JavaScript is disabled on your browser.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Manager.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* Next Class
* [Frames](http://docs.google.com/index.html?onlineTest/Manager.html)
* [No Frames](http://docs.google.com/Manager.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* Constr |
* [Method](#3znysh7)
* Detail:
* Field |
* Constr |
* [Method](#2et92p0)

onlineTest

## Interface Manager

* public interface Manager

### Method SummaryMethods

|  |  |
| --- | --- |
| * + Modifier and Type | * + Method and Description |
| * + boolean | * + [**addExam**](http://docs.google.com/onlineTest/Manager.html#addExam(int,%20java.lang.String))(int examId, java.lang.String title) Adds the specified exam to the database. |
| * + void | * + [**addFillInTheBlanksQuestion**](http://docs.google.com/onlineTest/Manager.html#addFillInTheBlanksQuestion(int,%20int,%20java.lang.String,%20double,%20java.lang.String%5B%5D))(int examId, int questionNumber, java.lang.String text, double points, java.lang.String[] answer) Adds a fill-in-the-blanks question to the specified exam. |
| * + void | * + [**addMultipleChoiceQuestion**](http://docs.google.com/onlineTest/Manager.html#addMultipleChoiceQuestion(int,%20int,%20java.lang.String,%20double,%20java.lang.String%5B%5D))(int examId, int questionNumber, java.lang.String text, double points, java.lang.String[] answer) Adds a multiple choice question to the specified exam. |
| * + boolean | * + [**addStudent**](http://docs.google.com/onlineTest/Manager.html#addStudent(java.lang.String))(java.lang.String name) Adds the specified student to the database. |
| * + void | * + [**addTrueFalseQuestion**](http://docs.google.com/onlineTest/Manager.html#addTrueFalseQuestion(int,%20int,%20java.lang.String,%20double,%20boolean))(int examId, int questionNumber, java.lang.String text, double points, boolean answer) Adds a true and false question to the specified exam. |
| * + void | * + [**answerFillInTheBlanksQuestion**](http://docs.google.com/onlineTest/Manager.html#answerFillInTheBlanksQuestion(java.lang.String,%20int,%20int,%20java.lang.String%5B%5D))(java.lang.String studentName, int examId, int questionNumber, java.lang.String[] answer) Enters the question's answer into the database. |
| * + void | * + [**answerMultipleChoiceQuestion**](http://docs.google.com/onlineTest/Manager.html#answerMultipleChoiceQuestion(java.lang.String,%20int,%20int,%20java.lang.String%5B%5D))(java.lang.String studentName, int examId, int questionNumber, java.lang.String[] answer) Enters the question's answer into the database. |
| * + void | * + [**answerTrueFalseQuestion**](http://docs.google.com/onlineTest/Manager.html#answerTrueFalseQuestion(java.lang.String,%20int,%20int,%20boolean))(java.lang.String studentName, int examId, int questionNumber, boolean answer) Enters the question's answer into the database. |
| * + double | * + [**getAverageScore**](http://docs.google.com/onlineTest/Manager.html#getAverageScore(int))(int examId) Returns the average score for the specified exam. |
| * + java.lang.String | * + [**getCourseGrades**](http://docs.google.com/onlineTest/Manager.html#getCourseGrades())() Returns a listing with the grades for each student. |
| * + java.lang.String | * + [**getCourseLetterGrade**](http://docs.google.com/onlineTest/Manager.html#getCourseLetterGrade(java.lang.String))(java.lang.String studentName) Computes a letter grade based on cutoffs provided. |
| * + double | * + [**getCourseNumericGrade**](http://docs.google.com/onlineTest/Manager.html#getCourseNumericGrade(java.lang.String))(java.lang.String studentName) Computes a numeric grade (value between 0 and a 100) for the student taking into consideration all the exams. |
| * + double | * + [**getExamScore**](http://docs.google.com/onlineTest/Manager.html#getExamScore(java.lang.String,%20int))(java.lang.String studentName, int examId) Returns the score the student got for the specified exam. |
| * + java.lang.String | * + [**getGradingReport**](http://docs.google.com/onlineTest/Manager.html#getGradingReport(java.lang.String,%20int))(java.lang.String studentName, int examId) Generates a grading report for the specified exam. |
| * + java.lang.String | * + [**getKey**](http://docs.google.com/onlineTest/Manager.html#getKey(int))(int examId) Returns a string with the following information per question:  "Question Text: " followed by the question's text  "Points: " followed by the points for the question  "Correct Answer: " followed by the correct answer. |
| * + double | * + [**getMaxScore**](http://docs.google.com/onlineTest/Manager.html#getMaxScore(int))(int examId) Returns the maximum score (among all the students) for the specified exam. |
| * + double | * + [**getMinScore**](http://docs.google.com/onlineTest/Manager.html#getMinScore(int))(int examId) Returns the minimum score (among all the students) for the specified exam. |
| * + [Manager](http://docs.google.com/onlineTest/Manager.html) | * + [**restoreManager**](http://docs.google.com/onlineTest/Manager.html#restoreManager(java.lang.String))(java.lang.String fileName) It will return a Manager object based on the serialized data found in the specified file. |
| * + void | * + [**saveManager**](http://docs.google.com/onlineTest/Manager.html#saveManager(onlineTest.Manager,%20java.lang.String))([Manager](http://docs.google.com/onlineTest/Manager.html) manager, java.lang.String fileName) It will serialize the Manager object and store it in the specified file. |
| * + void | * + [**setLetterGradesCutoffs**](http://docs.google.com/onlineTest/Manager.html#setLetterGradesCutoffs(java.lang.String%5B%5D,%20double%5B%5D))(java.lang.String[] letterGrades, double[] cutoffs) Sets the cutoffs for letter grades. |

