Quality Management Report Version 0.5

CSC 480/HCI 521: Software Design
Updated 5-8-19

Table of Contents

1. Quality Management Approach

Our goal is to ensure that bubble sheets are evaluated correctly and consistently. There should be a low margin of error, ideally as close to zero errors as possible, with up to 5% of marking errors being acceptable. It is very important that a students grade is correct, and that the bubble sheet answers are scanned correctly. Each component of this system shall be tested thoroughly and the system should display no incorrect data. The testing that shall be done should ensure that each component works as expected.

1.1 Synopsis

Creating a software which calculates student's grades is highly beneficial considering the tedious task at hand, however, extensive testing is needed to ensure the system requirements are operating correctly and efficiently.

1.2 Testing Plan

We will use **Spock** to unit test Java code, we will use **PyTest** to test Python code, and we will be using **Travis Continuous Integration** to help streamline our pull requests and code changes, Travis will also run all unit tests to ensure that the code does not break the build before a pull request can be submitted. Java code will be compiled and automatically tested with a gradle build. We may also use Selenium and or Jest to test GUI's webpage, allowing for automatic testing and scheduled testing which can be used to test if something is wrong with our webpage in the future.

Two people will check pull requests. A member of QA to run various tests and a different member of the team pushing the changes, to ensure the code changes are up to the teams standards.

1.2.1 Possible tests

Possible tests to be chosen from for testing each component. We most likely will not create each possible test for each component as that would be an insane amount of tests

Almost all code should have unit testing at minimum. All other options for testing are listed below:

1. Specification Based Testing
2. Control Flow Based Testing
    2.1. Minimal Multiple Condition Testing

(This test subsumes branch testing which subsumes statement testing, making those tests unnecessary if this is done.)

    2.2. Path Coverage Testing
    2.3. Data Flow Testing
3. Class Testing
    3.1. Category Partition Test
    3.2. Source Code Test
    3.3. Polymorphism Test
4. Class Scope Test
    4.1. State Transition Matrix
    4.2. All Events Criterion
    4.3. Flattened Class Scope Test
    4.4. Class Interaction Test
5. Unit Tests
6. Integration Tests
7. System Tests
8. Acceptance Test
9. Ongoing Tests and Future Testability (Excitor Not MVP)

1.2.2 Testing based Use Case In SRS.

Each use-case must work as expected. Using use-cases as a basis will allow us to test every potential scenario of the system. This will also enable us to test how the different components interact, not just the components acting alone that unit testing will accomplish. This will be very useful for testing the system once it is operational to ensure it is ready for deployment, and could be specifically helpful for System Testing and Acceptance Testing.

Use cases define specific scenarios and what should happen, giving us clear guidelines for our system to follow, and for testing clear criteria that our system should follow, by testing against this, we can easily see if our system is doing what it is required to do by these use cases.

1.3 Components

Last Updated - May 1 2019

1. Java Code:

a. Database:
    i. AnswerKeyDao.java - A database Java component.
    ii. ConnectionFactory.java - A database Java component.
    iii. CourseDao.java - A database Java component.
    iv. ExamDao.java - A database Java component.
    v. GenericDao.java - CourseDao.java - A database Java component.
    vi. InstructorDao.java - CourseDao.java - A database Java component.
b. GUIMiddleware
    i. ExamManager.java
    ii. Grader.java
    iii. JSONBuilder.java
    iv. LetterConverter.java
    v. Main.java
    vi. MiddlewareInterface.java
    vii. PseudoStudent.java
    viii. Question.java
    ix. Statistics.java
    x. Stats.java
    xi. Student.java
    xii. StudentCreater.java
c. Scanner
    i. Main.java
    ii. OrientTool.java
    iii. Utilities.java
d. Smartron
    i. AnswerKey.java entity
    ii. Exam.java entity
    iii. Professor.java entity
    iv. AnswerKeyServlet.java
    v. ExamServlet.java
    vi. NameChangeServlet.java
    vii. QuestionServlet.java
    viii. StatisticServlet.java
    ix. StudentServlet.java

2. Python Code:
    a. edgey.py
    b. imapConnect.py
    c. 3.35 sendEmail.py

    d. 3.36 sendEmailHandler.py
3. SQL
    a. SMARTron.sql
4. WebApp
    a. AnswerKey.js
    b. App.js
    c. Exam.js
    d. ExamList.js
    e. Header.js
    f. Home.js
    g. Login.js
    h. Router.js
    i. QuestionTable.js
    j. Results.js
    k. StatsTable.js
    l. StudentTable.js
    m. Tabs.js

1.4 Definitions

Term	Definition
SRS	SMARTron Software Requirements Specification **Last Referenced:** Feb 14th 2019
QA	Quality Assurance
Bubble Sheet	The answer sheet given to the student to fill out, this is the sheet that will be scanned and analyzed by the system.
High Criticality Tests	A high criticality test will be defined as a test pertaining to necessary to core functions of the system. For example any tests testing how the engine reads, and grades bubble sheets, computes statistics, connects to the database, and recomputes statistics, are all tests that must pass.

2. Quality Properties, Metrics, and Criteria

Property	Definition	Metric	Criterion
Correctness	The degree of quality and level of error-free the system is.	Based on the execution of test cases based off of the specifications provided by the	• The system is considered correct if it runs on Bastian's

		requirements team..	Macbook Pro ● All test cases marked as having high criticality must pass. ● Along with testing, the Component must be successfully reviewed by a third party member from either the Database or Engine teams to be considered correct.
Usability	The degree to which the system is able to used with ease by the user	To determine usability, the system must follow usability guidelines ● Satisfaction ● Efficiency	● The system is considered to be usable if the stakeholder is satisfied with the usability of the design ● Efficiency: The system should complete faster than previous method of grading exams, with less effort. The system will also grade tests as stated in the accuracy property. ● Success rate: The user success rate on specific tasks. Tasks will be based on system requirements
Security	All the measures that are taken to protect data, or to ensure that only users with appropriate permission have access.	Ability of the system to determine authorization and be usable only by said authorized users as well as only send data to authorized users. The system must prevent information loss.	● The system is considered Secure if the data is only accessible by the authorized user that requested the data. ● The system has to retain data until user indicates otherwise so that results can be altered by the user.
Traceability	The ability to link product requirements back to stakeholders' rationales and towards corresponding design artifacts, code, and	Based off of the traceability matrix are all use cases matched with a test case.	● The system is considered to conform to traceability if all of the requirements dictated by the stakeholder are

	test cases.		reflected by the code.
Accuracy	The degree to which the result of a measurement, calculation, or specification conforms to the correct value or a standard.	Ability of the system to grade bubble sheets with precision, to detect defects in the sheet, and to precisely compute a statistical analysis of the graded results.	• System is considered accurate when 95% of well marked bubble sheets are graded correctly. • bubble sheet that are not well marked or have some type of defect are not expected to be graded accurately but must be detected as defects for manual review. • All statistics should be computed accurately..

https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Master Test Table

ID	Component Description	QA tester(s)	Tester from other team	Tests Done	Version /Date
1	CourseDao.java			Specification based	outdated
2	ExamDao.java			Specification based	outdated
3	AnswerKeyDao.java			Specification based	outdated
4	ConnectionFactory.java, Class is deprecated	Tim Quinn		Specification based	removed
5	GenericDao.java, Class is depreciated in the current database	Sean		Specification based	removed
6	InstructorDao.java, is in charge of adding and removing instructors from the database.	Sean		Specification based	5-8-19
7.3.2	Web App, Testing that the components on the web page work and are displayed correctly.	Quinn		Selenium Functional/Unit Testing	4-20-19
7	ExamManager.java, a controller for creating student objects and grading	Quinn		Specification	4-20-19

	them				
8	Grader.java	Quinn		Specification CFG	5-1-19
9	JSONBuilder.java	Quinn		Specification	4-26-19 outdated
10	LetterConverter.java	Quinn		Specification CFG	4-25-19 4-28-19
11	Main.java			May not need testing	
12	MiddlewareInterface.java	Quinn		Specification CFG	4-29-19
13	Question.java	Quinn		Specification	4-25-19
14	Stats.java - In charge of all statistical analysis and formulas - replaced Statistics.java which was removed	Sean		Specification	5-8-19
16	Student.java	Quinn		Specification	5-7-19
17	StudentCreater.java	Quinn		Specification	5-7-19
19	Main.java			May not need testing	
20	OrientTool.java	Quinn		Specification	4-23-19 4-24-19
21	Utilities.java	Quinn		Specification	4-24-19 Outdated
22	Answerkey.java	Ryan		Specification	5-3-19
23	Exam.java	Ryan		Specification	
24	Professor.java	Ryan		Specification	
25	AnswerkeyServlet.java				
26	ExamServlet.java				
27	NameChangeServlet.java				
28	QuestionServlet.java				

29	ResultServlet.java				
30	StatisticServlet.java				
31	StudentServlet.java				
32	edgey.py				
33	imapConnect.py				
34	sendEmail.py				
35	sendEmail.Handler.py				

3. Specification Based Testing

*Note, in all cases, x is the parameter

3.1 CourseDao.Java

3.1.1 CourseDao()

This method has no parameters, therefore equivalence classes do not apply.

