Computer Science 111 Intro to Algorithms and Programming: Java

Programming Project #6 Classes and Objects (30 points)

Due on 12/6/18

For this last project, you must implement a Java class which “simulates” a Student object. Student objects created from this class will represent grade records for students in a CMPSCI 111 lecture class. The UML class diagram below gives you an “outline” for the class:

Student
-name : String -sid : String -homework : double[] -quizzes : double[] -exams : double[] ~NUM_HOMEWORK : int ~NUM_QUIZZES : int ~NUM_EXAMS : int ~HOMEWORK_MAX_POINTS : double ~QUIZ_MAX_POINTS : double ~MIDTERM_MAX_POINTS : double ~FINAL_MAX_POINTS : double
+Student() +Student(newName : String) +Student(newName : String, newSid : String) +setName(newName : String) : void +getName() : String +setSid(newSid : String) +getSid() : String +setHomework(homeworkNumber : int, score : double) : void +getHomework(homeworkNumber : int) : double +setQuiz(quizNumber : int, score : double) : void +getQuiz(quizNumber : int) : double +setMidtermExam(score : double) : void +getMidtermExam() : double +setFinalExam(score : double) : void +getFinalExam() : double +toString() : String

The Java class definition that you write **must** implement all of the data fields (variables) and behaviors (methods) shown in the above UML class diagram. Therefore, the class you write must contain all of the following global variables, constants and methods:

Variables and constants

- A **private** String-type variable named **name** – this will represent a Student's name.

- A **private** String-type variable named **sid** – this will represent a Student's Student ID Number.

- A **private** reference variable named **homework** which references an array of double-type values – this array of double values will have **NUM_HOMEWORK** length and each element will be used to store one of the student's homework scores. For example, if homework[0] is set to a value of 5.0, this would mean that the student scored a 5.0 on Homework 1.

- A **private** reference variable named **quizzes** which references an array of double-type values – this array of double values will have **NUM_QUIZZES** length and each element will be used to store one of the student's quiz scores. For example, if quizzes[0] is set to a value of 19.5, this would mean that the student scored a 19.5 on Quiz 1.

- A **private** reference variable named **exams** which references an array of double-type values – this array of double values will have **NUM_EXAMS** length and each element will be used to store one of the student's exam scores. For example, if exams[0] is set to a value of 36.5, this would mean that the student scored a 36.5 on the Midterm Exam.

- A **final static int** constant named **NUM_HOMEWORK** – this constant will represent the number of homework assignments (4) which are given over the course of a semester.

- A **final static int** constant named **NUM_QUIZZES** – this constant will represent the number of quizzes (4) which are given over the course of a semester.

- A **final static int** constant named **NUM_EXAMS** – this constant will represent the number of exams (2) which are given over the course of a semester.

- A **final static double** constant named **HOMEWORK_MAX_POINTS** – this constant will represent the maximum number of points (5) for any homework assignment given over the course of a semester.

- A **final static double** constant named **QUIZ_MAX_POINTS** – this constant will represent the maximum number of points (20) for any quiz given over the course of a semester.

- A **final static double** constant named **MIDTERM_MAX_POINTS** – this constant will represent the maximum number of points (40) for the Midterm Exam.

- A **final static double** constant named **FINAL_MAX_POINTS** – this constant will represent the maximum number of points (60) for the Final Exam.

Constructors

- A no-arg constructor which takes no arguments. This constructor is responsible for setting the object's **name** variable to a default value of "Newstudent, A." and the object's **sid** variable to a default value of "0000000". This constructor should also allocate (create) the three arrays; **homework** with a length of **NUM_HOMEWORK**, **quizzes** with a length of **NUM_QUIZZES** and **exams** with a length of **NUM_EXAMS**.

- A constructor which takes one argument, a String, and stores that argument in a String-type parameter named **newName**. This parameter represents the name of a new Student object. This constructor is responsible for setting the object's **name** variable to the given **newName** and the object's **sid** variable to a default value of "0000000". This constructor should also allocate (create) the three arrays; **homework** with a length of **NUM_HOMEWORK**, **quizzes** with a length of **NUM_QUIZZES** and **exams** with a length of **NUM_EXAMS**.