### Method Detail

#### addExam boolean addExam(int examId, java.lang.String title) Adds the specified exam to the database.Parameters:examId - title - Returns:false is exam already exists.

#### addTrueFalseQuestion void addTrueFalseQuestion(int examId, int questionNumber, java.lang.String text, double points, boolean answer) Adds a true and false question to the specified exam. If the question already exists it is overwritten.Parameters:examId - questionNumber - text - Question textpoints - total pointsanswer - expected answer

#### addMultipleChoiceQuestion void addMultipleChoiceQuestion(int examId, int questionNumber, java.lang.String text, double points, java.lang.String[] answer) Adds a multiple choice question to the specified exam. If the question already exists it is overwritten.Parameters:examId - questionNumber - text - Question textpoints - total pointsanswer - expected answer

#### addFillInTheBlanksQuestion void addFillInTheBlanksQuestion(int examId, int questionNumber, java.lang.String text, double points, java.lang.String[] answer) Adds a fill-in-the-blanks question to the specified exam. If the question already exits it is overwritten. Each correct response is worth points/entries in the answer.Parameters:examId - questionNumber - text - Question textpoints - total pointsanswer - expected answer

* + - getKey  
      java.lang.String getKey(int examId)  
      Returns a string with the following information per question:  
        
      "Question Text: " followed by the question's text  
        
      "Points: " followed by the points for the question  
        
      "Correct Answer: " followed by the correct answer.  
        
      The format for the correct answer will be:  
        
      a. True or false question: "True" or "False"  
        
      b. Multiple choice question: [ ] enclosing the answer (each entry separated by commas) and in sorted order.  
        
      c. Fill in the blanks question: [ ] enclosing the answer (each entry separated by commas) and in sorted order.  
      Parameters:examId - Returns:"Exam not found" if exam not found, otherwise the key

#### addStudent boolean addStudent(java.lang.String name) Adds the specified student to the database. Names are specified in the format LastName,FirstNameParameters:name - Returns:false if student already exists.