3.1.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.1.3 addCourse(String courseCrn, String courseName, String secNum, String sem, String instId)

Input	Equivalence Class		Representative
	ID	Description	

courseCrn	1.1	x is a non-empty string	“K9632”
	1.a	x is an empty string	“”
	1.b	x is not a string	42.0

** courseName,secNum, sem, instId are all string inputs as well and share the same equivalence classes as courseCrn. Equivalence classes {1.1, 1.a, 1.b}.

3.1.4 deleteCourse(String courseCrn, String secNum, String sem)

Input	Equivalence Class		Representative
	ID	Description	
courseCrn	1.1	x is a non-empty string	“SoftwareDesign”
	1.a	x is an empty string	“”
	1.b	x is not a string	NULL

** secNum and sem are both string inputs as well and share the same equivalence classes as courseCrn. Equivalence classes {1.1, 1.a, 1.b}.

3.1.5 closeConnections()

This method has no parameters, therefore equivalence classes do not apply.

3.2 ExamDao.Java

3.2.1 ExamDao()

This method has no parameters, therefore equivalence classes do not apply.

3.2.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.2.3 addExam(String firstName, String lastName, String stdtId, String semester, String birthDate, String courseCrn, String instId, String examId)

Input	Equivalence Class		Representative
	ID	Description	
firstName	1.1	x is a non-empty string	"Kevin"
	1.a	x is an empty string	""
	1.b	x is not a string	Transponder()

** lastName, stdtId, semester, birthDate, courseCrn, instId, and examId are all string inputs as well and share the same equivalence classes as firstName. Equivalence classes {1.1, 1.a, 1.b}

3.2.4 deleteExam(String examId)

Input	Equivalence Class		Representative
	ID	Description	
examId	1.1	x is a non-empty string	"SQDZR"
	1.a	x is an empty string	""
	1.b	x is not a string	NULL

3.2.5 closeConnections()

This method has no parameters, therefore equivalence classes do not apply.

3.3 AnswerKeyDao.java

3.3.1 AnswerKeyDao()

This method has no parameters, therefore equivalence classes do not apply.

3.3.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.3.3 addAnswerKey (String examId, String instId, String answers, String answersLength)

Input	Equivalence Class		Representative
	ID	Description	
examId	1.1	x is a non-empty string	“Exam73”
	1.a	x is an empty string	“”
	1.b	x is not a string	NULL

** instId, answers, and answersLength is also a string and it shares equivalence classes {1.1, 1.a, 1.b}

Test Case ID	instId	EQ Class(es) demonstrated	Expected Result

TC#1.1	“TESTID213524”	1.1	Returns 1
TC#2	“”	1.a	Returns 0
TC#3	NULL	1.b	SQLException thrown
TC#4	5	1.b	SQLException thrown

3.3.4 deleteAnswerKey (String examId)

Input	Equivalence Class		Representative
	ID	Description	
examId	1.1	x is a non-empty string	“A non-empty string”
	1.a	x is an empty string	“”
	1.b	x is not a string	NULL

3.3.5 closeConnections()

This method has no parameters, therefore equivalence classes do not apply.

3.4 ConnectionFactory.java, Deprecated, no longer used.

3.4.1 ConnectionFactory()

This method has no parameters, therefore equivalence classes do not apply.

3.4.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.4.3 getInstance()

This method has no parameters, therefore equivalence classes do not apply.

3.5 GenericDao.java, Deprecated no longer used.

3.5.1 GenericDao()

This method has no parameters, therefore equivalence classes do not apply.

3.5.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.5.3 insert(String sql)

Equivalence Classes

Input	Equivalence Class		Representative
	ID	Description	
sql	1.1	x is a valid sql string	"insert into exam (first_name, last_name, student_id, semester, birth_date, course_crn, instructor_id, exam_id) values (Kevin, Johnson, 804028361, "Spring2019", "03/09/97", 32474, "id7654", "SZVYT") "
	1.a	x is not a valid sql string	"Just a string."
	1.b	x is not a string	470

Test Cases

3.5.4 delete(String sql)

Equivalence Classes

Input	Equivalence Class ID	Equivalence Class Description	Representative
sql	1.1	x is a valid sql string	"delete from exam *"
	1.a	x is not a valid sql string	“Still just a string.”
	1.b	x is not a string	550.0

Test Cases

3.5.5 select(String sql)

3.5.5.1 Equivalence Classes

Input	Equivalence Class ID	Equivalence Class Description	Representative
sql	1.1	x is a valid sql string	“Select firstname from exam where firstname=’Kevin’;”
	1.a	x is not a valid sql string	“Where art thou Spring?”
	1.b	x is not a string	NULL

Test Cases

3.5.6 closeConnections()

This method has no parameters, therefore equivalence classes do not apply.

3.6 InstructorDao.java

3.6.1 InstructorDao()

This method has no parameters, therefore equivalence classes do not apply.

3.6.2 getConnection()

This method has no parameters, therefore equivalence classes do not apply.

3.6.3 addInstructor (String instId, String instFirstName, String instLastName)

3.6.3.1 Equivalence Classes

Input	Equivalence Class		Representative
	ID	Description	
instId	1.1	x is a non-empty string of size =< 200	"TESTID213524"
	1.a	x is an empty string	""
	1.b	x is not a string	NULL
	1.c	x is a string larger than 200	"STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524"

** instFirstName and instLastName are also strings that share equivalence classes {1.1, 1.a, 1.b, 1.c}

3.6.3.2 Test Cases

Test Case ID	instId	instFirstName	instLastName	EQ Class(es) demonstrated	Expected Result
TC#1	"TESTID213524"	Bastian	Tenbergen	1.1, 2.1, 3.1	Returns 1
TC#2	""	""	""	1.a, 2.a, 3.a	Returns 0
TC#3	null	null	null	1.b, 2.b, 3.b	SQLException thrown
TC#4	5	6	7	1.b, 2.b, 3.b	SQLException thrown
TC#5	"STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524"	"STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524"	"STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524STID213524"	1.c, 2.c, 3.c	Data truncated exception thrown

3.6.3.3 Boundary Value Analysis

Boundary value analysis was simple and therefore included in the above test cases and equivalence classes.

3.6.4 deleteInstructor (String instId)

3.6.4.1 Equivalence Classes

Input	Equivalence Class		Representative
	ID	Description	

instId	1.1	x is a non-empty string	“TESTID213524”
	1.a	x is an empty string	“”
	1.b	x is not a string	null

3.6.4.2 Test Cases

Test Case ID	instId	EQ Class(es) demonstrated	Expected Result
TC#1	“TESTID213524”	1.1	Returns 1
TC#2	“”	1.a	Returns 0
TC#3	NULL	1.b	SQLException thrown
TC#4	5	1.b	SQLException thrown

3.6.4.3 Boundary Value Analysis

This test is included in 3.6.3

3.6.5 selectInstructor (String firstName, String lastName)

3.6.5.1 Equivalence Classes

Parameter	Equivalence Class	Representative

	ID	Description	
firstName*	1.1	x is a non-empty string	“TESTID213524”
	1.a	x is an empty string	“”
	1.b	x is not a string	NULL

*String lastName has same functionality as firstName and as such shares EQ’s and test cases under 2.1-2.b

3.6.5.2 Test Cases

Test Case ID	instId	EQ Class(es) demonstrated	Expected Result
TC#1	“TESTNAME”	1.1, 2.1	[“TESTID”]
TC#2	“”	1.a, 2.a	Exception thrown
TC#3	NULL	1.b, 2.b	Exception thrown
TC#4	5	1.b, 2.b	Exception thrown

3.6.6 closeConnections()

This method has no parameters, therefore equivalence classes do not apply.

3.6.7 getInstIdFromEmail(String email)

This method is never used and therefore requires no testing at this time.