- A constructor which takes two String arguments and stores those arguments in String-type parameters named **newName** and **newSid**. These parameters represent the name and sid of a new Student object. This constructor is responsible for setting the object's **name** variable to the given **newName** and the object's **sid** variable to the given **newSid**. This constructor should also allocate (create) the three arrays; **homework** with a length of **NUM_HOMEWORK**, **quizzes** with a length of **NUM_QUIZZES** and **exams** with a length of **NUM_EXAMS**.

Public Interface Methods

- **setName** – A method which takes one String argument and stores that argument in a String-type parameter named **newName**. This method is responsible for setting the object's **name** variable to the given **newName**. This method should return **void**.

- **getName** – A method which takes no arguments. This method is responsible for returning the value of the object's **name** variable. This method should return **String**.

- **setSid** – A method which takes one String argument and stores that argument in a String-type parameter named **newSid**. This method is responsible for setting the object's **sid** variable to the given **newSid**. This method should return **void**.

- **getSid** – A method which takes no arguments. This method is responsible for returning the value of the object's **sid** variable. This method should return **String**.

- **setHomework** – A method which takes two arguments, an int and a double, and stores those arguments in an int-type parameter named **homeworkNumber** and a double-type parameter named **score**. This method should check to ensure **homeworkNumber** is a number between 1 and **NUM_HOMEWORK** and that **score** is a number between 0 and **HOMEWORK_MAX_POINTS** and if these conditions are true, the program should assign the corresponding element of the **homework** array the value of **score**. For example, if **homeworkNumber** is 1 and score is 5, then the element homework[0] should be assigned the value of 5. This method should return **void**.

- **getHomework** – A method which takes one int argument and stores this argument in a parameter named **homeworkNumber**. This method is responsible for returning one of the values in the object's **homework** array. This method should check to ensure **homeworkNumber** is a number between 1 and **NUM_HOMEWORK** and if this condition is true, the method should return the double value at index **homeworkNumber – 1** in the **homework** array. Otherwise, the method should return **0**. This method should return **double**.

- **setQuiz** – A method which takes two arguments, an int and a double, and stores those arguments in an int-type parameter named **quizNumber** and a double-type parameter named **score**. This method should check to ensure **quizNumber** is a number between 1 and **NUM_QUIZZES** and that **score** is a number between 0 and **QUIZ_MAX_POINTS** and if these conditions are true, the program should assign the corresponding element of the **quizzes** array the value of **score**. For example, if **quizNumber** is 1 and score is 18, then the element quiz[0] should be assigned the value of 18. This method should return **void**.

- **getQuiz** – A method which takes one int argument and stores this argument in a parameter named **quizNumber**. This method is responsible for returning one of the values in the object's **quizzes** array. This method should check to ensure **quizNumber** is a number between 1 and **NUM_QUIZZES** and if this condition is true, the method should return the double value at index **quizNumber – 1** in the **quizzes** array. Otherwise, the method should return **0**. This method should return **double**.

- **setMidtermExam** – A method which takes one double argument and stores this argument in a parameter named **score**. This method should check to ensure **score** is between 0 and **MIDTERM_MAX_POINTS**. If this condition is true, the method should assign the element exams[0] the value of **score**. This method should return **void**.

- **getMidtermExam** – A method which takes no arguments. This method should return the value of exams[0]. This method should return **double**.

- **setFinalExam** – A method which takes one double argument and stores this argument in a parameter named **score**. This method should check to ensure that score is between 0 and

FINAL_MAX_POINTS. If this condition is true, the method should assign the element exams[1] the value of **score**. This method should return **void**.

- **getFinalExam** – A method which takes no arguments. This method should return the value of exams[1]. This method should return **double**.

- **toString** – A method which takes no arguments. This method should return a String onto which the student object's name, sid, homework scores, quiz scores and exam scores have all been concatenated.

NOTE: Your class MUST be named Student, and all of the variable declarations and method definitions must follow this specification *exactly*. At a later time, I will post a test program which will allow you to test your Student class by creating objects from it, storing data in those objects, and then processing the data in those objects. Your class should work with my test program with no modification to either my test program or your Student class.