#### answerTrueFalseQuestion void answerTrueFalseQuestion(java.lang.String studentName, int examId, int questionNumber, boolean answer) Enters the question's answer into the database.Parameters:studentName - examId - questionNumber - answer -

#### answerMultipleChoiceQuestion void answerMultipleChoiceQuestion(java.lang.String studentName, int examId, int questionNumber, java.lang.String[] answer) Enters the question's answer into the database.Parameters:studentName - examId - questionNumber - answer -

#### answerFillInTheBlanksQuestion void answerFillInTheBlanksQuestion(java.lang.String studentName, int examId, int questionNumber, java.lang.String[] answer) Enters the question's answer into the database.Parameters:studentName - examId - questionNumber - answer -

#### getExamScore double getExamScore(java.lang.String studentName, int examId) Returns the score the student got for the specified exam.Parameters:studentName - examId - Returns:score

* + - getGradingReport  
      java.lang.String getGradingReport(java.lang.String studentName,  
       int examId)  
      Generates a grading report for the specified exam. The report will include the following information for each exam question:  
        
      "Question #" {questionNumber} {questionScore} " points out of " {totalQuestionPoints}  
        
      The report will end with the following information:  
        
      "Final Score: " {score} " out of " {totalExamPoints};Parameters:studentName - examId - Returns:report

#### setLetterGradesCutoffs void setLetterGradesCutoffs(java.lang.String[] letterGrades, double[] cutoffs) Sets the cutoffs for letter grades. For example, a typical curve we will have new String[]{"A","B","C","D","F"}, new double[] {90,80,70,60,0}. Anyone with a 90 or above gets an A, anyone with an 80 or above gets a B, etc. Notice we can have different letter grades and cutoffs (not just the typical curve).Parameters:letterGrades - cutoffs -

#### getCourseNumericGrade double getCourseNumericGrade(java.lang.String studentName) Computes a numeric grade (value between 0 and a 100) for the student taking into consideration all the exams. All exams have the same weight.Parameters:studentName - Returns:grade

#### getCourseLetterGrade java.lang.String getCourseLetterGrade(java.lang.String studentName) Computes a letter grade based on cutoffs provided. It is assumed the cutoffs have been set before the method is called.Parameters:studentName - Returns:letter grade

* + - getCourseGrades  
      java.lang.String getCourseGrades()  
      Returns a listing with the grades for each student. For each student the report will include the following information:  
        
      {studentName} {courseNumericGrade} {courseLetterGrade}  
        
      The names will appear in sorted order.Returns:grades

#### getMaxScore double getMaxScore(int examId) Returns the maximum score (among all the students) for the specified exam.Parameters:examId - Returns:maximum

#### getMinScore double getMinScore(int examId) Returns the minimum score (among all the students) for the specified exam.Parameters:examId - Returns:minimum

#### getAverageScore double getAverageScore(int examId) Returns the average score for the specified exam.Parameters:examId - Returns:average

#### saveManager void saveManager([Manager](http://docs.google.com/onlineTest/Manager.html) manager, java.lang.String fileName) It will serialize the Manager object and store it in the specified file.

#### restoreManager [Manager](http://docs.google.com/onlineTest/Manager.html) restoreManager(java.lang.String fileName) It will return a Manager object based on the serialized data found in the specified file.

* [Overview](http://docs.google.com/overview-summary.html)
* [Package](http://docs.google.com/package-summary.html)
* Class
* [Use](http://docs.google.com/class-use/Manager.html)
* [Tree](http://docs.google.com/package-tree.html)
* [Deprecated](http://docs.google.com/deprecated-list.html)
* [Index](http://docs.google.com/index-files/index-1.html)
* [Help](http://docs.google.com/help-doc.html)
* Prev Class
* Next Class
* [Frames](http://docs.google.com/index.html?onlineTest/Manager.html)
* [No Frames](http://docs.google.com/Manager.html)
* [All Classes](http://docs.google.com/allclasses-noframe.html)
* Summary:
* Nested |
* Field |
* Constr |
* [Method](#3znysh7)
* Detail:
* Field |
* Constr |
* [Method](#2et92p0)