3.7 ExamManager.java

3.7.1 addStudentExam(String[][] answers, String[] studentData)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
answers	1.1	x is a valid String [][]	{{"1"}{"3"}{"1","4"}{}}
	1.a	x is not a valid String [][]	[{"IMASTRING"}{"Yolo"}{"Memes"}]
	1.b	x is not a String [][]	NULL
studentData	2.1	x is a valid String []	{"Quinn Riley","","","","","804823115","111111"}
	2.a	x is not a valid String []	{"HEHEHEHEHE1231"}
	2.b	x is not a String []	NULL

Test Cases

Test Case ID	answers	studentData	EQ Class(es) demonstrated	Expected Result
TC#1	{{“1”}{“3”}{“1”,”4”}{}}	{“Quinn Riley”,””,””,””,”8048823115”,”111111”}	1.1 , 2.1	New Student created and added to the studentExams ArrayList
TC#2	{{“IMASTRING”}{“Yolo”}{“Memes”}}	{“123 abc ”,”SOMETIMES CODE IS HARD”}	1.a , 2.a	FAIL
TC#3	42	‘a’	1.b , 2.b	FAIL
TC#4	NULL	NULL	1.b , 2.b	FAIL

3.7.2 getGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.7.3 getExamGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.7.4 getStats()

This method has no parameters, therefore equivalence classes do not apply.

3.7.5 getExamLetterGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.7.6 examGradeIntegerConverter()

This method has no parameters, therefore equivalence classes do not apply.

3.7.7 getExamID()

This method has no parameters, therefore equivalence classes do not apply.

3.7.8 assignStudentGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.7.9 getStudents()

This method has no parameters, therefore equivalence classes do not apply.

3.8 Grader.java

3.8.1 getGrades(List<Student> studentExams, Student key)

Parameter	Equivalence Class		Representative
	ID	Description	
studentExams	1.1	x is a valid List<Student>	{Student,Student}
	1.a	x is not a valid List	{"Awesome","String","List"}
	1.b	x is not a List	NULL
key	2.1	x is a valid Student	Student
	2.a	x is not a valid Student	
	2.b	x is not a Student	NULL

Test Cases

Test Case ID	studentExams	key	EQ Class(es) demonstrated	Expected Result
TC#1	{Student, Student}	Student	1.1 , 2.1	Returns a list of floats for grades
TC#2	{“!!!!!”,”Yikes”}		1.a , 2.a	FAIL
TC#3	“STUDENTS ALL FAIL”	1234	1.b , 2.b	FAIL
TC#4	NULL	NULL	1.b , 2.b	FAIL

3.8.2 initializeStatsByQuestion(List<String> key)

Parameter	Equivalence Class		Representative
	ID	Description	
key	1.1	x is a valid List<Strings>	{“A”,”B”,”A”}
	1.a	x is not a valid List<String>	{“Awesome”,”String”,”List”,”Part”,”2”}
	1.b	x is not a List<String>	{1,2,3,4,5}
	1.c	x is not a List	NULL

Test Cases

Test Case ID	key	EQ Class(es) demonstrated	Expected Result
TC#1	{“A”,”B”,”A”}	1.1	Adds key.size() number of new Question() to questions
TC#2	{“Awesome”,”String” ,”List”,”Part”,”2”}	1.a	Adds key.size() number of new Question() to questions
TC#3	{1234}	1.b	FAIL
TC#4	“HI”	1.c	FAIL
TC#5	NULL	1.c	Fail

3.8.3 getStatsByQuestion()

This method has no parameters, therefore equivalence classes do not apply.

3.8.4 findKeyLength(Student key)

Parameter	Equivalence Class		Representative
	ID	Description	

key	1.1 x is a valid Student 1.a x is not a valid Student 1.b x is not a Student	New Student() NULL

Test Cases

Test Case ID	key	EQ Class(es) demonstrated	Expected Result
TC#1	New Student()	1.1	Returns the size of the students answers AKA the length of the answer key
TC#2		1.a	
TC#3	{1234}	1.b	FAIL
TC#4	NULL	1.b	FAIL

3.8.5 getExamLength()

This method has no parameters, therefore equivalence classes do not apply.

3.9 JSONBuilder.java

3.9.1 setExamCode(String examID)

Parameter	Equivalence Class		Representative
	ID	Description	
examID	1.1	x is a valid String	"UWEGIA"
	1.a	x is not a valid String	"Hi 1234"
	1.b	x is not a String	NULL

Test Cases

Test Case ID	examID	EQ Class(es) demonstrated	Expected Result
TC#1	"UWEGIA"	1.1	examID is set to be "UWEGIA"
TC#2	"THIS STRING 1234"	1.a	examID is set to be "THIS STRING 1234"
TC#3	12	1.b	FAIL
TC#4	NULL	1.c	FAIL

3.9.2 setExamName(String examName)

Parameter	Equivalence Class		Representative
	ID	Description	
examName	1.1	x is a String	"Exam 1 csc212"
	1.a	x is not a String	NULL

Test Cases

Test Case ID	examID	EQ Class(es) demonstrated	Expected Result
TC#1	"Exam 2 Sp2019"	1.1	examName is set to "Exam 2 Sp2019"
TC#2	{1,2,3}	1.a	FAIL
TC#3	Null	1.a	FAIL

3.9.3 buildAnswerKeyJSON(Student key)

Parameter	Equivalence Class		Representative
	ID	Description	

key	1.1 x is a valid Student 1.a x is not a valid Student 1.b x is not Student	new student() NULL

Test Cases

Test Case ID	key	EQ Class(es) demonstrated	Expected Result
TC#1	new student()	1.1	New json outputted to file
TC#2		1.a	FAIL
TC#3	Null	1.b	FAIL

3.9.4 buildByStudentJSON(List<Student> studentExams)

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
studentExams	1.1	x is a valid List<Students>	{student,student}
	1.a	x is not a valid List<Student>	{}
	1.b	x is not List<Student>	{“a,b,c,s”,””}
	1.c	x is not List	NULL

Test Cases

Test Case ID	studentExams	EQ Class(es) demonstrated	Expected Result
TC#1	{student,student}	1.1	New json outputted to file
TC#2	{}	1.a	FAIL
TC#3	{“a,b,c,s”,””}	1.b	FAIL
TC#4	Null	1.c	Fail
TC#4	“www.com”	1.c	Fail

3.9.5 getBystudent()

This method has no parameters, therefore equivalence classes do not apply.

3.9.6 buildByQuestion(List<Question> questions)

Parameter	Equivalence Class		Representative
	ID	Description	
questions	1.1	x is a valid List<Question>	{question,question}
	1.a	x is not a valid List<Question>	{}
	1.b	x is not List<Question>	{“a,b,c,s”,””}
	1.c	x is not List	NULL

Test Cases

Test Case ID	questions	EQ Class(es) demonstrated	Expected Result
TC#1	{question,question}	1.1	New json outputted to file
TC#2	{}	1.a	FAIL
TC#3	{“a,b,c,s”,””}	1.b	FAIL
TC#4	Null	1.c	Fail
TC#4	“www.com”	1.c	Fail

3.9.7 getByquestion()

This method has no parameters, therefore equivalence classes do not apply.

3.9.8 buildStatsJSON(Stats stats)

Parameter	Equivalence Class		Representative
	ID	Description	
stats	1.1	x is a valid Stat	new stat()
	1.a	x is not a valid Stat	
	1.b	x is notStat	NULL

Test Cases

Test Case ID	stats	EQ Class(es) demonstrated	Expected Result

TC#1	new stat()	1.1	New json outputted to file
TC#2		1.a	FAIL
TC#3	Null	1.b	FAIL

3.9.9 getBystats()

This method has no parameters, therefore equivalence classes do not apply.

3.9.10 buildMainpageJSON()

This method has no parameters, therefore equivalence classes do not apply.

3.9.11 resetAllJSON()

This method has no parameters, therefore equivalence classes do not apply.

3.10 LetterConverter.java

3.10.1 genLetterGrade(List<Float> examGrade)

Parameter	Equivalence Class		Representative
	ID	Description	
examGrade	1.1	x is a valid List<Float>	{4.0,55.7,80}
	1.a	x is not a valid List<Float>	{8000.99990,-472,.00001}
	1.b	x is not a List<Float>	{30,70,78,88,89,90,90,98}
	1.c	x is not List	NULL

Parameter	Boundary Values	Test Case ID(s)

	On	In	Off	
examGrade	100 0	99 1	101 -1	TC#6,7,8 TC#9,10,11

Test Cases

Test Case ID	examGrade	EQ Class(es) demonstrated	Expected Result
TC#1	{4.0,55.7,80}	1.1	Outputs String List of letter grades
TC#2	{8000.99990,-472,.00001}	1.a	Outputs String List of letter grades May cause issues later
TC#3	{30,70,78,88,89,90,90,98}	1.b	Outputs String List of letter grades if the input is a number type (ints, floats, doubles, longs) as long as it isnt too big. Chars and strings and other non number types cause a failure.
TC#4	“I AM STRING”	1.c	Fails
TC#5	Null	1.c	Fails
TC#6	{100}		Adds grade of A
TC#7	{99}		Adds grade of A

TC#8	{101}		Adds incorrect grade of F
TC#9	{0}		Adds grade of F
TC#10	{1}		Adds grade of F
TC#11	{-1}		Adds grade of F, may cause issues

Boundary Value Analysis

3.10.2 letterDistribution(String distribution)

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
distribution	1.1	x is a valid String	“30,70,78,88,89,90,90,98”
	1.a	x is not a valid String	“8000.99990,-472,.00001”
	1.b	x is not a String	{“99”,”100”}

Test Cases

Test Case ID	distribution	EQ Class(es) demonstrated	Expected Result
TC#1	“30,70,78,88,89,90,90,98”	1.1	Populates distribution
TC#2	“8000.99990,-472,.00001”	1.a	Populates distribution May cause issues

			later
TC#3	{30,70,78,88,89,90,90,98}	1.b	Fails.
TC#4	Null	1.b	Fails

3.10.3 getAboveForA()

This method has no parameters, therefore equivalence classes do not apply.

