getId() == id)
            return (int)i;
    return -1;
}
// Adds a new student to the file
public void addNewStudent(Student s) throws IOException {
    RandomAccessFile fb = new RandomAccessFile(fname, "rw");
    fb.seek(fb.length()); // go to end of file
    // Write out data
    fb.writeInt(s.getId()); // Student ID
    fb.writeBytes(s.getPass()); // Write out student pass
    StringBuffer sb = new StringBuffer(s.getPass());
    sb.setLength(36);
    fb.writeBytes(sb.toString());
    // Write out grades
    int[][] grades = s.getScores();
    for (int i = 0; i < grades.length; i++) {
        fb.writeInt(grades[i][0]); // Test ID
        fb.writeInt(grades[i][1]); // Score
    }
    fb.close();
}
// Shows all the students in a file
// Will display in the format ".n.) <ID>" where n is the index + 1
public void showAll() throws IOException {
    RandomAccessFile fb = new RandomAccessFile(fname, "r");
    for (int i = 0; i < count(); i++) {
        System.out.println((i+1) + ".") + read(i, fb).getId();
    }
public void delete(int n) throws IOException {
    // Open File Buffer
    RandomAccessFile fb = new RandomAccessFile(fname, "rw");
    // Get count of items
    int count = (int)(fb.length() / STUDENT_SIZE);
    // Get the last item
    Student std = read(count - 1, fb);
    // Replace nth item with it
    write(n, std, fb);
    // Truncate file
    fb.setLength((count - 1) * STUDENT_SIZE);
}

// Deletes a student in the file (wrapper)
public void delete(Student s) throws IOException, StudentNotFoundException {
    int index = findById(s.getId());
    if (index == -1) {
        throw new StudentNotFoundException("Student number "+
            s.getId() + " not found!");
    }
    delete(index);
}

class StudentNotFoundException extends Exception {
    public StudentNotFoundException(String s) {
        super(s);
    }
}
/* EOF */
/*
 * 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 */
/ * 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.IlegalArgumentException;

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 beginning 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 {

}
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();
}

// Passed in a teacher, resets the password for
// that teacher in file
// Throws NoSuchTeacherException if not found
public void changePass(Teacher t) 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 begging of the record
        if (fb.readInt() == t.getId()) { // the id is first
            StringBuffer sb = new StringBuffer(t.getPass());
            sb.setLength(TEACHPASS_BYTES);
            fb.writeBytes(sb.toString());
            fb.close();
            return;
        }
    }
    fb.close();
    throw new NoSuchTeacherException("Teacher number" + t.getId() + " not found!");
}

// Deletes a teacher that is passed in
// Since the teacher file is unordered, we will move
// the last item into the empty space.
// This will cause less I/O usage and make the operation MUCH faster
// (O(1) rather than O(n))
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 beginning 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!");
} }

class NoSuchTeacherException extends Exception {
    public NoSuchTeacherException(String s) {
        super(s);
    }
}
/** EOF */
/*
 * 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
public Test(String fname) throws IOException {
    readFrom(fname);
}

// Default constructor
public Test(int testid) {
    questions = new DoubleLinkedList();
    testID = testid;
    showGrade = false;
    showAnswers = false;
    testPass = "";
}

public int getTestId() {
    return testID;
}

public String getPasswd() {
    return testPass;
}

public int getNumQuestions() {
    return numQuestions;
}

public boolean getShowAnswers() {
    return showAnswers;
}

public boolean getShowGrade() {
    return showGrade;
}
public void setShowGrade(boolean sg) {
    showGrade = sg;
}

public void setShowAnswers(boolean sa) {
    showAnswers = sa;
}

public int takeTest() {
    int right = 0;
    Iterator it = questions.iterator();
    while (it.hasNext())
        if (((Question)it.next()).askQuestion())
            right += 1;
    int score = (int)((float)right / (float)numQuestions * 100);
    System.out.println("Test Complete!");
    if (showGrade)
        System.out.println("You made a score of: " + score);
    return score;
}

// Adds a question to the test
public void addQuestion(Question q) {
    numQuestions+=1;
    questions.append(q);
}

// Creates a question from input
// For use by the teacher
public Question createQuestion() {
    // Prompt for information
    System.out.print("Enter the question (200 character max): ");
    String qtext = GetInput.getString();
    System.out.print("Enter answer choice A (100 character max): ");
    String a = GetInput.getString();
    System.out.print("Enter answer choice B (100 character max): ");
    String b = GetInput.getString();
    System.out.print("Enter answer choice C (100 character max): ");
    String c = GetInput.getString();
    System.out.print("Enter answer choice D (100 character max): ");
    String d = GetInput.getString();
System.out.println("Enter the correct choice (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.println("Enter the correct choice (a, b, c, or d): ");
    ca = Character.toLowerCase(GetInput.getChar());
} // Create and return the question
return new Question(qtext, a, b, c, d, ca, showAnswers, showGrade);