3.10.4 getAboveForB()

This method has no parameters, therefore equivalence classes do not apply.

3.10.5 getAboveForC()

This method has no parameters, therefore equivalence classes do not apply.

3.10.6 getAboveForD()

This method has no parameters, therefore equivalence classes do not apply.

3.11 Main.java

3.11.1 main(String[] args)

3.12 MiddlewareInterface.java

3.12.1 addStudentExam(String StudentData)

Parameter	Equivalence Class		Representive
	ID	Description	
StudentData	1.1	x is a valid String	"Valid String argument"
	1.a	x is not a valid String	"blargSS!#_(*@&#)(Q*#&$)(Q*#&$)(*"
	1.b	x is not a String	12345

Test Cases

Test Case ID	distribution	EQ Class(es) demonstrated	Expected Result
TC#1	"Valid String argument"	1.1	Adds a studentCreates a student from a database string using studentCreator.makeSecondPassStudent(StudentData)
TC#2	"blargSS!#_(*@&#)(Q*#&$)(Q*#&$)(*"	1.a	Trys to create a student with an invalid databaseString and fails.
TC#3	1234	1.b	Fails.
TC#4	Null	1.b	Fails

3.12.2 getGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.12.3 getExamGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.12.4 getStats()

This method has no parameters, therefore equivalence classes do not apply.

3.12.5 getExamLetterGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.12.6 examGradeIntegerConverter()

This method has no parameters, therefore equivalence classes do not apply.

3.12.7 getExamID()

This method has no parameters, therefore equivalence classes do not apply.

3.12.8 setExamName(String examName)

Parameter	Equivalence Class		Representative
	ID	Description	
studentData	1.1	x is a valid String	“Exam1”
	1.a	x is not a valid String	“8000.99990-472,.00001”
	1.b	x is not a String	{“99”,”100”}

Test Cases

Test Case ID	distribution	EQ Class(es) demonstrated	Expected Result
TC#1	“Exam1”	1.1	Sets examName to “Exam1”
TC#2	“8000.99990,-472,.00001”	1.a	Sets examName to “8000.99990,-472,.00001”
TC#3	{30,70,78,88,89,90,90,98}	1.b	Fails.
TC#4	Null	1.b	Fails

3.12.9 assignStudentGrades()

This method has no parameters, therefore equivalence classes do not apply.

3.12.10 makeJSON()

This method has no parameters, therefore equivalence classes do not apply.

3.13 Question.java

3.13.1 increment(String n)

Parameter	Equivalence Class		Representative
	ID	Description	
n	1.1	x is a valid String	"0,1,0,2,0,2,3,3,1,0,1,1,4,0"
	1.a	x is not a valid String	"8000.99990,-472,.00001"
	1.b	x is not a String	'f'

Test Cases

Test Case ID	distribution	EQ Class(es) demonstrated	Expected Result
TC#1	"0,1,0,2,0,2,3,3,1,0,1,1,4,0"	1.1	Populates the a,b,c,d,e ints
TC#2	"8000.99990,-472,.00001"	1.a	Does not populate any ints, will not crash May cause issues

			later
TC#3	{30,70,78,88,89,90,90,98}	1.b	Fails.
TC#4	Null	1.b	Fails

3.13.2 returnList()

This method has no parameters, therefore equivalence classes do not apply.

3.14 Stats.java

3.14.1 - 3.14.13

Methods 1-13 are all get methods that require no testing.

3.14.14 setScores(List<Integer> scores, List<String> key, List<Student> students)

3.14.14.1 Equivalence Classes

3.14.14.2 Test Cases

3.14.14.3 Boundary Value Analysis

3.14.15 examGenerator(int questionCount)

The examGenerator method is used for testing the functionality of the class and therefore requires no testing. It is recommended that this is removed by the final iteration of the project.

3.14.16 examGrader(List<List<String>> exams, List<String> answerKey, List<Integer> weight)

The examGrader method is used for testing the functionality of the class and therefore requires no testing. It is recommended that this is removed by the final iteration of the project.

3.14.17 weightGenerator(int questionCount)

The weightGenerator method is never used.

3.14.18 meanDecimal(List<BigDecimal> scores)

3.14.18.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are Big Decimals	`List<BigDecimal> scores = [4,34,5,3] as BigDecimal[]`
	1.a	Not a bigDecimal list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6

3.14.18.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[4,34,5,3]	1.1	11.5
TC#2	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#3	6	1.b	Exception thrown

3.14.18.3 Boundary Value Analysis

The method meanDecimal is a lower level function that has no specification boundary constraints so this test is unnecessary.

3.14.19 meanInteger(List<Integer> scores)

3.14.19.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are Integers	`List<BigDecimal> scores = [4,34,5,3] as Integer[]`
	1.a	Not an Integer list	{"","whoops","string"}
	1.b	Not a valid list	6

3.14.19.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[4,34,5,3]	1.1	11.5
TC#2	["","whoops","string"]	1.a	Exception thrown
TC#3	6	1.b	Exception thrown

3.14.19.3 Boundary Value Analysis

The method meanInteger is a lower level function that has no specification boundary constraints so this test is unnecessary.

3.14.20 overallVariance(List<Integer> scores)

3.14.20.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are Integers	[4,34,5,3]
	1.2	Sample size is 1	0
	1.a	Not an Integer list	[“”,”whoops”,”string”]
	1.b	Not a valid list	6
	1.c	Sample size is 0	[]

3.14.20.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[4,34,5,3]	1.1	169.25
TC#2	[4]	1.2	0
TC#3	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#4	6	1.b	Exception thrown
TC#5	[]	1.c	Exception thrown

3.14.20.3 Boundary Value Analysis

Boundary Value Analysis for this method was simple and therefore done above.

3.14.21 overallStandardDeviation(List<Integer> scores)

This method is tested through 3.14.20 and 3.14.40, and will fail or succeed based on them. However, this method is never used so that is irrelevant.

3.14.22 gradesByQuestion(List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.22.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
exams	1.1	x is a list of lists of Strings of a-e of the size 1, all of equal length	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}
	1.2	x is a list of lists of strings where some are null or -1	{{a,b,c,d,e,c,null,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,-1,c,b,a,c},{a,a,c,d,e,c,-1,null,d}}
	1.a	x is a list of lists of strings of unequal lengths	{{a,b,c,d,e,c,b,a,d},{a,b,c,d,e,c,b,a,d,a,a,a,a},{{a,b,c,d,e,c,b,a,d}},{a,b,c}}
	1.b	x is a list of lists of strings where some are not a-e	{{a,b,c,d,e,c,b,a,z},{a,a,z,d,c,c,z,a,b},{b,b,c,g,e,c,b,a,c},{a,a,c,d,e,c,d,c,l}}
	1.c	x is a list of lists of strings where some strings are longer than 1 char	{{a,b,c,d,e,c,b,a,dc},{a,ab,c,d,c,c,b,a,ba},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,dd}}
	1.d	x is not a list of lists	{6, 8}
	1.e	x is null	null

answerKey	2.1 x is a list of answer strings of a-e of the size 1-5 2.a x is a list of answer strings where one is larger than 5 chars 2.b X is a list of answer strings where a letter other than a-e is present 2.c X is null 2.d X is not a list of strings	{a,b,c,d,ae,c,b,a, abcde} {abcdea} {a,v,b,f} null {4, 2, 7}
weight	This is always a list of 1's and is of outdated usage and therefore ignorable	----------

3.14.22.2 Test Cases

Test Cases 2-9 are the same as in 3.14.36.

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	{{1,1,1,1,1,1,1,1,1,1}, {1,0,1,1,0,1,1,1,1}, {0,1,1,0,1,1,1,1,1} ,{1,0,1,1,1,1,0,0,1}}

TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	{{0,0,0,0,0,0,0,0,1},{0,0,0,0,0,0,0,0,1},{0,0,0,0,0,0,0,0,1},{0,0,0,0,0,1,0,0,1}}
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	{{0,0,0,0,0,0,0,0,1},{1,0,0,0,1,0,0,1,1},{0,0,0,0,0,0,0,0,1},{0,0,0,0,0,1,0,0,1}}
TC#12	{{c,c,c,d,e,c,b,b,e},{a,b,d,c,e,a,a,c,e},{a,a,c,d,e,c,d,d,b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	{{0,0,1,1,1,1,1,0,1},{1,1,0,0,1,0,0,0,1},{1,0,1,1,1,0,0,0,1},{1,1,1,1,1,1,1,1,1}}
TC#13	{{a,b,c,d,e,c,error,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,-1,c,b,a,c},{a,a,c,d,e,c,-1,error,error}}	{a,b,c,d,ae,c,b,a,abcde}	1.2, 2.1	{{1,1,1,1,1,1,0,1,1},{1,0,1,1,0,1,1,1,1},{0,1,1,0,0,1,1,1,1},{1,0,1,1,1,1,0,0,0}}

3.14.23 meanByQuestion(List<List<Integer>> gradesByQuestion)

THIS IS THE SAME THING AS PROPORTIONPASSINGBYTEST

3.14.24 differenceFromMean(List<List<Integer>> gradesByQuestion, List<BigDecimal> meanByQuestion)

THIS IS THE SAME THING AS PROPORTIONFAILINGBYTEST

3.14.25 squaredData(List<List<BigDecimal>> input)

3.14.25.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are Integers and of equal length	[[4, 34, 5, 3], [17, 99, 191, 42], [45, 19, 81, 111]]
	1.a	Not an Integer list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6
	1.c	Lists in List are unequal lengths	[[4, 34, 5, 3], [17, 191, 42], [45, 111]]

3.14.25.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[[4, 34, 5, 3], [17, 99, 191, 42], [45, 19, 81, 111]]	1.1	[[16, 1156, 25, 9], [289, 9801, 36481, 1764], [2025, 361, 6561, 12321]]
TC#2	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#3	6	1.b	Exception thrown
TC#4	[[4, 34, 5, 3], [17, 191, 42], [45, 111]]	1.c	Out of bounds Exception thrown

3.14.26 additionOfData(List<List<BigDecimal>> input)

Testing of the method is suspended until 3.14.29 works.

3.14.27 divisionOfData(List<BigDecimal> input, int divisor)

Testing of the method is suspended until 3.14.29 works.

3.14.28 summationOfList(List<BigDecimal> input)

Testing of the method is suspended until 3.14.29 works.

3.14.29 cronbachsAlpha(List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.29.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
exams	1.1	x is a list of lists of Strings of a-e of the size 1, all of equal length	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}
	1.a	x is a list of lists of strings of unequal lengths	{{a,b,c,d,e,c,b,a,d},{a,b,c,d,e,c,b,a,d,a,a,a,a},{{a,b,c,d,e,c,b,a,d}},{a,b,c}}
	1.b	x is a list of lists of strings where some are not a-e	{{a,b,c,d,e,c,b,a,z},{a,a,z,d,c,c,z,a,b},{b,b,c,g,e,c,b,a,c},{a,a,c,d,e,c,d,c,l}}
	1.c	x is a list of lists of strings where some strings are longer than 1 char	{{a,b,c,d,e,c,b,a,dc},{a,ab,c,d,c,c,b,a,ba},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,dd}}
	1.d	x is not a list of lists	
	1.e	x is null	

		{6, 8} --- null
answerKey	2.1 x is a list of answer strings of a-e of the size 1-5 --- 2.a x is a list of answer strings where one is larger than 5 chars --- 2.b X is a list of answer strings where a letter other than a-e is present --- 2.c X is null --- 2.d X is not a list of strings	{a,b,c,d,ae,c,b,a, abcde} --- {abcdea} --- {a,v,b,f} --- null --- {4, 2, 7}
weight	This is always a list of 1's and is of outdated usage and therefore ignorable	----------

3.14.29.2 Test Cases

TC#'s 2-9 are the same as in sections 3.14.36, and as such can be found there.

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0

	a,c,d,e,c,d,c,d}}			
TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0.5625
TC#12	{{c,c,c,d,e,c,b,b, e},{a,b,d,c,e,a,a,c, e},{a,a,c,d,e,c,d,d, b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0.642857

3.14.29.3 Boundary Value Analysis

3.14.30 lowestScore(List<Integer> scores)

3.14.30.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are integers	[56, 24, 65, 89]
	1.a	Not a Integer list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6

3.14.30.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[56, 24, 65, 89]	1.1	24
TC#2	[56, 24.5, 65, 89]	1.1	24.5
TC#3	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#4	6	1.b	Exception thrown

3.14.30.3 Boundary Value Analysis

No boundary value analysis needed for this method.

3.14.31 highestScore(List<Integer> scores)

3.14.31.1 Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
scores	1.1	x is a valid List of scores that are Integers	[56, 24, 65, 89]
	1.a	Not an Integer list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6

3.14.31.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[56, 24, 65, 89]	1.1	89
TC#2	[56, 24, 65, 89.5]	1.1	89.5

TC#3	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#4	6	1.b	Exception thrown

3.14.31.3 Boundary Value Analysis

No boundary value analysis needed for this method.

3.14.32 median(List<Integer> scores)

3.14.32.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid even List of scores that are Integers	[56, 24, 65, 89]
	1.2	X is a valid odd list of scores that are Integers	[56, 59, 24, 65, 89]
	1.a	Not an Integer list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6

3.14.32.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[56, 24, 65, 89]	1.1	60.5
TC#2	[56, 59, 24, 65, 89]	1.2	59
TC#3	[“”,”whoops”,”string”]	1.a	Exception thrown

TC#4	6	1.b	Exception thrown

3.14.32.3 Boundary Value Analysis

3.14.33 rangeOfScores(List<Integer> scores)

3.14.33.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
scores	1.1	x is a valid List of scores that are Integers	[56, 24, 65, 89]
	1.a	Not an Integer list	{“”,”whoops”,”string”}
	1.b	Not a valid list	6

3.14.33.2 Test Cases

Test Case ID	scores	EQ Class(es) demonstrated	Expected Result
TC#1	[56, 24, 65, 89]	1.1	65
TC#2	[56, 24, 65, 89.5]	1.1	65.5
TC#3	[56, 24, 65, 89.5, 77]	1.1	65.5
TC#4	[“”,”whoops”,”string”]	1.a	Exception thrown
TC#5	6	1.b	Exception thrown

3.14.33.3 Boundary Value Analysis

3.14.34 proportionPassingByQuestion(List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.34.1 Equivalence Classes

Equivalence classes are the same as section 3.14.36

3.14.34.2 Test Cases

TC#'s 2-9 are the same as in section 3.14.36

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.75,0.5,1,0.75,0.75,1,0.75,0.75,0]
TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0,0,0,0,0,0.25,0,0,0]
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.25,0,0,0,0.25,0.25,0,0.25,0]
TC#12	{{c,c,c,d,e,c,b,b,e},{a,b,d,c,e,a,a,c,e},{a,a,c,d,e,c,d,d,b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.75,0.5,0.75,0.75,1,0.5,0.5,0.25,0]

3.14.35 proportionFailingByQuestion(List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.35.1 Equivalence Classes

These EQ are the same as in section 3.14.36

3.14.35.2 Test Cases

TC#'s 2-9 are the same as in section 3.14.36

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.25,0.5,0,0.25,0.25,0,0.25,0.25,1]
TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[1,1,1,1,1,0.75,1,1,1]
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.75,1,1,1,0.75,0.75,1,0.75,1]
TC#12	{{c,c,c,d,e,c,b,b,e},{a,b,d,c,e,a,a,c,e},{a,a,c,d,e,c,d,d,b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	[0.25,0.5,0.25,0.25,0,0.5,0.5,0.75,1]