// Displays a question and all the answer choices.
// For use by the teacher
public static void displayQuestion(Question q) {
    System.out.println("Question: " + q.getQText());
    System.out.println("a. " + q.getChoiceA());
    System.out.println("b. " + q.getChoiceB());
    System.out.println("c. " + q.getChoiceC());
    System.out.println("d. " + q.getChoiceD());
    System.out.println("Correct Answer: " + q.getCorrectAnswer());
}

// Allows modifications to a question
// For use by the teacher
public static void editQuestion(Question q) {
    char choice = '\0';
    while (choice != 'Q') {
        // Show the question
displayQuestion(q);
        // Prompt for modification
        System.out.println("Modify what?");
        System.out.println("Question Text");
        System.out.println("Answer A");
        System.out.println("Answer B");
        System.out.println("Answer C");
        System.out.println("Answer D");
        System.out.println("The Right Answer");
        System.out.println("Quit editing");
        System.out.println("Enter Your Choice: ");
        choice = Character.toUpperCase(GetInput.getChar());
        while (choice != 'A' && choice != 'B' && choice != 'C' && choice != 'D' && choice != 'R' && choice != 'Q') { // All possible choices
            System.out.println("Your choice was invalid, please retry.");
        System.out.println("Enter Your Choice: ");
        choice = Character.toUpperCase(GetInput.getChar());
    } // Actually edit
    switch (choice) {
        case 'A':
System.out.println("Enter New Answer Choice");
    }
(Max 100 characters): ");
System.out.println("Enter New Answer Choice ");
q.setChoiceA(GetInput.getString());
break;
case 'B':
System.out.println("Enter New Answer Choice ");
q.setChoiceB(GetInput.getString());
break;
case 'C':
System.out.println("Enter New Answer Choice ");
q.setChoiceC(GetInput.getString());
break;
case 'D':
System.out.println("Enter New Answer Choice ");
q.setChoiceD(GetInput.getString());
break;
case 'T':
System.out.println("Enter New Question Text ");
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 ");
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
System.out.println("Enter test password (up to 36 characters): ");
testPass = GetInput.getString(); // We don't need to check length,
// it will just be truncated.

// Number of questions
System.out.println("How many questions: ");
int nq = GetInput.getInt();
// Test options
System.out.println("Are students allowed to see their grades?");
System.out.println("Enter 1 for yes, 0 for no: ");
int toBool = GetInput.getInt();
while (toBool != 1 && toBool != 0) {
    System.out.println("Invalid answer.");
    System.out.println("Enter 1 for yes, 0 for no: ");
}
showGrade = (toBool == 1);
System.out.println("Are students allowed to see the correct answers? ");
System.out.println("Enter 1 for yes, 0 for no: ");
toBool = GetInput.getInt();
while (toBool != 1 && toBool != 0) {
    System.out.println("Invalid answer.");
    System.out.println("Enter 1 for yes, 0 for no: ");
}
showAnswers = (toBool == 1);

// Get the questions
for (int i = 0; i < nq; i++) {
    addQuestion(createQuestion());
} System.out.println("Is the test correct?");
System.out.println("Enter 1 for yes, 0 for no: ");
toBool = GetInput.getInt();
while (toBool != 1 && toBool != 0) {
    System.out.println("Invalid answer.");
    System.out.println("Enter 1 for yes, 0 for no: ");
}
while (toBool == 0) {
    editTest(); // allow the user to make modifications
    System.out.println("Is the test correct?");
    System.out.println("Enter 1 for yes, 0 for no: ");
toBool = GetInput.getInt();
while (toBool != 1 && toBool != 0) {
    System.out.println("Invalid answer.");
    System.out.println("Enter 1 for yes, 0 for no: ");
}
}
System.out.println("Enter filename to save test to");
System.out.println("Filename: ");
String fname = GetInput.getString();
try {
    writeToFile(fname);
} catch (IOException e) {
    System.out.println("Error writing file!");
    System.out.println("Error: " + e.getMessage());
}
    return fname;