3.14.36 kuderRichardson21 (List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.36.1 Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
exams	1.1	x is a list of lists of Strings of a-e of the size 1, all of equal length	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}
	1.a	x is a list of lists of strings of unequal lengths	{{a,b,c,d,e,c,b,a,d},{a,b,c,d,e,c,b,a,d,a,a,a,a},{{a,b,c,d,e,c,b,a,d}},{a,b,c}}
	1.b	x is a list of lists of strings where some are not a-e	{{a,b,c,d,e,c,b,a,z},{a,a,z,d,c,c,z,a,b},{b,b,c,g,e,c,b,a,c},{a,a,c,d,e,c,d,c,l}}
	1.c	x is a list of lists of strings where some strings are longer than 1 char	{{a,b,c,d,e,c,b,a,dc},{a,ab,c,d,c,c,b,a,ba},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,dd}}
	1.d	x is not a list of lists	{6, 8}
	1.e	x is null	null
answerKey	2.1	x is a list of answer strings of a-e of the size 1-5	{a,b,c,d,ae,c,b,a, abcde}
	2.a	x is a list of answer strings where one is larger than 5 chars	{abcdea}
	2.b	X is a list of answer strings where a letter other than a-e is present	{a,v,b,f}
			null
			{4, 2, 7}

	2.c X is null 2.d X is not a list of strings	
weight	This is always a list of 1's and is of outdated usage and therefore ignorable	----------

3.14.36.2 Test Cases

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	0
TC#2	{{a,b,c,d,e,c,b,a,d},{a,b,c,d,e,c,b,a,d,a,a,a,a},{{a,b,c,d,e,c,b,a,d}},{a,b,c}}	{a,b,c,d,ae,c,b,a,abcde}	1.a	Exception Thrown
TC#3	{{a,b,c,d,e,c,b,a,z},{a,a,z,d,c,c,z,a,b},{b,b,c,g,e,c,b,a,c},{a,a,c,d,e,c,d,c,l}}	{a,b,c,d,ae,c,b,a,abcde}	1.b	Exception thrown
TC#4	{{a,b,c,d,e,c,b,a,dc},{a,ab,c,d,c,c,b,a,ba},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,dd}}	{a,b,c,d,ae,c,b,a,abcde}	1.c	Exception thrown
TC#5	{6,8}	{a,b,c,d,ae,c,b,a,abcde}	1.d	Exception thrown
TC#6	null	null	1.e, 2.c	Exception

				thrown
TC#7	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{4, 2, 7}	2.d	Exception thrown
TC#8	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{abcdea}	2.a	Exception thrown
TC#9	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,v,b,f,aeg,l,b,a,n}	2.b	Exception thrown
TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0
TC#12	{{c,c,c,d,e,c,b,b, e},{a,b,d,c,e,a,a,c, e},{a,a,c,d,e,c,d,d, b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a, abcde}	1.1, 2.1	0.482142857

3.14.37 kuderRichardson20 (List<List<String>> exams, List<String> answerKey, List<Integer> weight)

3.14.37.1 Equivalence Classes

These equivalence classes are the same as in section 3.14.36

3.14.37.2 Test Cases

TC#'s 2-9 are identical to section 3.14.36 and can be found there

Test Case ID	exams	answerKey	EQ Class(es) demonstrated	Expected Result
TC#1	{{a,b,c,d,e,c,b,a,d},{a,a,c,d,c,c,b,a,b},{b,b,c,b,e,c,b,a,c},{a,a,c,d,e,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	0
TC#10	{{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	0
TC#11	{{c,a,b,b,b,a,d,d,d},{a,a,a,a,a,a,a,a,a},{c,a,b,b,b,a,d,d,d},{c,a,a,b,b,c,d,c,d}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	0.5625
TC#12	{{c,c,c,d,e,c,b,b,e},{a,b,d,c,e,a,a,c,e},{a,a,c,d,e,c,d,d,b},{a,b,c,d,a,c,b,a,e}}	{a,b,c,d,ae,c,b,a,abcde}	1.1, 2.1	0.642857

3.14.38 questionFrequency(List<Integer> scores)

3.14.38.1 Equivalence Classes

Exams is the same exams in section 3.14.36

3.14.28.2 Test Cases

Some of the other test cases from 3.14.36 are used as well.

Test Case ID	exams	EQ Class(es) demonstrated	Expected Result
TC#1	[["0","1","2","3","4","2","1","0","3"],["0","0","2","3","2","2","1","0","1"],["1","1","2","1","4","2","1","0","2"],["0","0","2","3","4","2","3","2","3"]]	1.1, 2.1	[["3", "1", "0", "0", "0"], ["2", "2", "0", "0", "0"], ["0", "0", "4", "0", "0"], ["0", "1", "0", "3", "0"], ["0", "0", "1", "0", "3"], ["0", "0", "4", "0", "0"], ["0", "3", "0", "1", "0"], ["3", "0", "1", "0", "0"], ["0", "1", "1", "2", "0"]]
TC#10	[["2", "0", "1", "1", "1", "0", "3", "3", "3"], ["2", "0", "1", "1", "1", "0", "3", "3", "3"], ["2", "0", "1", "1", "1", "0", "3", "3", "3"], ["2", "0", "0", "1", "1", "2", "3", "2", "3"]]	1.1, 2.1	[["0", "0", "4", "0", "0"], ["4", "0", "0", "0", "0"], ["1", "3", "0", "0", "0"], ["0", "4", "0", "0", "0"], ["0", "4", "0", "0", "0"], ["3", "0", "1", "0", "0"], ["0", "0", "0", "4", "0"], ["0", "0", "1", "3", "0"], ["0", "0", "0", "4", "0"]]
TC#11	[["2", "0", "1", "1", "1", "0", "3", "3", "3"], ["0", "0", "0", "0", "0", "0", "0", "0", "0"], ["2", "0", "1", "1", "1", "0", "3", "3", "3"], ["2", "0", "0", "1", "1", "2", "3", "2", "3"]]	1.1, 2.1	[["1", "0", "3", "0", "0"], ["4", "0", "0", "0", "0"], ["2", "2", "0", "0", "0"], ["1", "3", "0", "0", "0"], ["1", "3", "0", "0", "0"], ["3", "0", "1", "0", "0"], ["1", "0", "0", "3", "0"], ["1", "0", "1", "2", "0"], ["1", "0", "0", "3", "0"]]
TC#12	[["2", "2", "2", "3", "4", "2", "1", "1", "4"], ["0", "1", "3", "2", "4", "0", "0", "2", "4"], ["0", "0", "3", "3", "4", "2", "3", "3", "1"], ["0", "1", "2", "3", "0", "2", "1", "0", "4"]]	1.1, 2.1	[["3", "0", "1", "0", "0"], ["1", "2", "1", "0", "0"], ["0", "0", "2", "2", "0"], ["0", "0", "1", "3", "0"], ["1", "0", "0", "0", "3"], ["1", "0", "3", "0", "0"], ["1", "2", "0", "1", "0"], ["1", "1", "1", "1", "0"], ["0", "1", "0", "0", "3"]]

3.14.39 percentiles(List<Integer> scores)

There is no documentation anywhere of what this is supposed to do, exactly. I can't tell what so ever.

3.14.40 squareRoot(List<Integer> scores)

3.14.41 quartiles(List<Integer> scores)

This method is never used.

3.14.42 babylonianSqrt(BigDecimal s, BigDecimal threshold)

This Method is never used.

3.14.43 babylonianSqrt(BigDecimal s)

This Method is never used.

3.16 Student.java

3.16.1 initializeStudentFromStudentData(List<String> studentData)

Parameter	Equivalence Class		Representative
	ID	Description	

studentData	1.1 x is a valid List<String> 1.a x is not a valid List<Strings> 1.b x is not a List<Strings> 1.c x is not a List	{“Quinn Riley”} {“12345567890-=-09876543234 5678987654321”,”@@@#$$#”} {1,2,3} NULL

Test Cases

Test Case ID	studentData	EQ Class(es) demonstrated	Expected Result
TC#1	{“Quinn Riley”}	1.1	Fills out any info about the student and their exam it can find
TC#2	{“12345567890-=-09 8765432345678987 654321”,”@@@#$$ #”}	1.a	FAIL
TC#3	{1,2,3}	1.b	FAIL
TC#4	12	1.c	FAIL
TC#5	NULL	1.c	FAIL

3.16.2 initializeStudentFromDatabaseString(String databaseString)

Parameter	Equivalence Class		Representative
	ID	Description	
databaseString	1.1	x is a valid String	“Quinn Riley”
	1.a	x is not a valid String	“12345567890-=-09876543234 5678987654321@@@#$$#”
	1.b	x is not a String	NULL

Test Cases

Test Case ID	studentData	EQ Class(es) demonstrated	Expected Result
TC#1	{“Quinn Riley”}	1.1	Fills out any info about the student and their exam it can find
TC#2	{“12345567890-=-09 8765432345678987 654321”,”@@@#$$ #”}	1.a	FAIL
TC#3	{1,2,3}	1.b	FAIL
TC#4	12	1.c	FAIL
TC#5	NULL	1.c	FAIL

3.16.3 initializeAnswers(List<String> answers)

Parameter	Equivalence Class		Representative
	ID	Description	
answers	1.1	x is a valid List<String>	{“123”,”2”,”345”}
	1.a	x is not a valid List<Strings>	{“12345567890-=-09876543234 5678987654321”,”@@@#$$#”}
	1.b	x is not a List<Strings>	{1,2,3}
	1.c	x is not a List	NULL

Test Cases

Test Case ID	studentData	EQ Class(es) demonstrated	Expected Result
TC#1	{“123”,”2”,”345”}	1.1	Fills out the answers list in the class
TC#2	{“12345567890-=-09 8765432345678987 654321”,”@@@#$$ #”}	1.a	Fills out answers list with this nonsense information, will cause issues later
TC#3	{1,2,3}	1.b	FAIL
TC#4	12	1.c	FAIL
TC#5	NULL	1.c	FAIL

3.16.4 findName()

This method has no parameters, therefore equivalence classes do not apply.