    // Edits a test
    // For use by the teacher
    // Note - Saving the test MUST be handled by the calling method
    public void editTest() {
        showTest();
        System.out.println("What do you want to edit? Type " + testID);
        System.out.print("Edit : ");
        System.out.println("Type \"o\" for test options or \"q\" to edit a question");
        char ch = Character.toLowerCase(GetInput.getChar());
        while (ch != 'o' && ch != 'q') {
            System.out.println("That was not a valid response.\n");
            System.out.println("What do you want to edit? Type \"o\" for
test options or \"q\" to edit a question");
            ch = Character.toLowerCase(GetInput.getChar());
        }
        if (ch == 'o') {
            // Test Option code here
            System.out.println("Test Details: ");
            System.out.println("Test ID: ");
            System.out.println("Test (p)assword: ");
            System.out.println("Show (a)swers: ");
            System.out.println("yes": "no")
            System.out.println("Show (g)rades: ");
            System.out.println("What do you want to change? (Enter a,
g, or p)\n");
            System.out.println("Enter your choice: ");
            ch = Character.toLowerCase(GetInput.getChar());
            while (ch != 'a' && ch != 'g' && ch != 'p') {
                System.out.println("Error! That was not an
                option!\n");
                System.out.println("What do you want to change?
Enter a, g, or p\n");
            }
        }
        switch (ch) {
            case 'a':
                System.out.println("Are students allowed to
                see the correct answers?\n");
                System.out.println("Enter 1 for yes, 0 for
                no: ");
                int toBool = GetInput.getInt();
                while (toBool != 1 || toBool != 0) {
                    System.out.println("Invalid
answer.\n");
                    System.out.println("Enter 1 for yes,
                    0 for no: ");
                }
                showAnswers = (toBool == 1);
                break;
            case 'g':
                System.out.println("Are students allowed to
System.out.println("Enter 1 for yes, 0 for no: ");

int toBool = GetInput.getInt();

while (toBool != 1 && toBool != 0) {
    System.out.println("Invalid answer.");
    System.out.println("Enter 1 for yes, 0 for no: ");
    toBool = GetInput.getInt();
}

showGrade = (toBool == 1);

break;

} else {
    // Pick and edit a question
    System.out.println("Which question do you want to edit? ");
    System.out.print("Enter question number (1-" + numQuestions + "):");

    int questionNum = GetInput.getInt();
    while (!(questionNum > 0 && questionNum <= numQuestions)) {
        System.out.println("Invalid Question Number!");
        System.out.println("Enter question number (1-" + numQuestions + "):");
        questionNum = GetInput.getInt();
    }

    editQuestion((Question)questions.get(questionNum - 1)); // Zero-indexed

    // Shows a test
    // Outputs the test meta-data and the questions, but not their answers
    public void showTest() {
        System.out.println("Test Details:");
        System.out.println("Test ID: " + testID);
        System.out.println("Test (p)assword: " + testPass);
        System.out.println("Show (a)nswers: " + ((showAnswers) ? "yes": "no"));
        System.out.println("Show (g)rades: " + ((showGrade) ? "yes": "no"));

        int i = 1;
        Iterator it = questions.iterator();
        while (it.hasNext()) {
            System.out.println(i++ + "." + ((Question)it.next()).getText());
        }
    }

    // End Work Functions
    // IO Functions
// Reads out of a file into the test
// We throw IOException so it can be handled higher up in the code
public void readFrom(String fname) throws IOException {
    RandomAccessFile fb = new RandomAccessFile(fname, "r");
    numQuestions = 0; // this will be set by addQuestion
    questions = new DoubleLinkedList();
    int nq = fb.readInt(); // number of questions in the file
    testID = fb.readInt();
    showGrade = fb.readBoolean();
    showAnswers = fb.readBoolean();
    byte[] ba = new byte[TEST_PASS_BYTES]; // byte array to read into
    fb.readFully(ba); // read test pass
    testPass = new String(ba).trim(); // turn into string
    for (int i = 1; i <= nq; i++) {
        addQuestion(readQuestion(fb));
    }
    fb.close();
}

// Reads a question from the passed RandomAccessFile
// This assumes fb is open and at right offset
// We throw IOException so it can be handled higher up in the code
public Question readQuestion(RandomAccessFile fb) throws IOException {
    byte[] ba = new byte[QTEXT_BYTES]; // byte array to read into
    fb.readFully(ba); // Read in the question
    String qt = new String(ba).trim(); // The question
    ba = new byte[CHOICE_BYTES];
    char ans = (char) fb.read(); // Reads in one byte, the correct answer
    fb.readFully(ba); // read in choice a
    String a = new String(ba).trim();
    fb.readFully(ba); // read in choice b
    String b = new String(ba).trim();
    fb.readFully(ba); // read in choice c
    String c = new String(ba).trim();
    fb.readFully(ba); // read in choice d
    String d = new String(ba).trim();
    return new Question(qt, a, b, c, d, ans, showGrade, showAnswers);
}