3.16.5 findSex()

This method has no parameters, therefore equivalence classes do not apply.

3.16.6 findGrade()

This method has no parameters, therefore equivalence classes do not apply.

3.16.7 findBirthday()

This method has no parameters, therefore equivalence classes do not apply.

3.16.8 findId()

This method has no parameters, therefore equivalence classes do not apply.

3.16.9 findCode()

This method has no parameters, therefore equivalence classes do not apply.

3.16.10 genJsonData()

This method has no parameters, therefore equivalence classes do not apply.

3.16.11 alphaConverter(String entry)

Parameter	Equivalence Class		Representative
	ID	Description	

entry	1.1 x is a valid String 1.a x is not a valid String 1.b x is not a String	“25” “Holla” NULL

Test Cases

Test Case ID	entry	EQ Class(es) demonstrated	Expected Result
TC#1	“1”	1.1	Converts to the letter A
TC#2	{“12345567890-=-09 8765432345678987 654321”,”@@@#$$ #”}	1.a	Trys and fails to convert, returns an empty string, “”
TC#3	3	1.b	FAIL
TC#4	NULL	1.b	FAIL

3.16.12 setExamGrade(float grade)

Parameter	Equivalence Class		Representative
	ID	Description	

grade	1.1 x is a valid Float 1.a x is not a valid Float 1.b x is not a Float	25.003 10003.9999999 NULL

Test Cases

Test Case ID	grade	EQ Class(es) demonstrated	Expected Result
TC#1	1	1.1	Sets grade to 1
TC#2	10003.9999999	1.a	Sets grade to 10003.9999999, Which is invalid and will cause issues, most likely in displaying or storing the information
TC#3	a	1.b	FAIL
TC#4	NULL	1.b	FAIL

3.16.13 setLetterGrade(String letterGrade)

Parameter	Equivalence Class		Representative
	ID	Description	

letterGrade	1.1 x is a valid String 1.a x is not a valid String 1.b x is not a String	"C" "Trying to set grade to this" NULL

Test Cases

Test Case ID	letterGrade	EQ Class(es) demonstrated	Expected Result
TC#1	"C"	1.1	Sets letterGrade to C
TC#2	"Trying to set grade to this"	1.a	Sets letterGrade to Trying to set grade to this, Which is invalid and will cause issues, most likely in displaying or storing the information
TC#3	1234	1.b	FAIL
TC#4	NULL	1.b	FAIL

3.16.14 getName()

This method has no parameters, therefore equivalence classes do not apply.

3.16.15 getSex()

This method has no parameters, therefore equivalence classes do not apply.

3.16.16 getBirthday()

This method has no parameters, therefore equivalence classes do not apply.

3.16.17 getGrade()

This method has no parameters, therefore equivalence classes do not apply.

3.16.18 getId()

This method has no parameters, therefore equivalence classes do not apply.

3.16.19 getCode()

This method has no parameters, therefore equivalence classes do not apply.

3.16.20 getAnswers()

This method has no parameters, therefore equivalence classes do not apply.

3.16.21 getLetterGrade()

This method has no parameters, therefore equivalence classes do not apply.

3.16.22 getExamGrade()

This method has no parameters, therefore equivalence classes do not apply.

3.17 StudentCreater.java

3.17.1 makeStudent(String[][] answers, String[] StudentData)

Parameter	Equivalence Class		Representative
	ID	Description	

letterGrade	1.1 x is a valid String 1.a x is not a valid String 1.b x is not a String	"C" "Trying to set grade to this" NULL

Test Cases

Test Case ID	letterGrade	EQ Class(es) demonstrated	Expected Result
TC#1	"C"	1.1	Sets letterGrade to C
TC#2	"Trying to set grade to this"	1.a	Sets letterGrade to Trying to set grade to this, Which is invalid and will cause issues, most likely in displaying or storing the information
TC#3	1234	1.b	FAIL
TC#4	NULL	1.b	FAIL

3.17.2 makeStudent(String[][] answersFront, String[][] answersBack, String[] StudentData)

3.17.3 makeSecondPassStudent(String studentData)

3.19 Main.java

3.19.1 main(String[] args)

3.20 OrientTool.java

3.20.1 orient()

This method has no parameters, therefore equivalence classes do not apply.

3.20.2 rotateClockwise90(BufferedImage src)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
src	1.1	x is a valid BufferedImage	New BufferedImage()
	1.a	x is not a BufferedImage	[{"IMASTRING"}{"Yolo"}{"Memes"}] NULL

Test Cases

Test Case ID	src	EQ Class(es) demonstrated	Expected Result
TC#1	BufferedImage()	1.1	New Student created and added to the studentExams ArrayList
TC#2	{{“IMASTRING”}{“Yolo”}{“Memes”}}	1.a	FAIL
TC#3	NULL	1.a	FAIL

3.21 Utilities.java

3.21.1 RetrieveEmails()

This method has no parameters, therefore equivalence classes do not apply.

3.21.2 changeDirectory()

This method has no parameters, therefore equivalence classes do not apply.

3.21.3 pdf2jpeg()

This method has no parameters, therefore equivalence classes do not apply.

3.21.4 orient()

This method has no parameters, therefore equivalence classes do not apply.

3.21.5 rotateClockwise90(BufferedImage src)

Can be removed

```
//leagacy version no need to test, OrientTool took over this method that is the new one
```

3.21.6 runScanner()

This method has no parameters, therefore equivalence classes do not apply.

3.21.7 main(String[] args)

3.21.8 getAnswers(int[][] a)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
a	1.1	x is a valid int[][]	
	1.a	x is not a valid int[][]	
	1.b	x is not an int[][]	NULL

Test Cases

Test Case ID	a	EQ Class(es) demonstrated	Expected Result
TC#1		1.1	New Student created and added to the studentExams ArrayList
TC#2		1.a	FAIL
TC#3	"hi"	1.b	FAIL
TC #4	NULL	1.b	Fail

3.21.9 multi(List<String> a)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
a	1.1	x is a valid List<String>	
	1.a	x is not a valid List<String>	
	1.b	x is not a List<String>	{1,3,5,6}
	1.c	x is not a List	NULL

Test Cases

Test Case ID	a	EQ Class(es) demonstrated	Expected Result
TC#1		1.1	New Student created and added to the studentExams ArrayList
TC#2		1.a	FAIL
TC#3	{1,3,5,6}	1.b	FAIL
TC#4	“hi”	1.c	FAIL
TC #5	NULL	1.c	FAIL

3.21.10 oldmulti(String[] a)

//OLD METHOD WILL NOT TEST, need to ask if still in use, if not recommend deletion

3.21.11 gradeCSV(String name, List<String> ids, List<Float> grades)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	

name	1.1 x is a valid 1.a x is not a valid 1.b x is not a	NULL
ids	1.1 x is a valid 1.a x is not a valid 1.b x is not a	NULL
grades	1.1 x is a valid 1.a x is not a valid 1.b x is not a	NULL

Test Cases

Test Case ID	names	ids	grades	EQ Class(es) demonstrated	Expected Result

TC#1				1.1	
TC#2				1.a	FAIL
TC#3				1.b	FAIL
TC #4				1.b	Fail

3.21.12 sendEmailProcessed(String address, String csvPath)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
address	1.1	x is a valid	
	1.a	x is not a valid	
	1.b	x is not a	NULL

csvPath	1.1 x is a valid 1.a x is not a valid 1.b x is not a	NULL

Test Cases

Test Case ID	address	csvPath	EQ Class(es) demonstrated	Expected Result
TC#1			1.1	
TC#2			1.a	FAIL
TC#3			1.b	FAIL
TC #4			1.b	Fail

3.21.13 sendEmailRecieved(String address)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	

address	1.a x is a valid String 1.b x is a valid String, but not an email 1.c x is not a valid String 1.d x is not a String	“bastian.tenbergen@oswego.edu” “bastian.tenbergen@oz” “” NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	“bastian.tenbergen@oswego.edu”	1.a	
TC#2	“bastian.tenbergen@oz”	1.b	FAIL
TC#3	“”	1.c	FAIL
TC #4	null	1.d	Fail

3.21.14 deleteAllFiles()

This method has no parameters, therefore equivalence classes do not apply.