// Writes out of the test into a file
// We throw IOException so it can be handled higher up in the code
public void writeToFile(String fname) throws IOException {
    RandomAccessFile fb = new RandomAccessFile(fname, "rw");
    // Make sure they want to overwrite the file. Throw exception if not
    if (new java.io.File(fname).exists()) {
        System.out.print("Warning. Writing to file " + fname + " will overwrite it. Continue? (Y/N): ");
        if(Character.toLowerCase(getInput().getChar()) != 'y')
            throw new IOException("User aborted");
    }
    fb.setLength(0); // Kill any existing data
    fb.writeInt(numQuestions); // Write the number of questions
    fb.writeInt(testID); // Write the testID
fb.writeBoolean(showGrade); // Write showGrade
fb.writeBoolean(showAnswers); // Write showAnswers
StringBuffer sb; // Scratch StringBuffer to fix the length of the strings
    sb = new StringBuffer(testPass);
    sb.setLength(TEST_PASS_BYTES); // Make the proper size
    fb.writeBytes(sb.toString()); // Write to file
    Iterator it = questions.iterator();
    while (it.hasNext()) // Iterate through the questions
        writeQuestion(fb, (Question)it.next());
    fb.close();
}

// Writes a question to the passed RandomAccessFile
// This assumes fb is open and at right offset
// We throw IOException so it can be handled higher up in the code
public void writeQuestion(RandomAccessFile fb, Question q) throws IOException {
    StringBuffer sb; // StringBuffer to manipulate the data
    byte[] ba; // Byte array
    // Get, set length and write out the question text
    sb = new StringBuffer(q.getQText());
    sb.setLength(QTEXT_BYTES);
    fb.writeBytes(sb.toString());

    // Write correct answer
    fb.write((byte)q.getCorrectAnswer());

    // Write choice A
    sb = new StringBuffer(q.getChoiceA());
    sb.setLength(CHOICE_BYTES);
    fb.writeBytes(sb.toString());

    // Write choice B
    sb = new StringBuffer(q.getChoiceB());
    sb.setLength(CHOICE_BYTES);
    fb.writeBytes(sb.toString());

    // Write choice C
    sb = new StringBuffer(q.getChoiceC());
    sb.setLength(CHOICE_BYTES);
    fb.writeBytes(sb.toString());

    // Write choice D
    sb = new StringBuffer(q.getChoiceD());
    sb.setLength(CHOICE_BYTES);
    fb.writeBytes(sb.toString());
}

    // End IO Functions

/* EOF */
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;

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

    // Returns 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");
fb.seek(fb.length()); // Go to end of file
fb.writeInt(id);
StringBuffer sb = new StringBuffer(filename);
sb.setLength(TESTNAME_BYTES);
fb.writeBytes(sb.toString());
fb.close();

// Shows all tests in the database
public void showAll() throws IOException {
    // Open file buffer
    RandomAccessFile fb = new RandomAccessFile(fileName, "r");
    int numItems = (int)(fb.length()/TEST_SIZE); // How many items
    System.out.println("Test ID: File Name:");
    for (int i = 0; i < numItems; i++) { // iterate through
        fb.seek(i * TEST_SIZE); // go to beginning of the record
        System.out.print(fb.readInt()+ "\t\t" ); // ID
        byte[] ba = new byte[TESTNAME_BYTES];
        fb.readFully(ba);
        System.out.println(new String(ba).trim());
    }
    fb.close();
}

// Removes a test based on the id
// Since the test file is unordered, we will move
// the last item into the empty space.
// This will cause less I/O usage and make the operation MUCH faster
// (O(1) rather than O(n))
public void removeTest(int testId) throws NoSuchTestException, IOException {
    // Open file buffer
    RandomAccessFile fb = new RandomAccessFile(fileName, "rw");
    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 beginning of the record
        if (fb.readInt() == testId) { // the id is first
            fb.seek((numItems - 1) * TEST_SIZE); // go to last item

            int newId = fb.readInt();
            byte[] ba = new byte[TESTNAME_BYTES];
            fb.readFully(ba); // read in file name
            fb.seek(i * TEST_SIZE);
            fb.writeInt(newId); // overwrite id
            fb.write(ba); // overwrite filename
            fb.setLength((numItems - 1) * TEST_SIZE); // truncate file
            return;
        }
    }
    fb.close();
    throw new NoSuchTestException("Test number " + testId + " not found"); // item not found
}
110 class NoSuchTestException extends Exception {
111       public NoSuchTestException(String s) {
112               super(s);
113       }
114   }
115
116  /* EOF */