3.22 AnswerKey.java

3.22.1 getQuestionId()

This method has no parameters, therefore equivalence classes do not apply.

3.22.2 getAnswerKey()

This method has no parameters, therefore equivalence classes do not apply.

3.22.3 setQuestionId(int questionId)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
questionId	1.1	x is a valid Integer	12
	1.a	x is not a valid Integer	-1
	1.b	x is not a Integer	NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	12	1.1	Pass
TC#2	-1	1.a	FAIL
TC#3	1	1.a	Pass

TC#4	199	1.a	Pass
TC#5	200	1.a	Pass
TC#6	201	1.a	FAIL
TC#7	null	1.b	FAIL
TC#8	0	1.a	Pass

Parameter	Boundary Values On | In | Off	Test Case ID(s)
questionId	200 199 201 0 1 -1	TC#4,5,6 TC#2,3,8

3.22.4 setAnswerKey(String[] answerKey)

Equivalence Classes

Parameter	Equivalence Class ID	Description	Representative
answerKey	1.1	x is a valid String[]	[“a”,”b”,”c”,”a”]
	1.a	x is not a valid String[]	new String[20]
	1.b	x is not a String[]	NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result

TC#1	[“a”,”b”,”c”,”a”]	1.1	
TC#2	[]	1.a	FAIL
TC#3	null	1.b	FAIL

3.23 Exam.java

3.23.1 getExamName()

This method has no parameters, therefore equivalence classes do not apply.

3.23.1 setExamName(String examName)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
examName	1.1	x is a valid String	“Bastian’s 1st Exam”
	1.a	x is not a valid String	“”
	1.b	x is not a String	NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result

TC#1	“Bastian’s 1st Exam”	1.1	
TC#2	“”	1.a	FAIL
TC#3		1.b	FAIL

3.23.1 getExamCode()

This method has no parameters, therefore equivalence classes do not apply.

3.23.1 setExamCode(String examCode)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	
examName	1.1	x is a valid String	“BTYSR”
	1.a	x is a valid String, also a name	“CHRIS”
	1.b	x is not a valid String	“”
	1.c	x is not a String	NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	“BTYSR”	1.1	

TC#2	“CHRIS”	1.a	Success, Failure possible when student enters name in exam
TC#3	“”	1.b	FAIL
TC#4	null	1.c	FAIL

3.23.1 getTimestamp()

This method has no parameters, therefore equivalence classes do not apply.

3.23.1 setTimestamp(String timestamp)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
examName	1.1	x is a valid String	“05031230”
	1.a	x is not a valid String	“”
	1.b	x is not a String	NULL, 12

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	“05031230”	1.1	

TC#2	“”	1.a	FAIL
TC#3	Null, 12	1.b	FAIL

3.24 Professor.java

3.24.1 getExams()

This method has no parameters, therefore equivalence classes do not apply.

3.24.2 setExams(List<Exam> list)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
list	1.1	x is a valid exam list	new Exam()
	1.a	x is not a valid exam list	
	1.b	x is not a list	NULL, “hello”, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1		1.1	
TC#2	new Exam()	1.a	FAIL
TC#3	NULL, “hello”, 15	1.b	FAIL

3.24.3 getExamName()

This method has no parameters, therefore equivalence classes do not apply.

3.24.4 setExamName(String name)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
name	1.1	x is a valid String	“Bastian’s Funhouse”
	1.a	x is not a valid String	“”
	1.b	x is not a String	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result

TC#1	“Bastian’s Funhouse”	1.1	
TC#2	“”	1.a	FAIL
TC#3	NULL, 15	1.b	FAIL

3.24.5 getEmail()

This method has no parameters, therefore equivalence classes do not apply.

3.24.5 setEmail(String email)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
name	1.1	x is a valid String	“Bastian.tenbergen@oswego.edu”
	1.a	x is a valid String, not an email	“helloImDilbert”
	1.b	x is not a String	NULL

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	“Bastian.tenbergen @oswego.edu”	1.1	

TC#2	“helloImDilbert”	1.a	FAIL
TC#3	null	1.b	FAIL

3.24.5 addExam(Exam e)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
name	1.1	x is a valid Exam	new Exam(String code, String name, String time)
	1.a	x is not a valid Exam	new Exam()
	1.b	x is not a String	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new Exam(String code, String name, String time)	1.1	
TC#2	new Exam()	1.a	FAIL
TC#3	NULL, 15	1.b	FAIL

53.25 AnswerKeyServlet.java

3.25.1 doGet(HttpServletRequest request, HttpServletResponse response)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
request	1.1	x is a valid HttpServletRequest	new HttpServletRequest()
	1.a	x is not a valid HttpServletRequest	new HttpServletResponse()
	1.b	x is not a HttpServletRequest	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new HttpServletRequest()	1.1	
TC#2	new HttpServletResponse()	1.a	FAIL
TC#3	Null, 15	1.b	FAIL

Parameter	Equivalence Class ID	Equivalence Class Description	Representative

response	1.1	x is a valid HttpServletResponse	new HttpServletResponse()
	1.a	x is not a valid HttpServletResponse	new HttpServletRequest()
	1.b	x is not a HttpServletResponse	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new HttpServletResponse()	1.1	
TC#2	new HttpServletRequest()	1.a	FAIL
TC#3	Null, 15	1.b	FAIL

3.25.2 doPost(HttpServletRequest request, HttpServletResponse response)

Equivalence Classes

Parameter	Equivalence Class		Representative
	ID	Description	

request	1.1 x is a valid HttpServletRequest 1.a x is not a valid HttpServletRequest 1.b x is not a HttpServletRequest	new HttpServletRequest() new HttpServletResponse() NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new HttpServletRequest()	1.1	
TC#2	new HttpServletResponse()	1.a	FAIL
TC#3	Null, “string”, 15	1.b	FAIL

Parameter	Equivalence Class		Representative
	ID	Description	
response	1.1	x is a valid HttpServletResponse	new HttpServletResponse()
	1.a	x is not a valid HttpServletResponse	new HttpServletRequest()
	1.b	x is not a HttpServletResponse	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new HttpServletResponse()	1.1	
TC#2	new HttpServletRequest()	1.a	FAIL
TC#3	NULL, 15	1.b	FAIL

3.25.3 buildKeys()

This method has no parameters, therefore equivalence classes do not apply.

3.25.4 getKey(Map<String, String> keyList, Object val)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
keyList	1.1	x is a valid String map	new Map<String, String>()
	1.a	x is not a valid String map	
	1.b	x is not a String map	NULL, 15, new String("Ryan")

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new Map<String, String>()	1.1	
TC#2		1.a	FAIL

TC#3	NULL, 15, new String("Ryan")	1.b	FAIL

Parameter	Equivalence Class		Representative
	ID	Description	
Object val	1.1	x is a valid String	"KJFTA"
	1.a	x is not a valid String	""
	1.b	x is not a String	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	"KJFTA"	1.1	
TC#2	""	1.a	FAIL
TC#3	Null, 15	1.b	FAIL

3.26 ExamServlet.java

3.26.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.26.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.27 NameChangeServlet.java

3.27.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.27.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.28 QuestionServlet.java

3.28.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.28.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.29 ResultServlet.java

3.29.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.29.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.29.3 response(HttpServletResponse resp, String msg)

Equivalence Classes

Parameter	Equivalence Class ID	Equivalence Class Description	Representative
resp	1.1	x is a valid HttpServletResponse	new HttpServletResponse()
	1.a	x is not a valid HttpServletResponse	new HttpServletRequest()
	1.b	x is not a HttpServletResponse	NULL, 15

Test Cases

Test Case ID	Scores	EQ Class(es) demonstrated	Expected Result
TC#1	new HttpServletResponse()	1.1	
TC#2	new HttpServletRequest()	1.a	FAIL
TC#3	NULL, 15	1.b	FAIL

3.30 StatisticsServlet.java

3.30.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.30.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.31 StudentServlet.java

3.31.1 doGet(HttpServletRequest request, HttpServletResponse response)

See section 3.25.1

3.31.2 doPost(HttpServletRequest request, HttpServletResponse response)

See section 3.25.2

3.32 edgey.py

3.33 imapConnect.py

3.34 sendEmail.py

3.35 sendEmailHandler.py

4. Control Flow Based Testing

A method does not need CFG testing if it is trivial or the method has only one path.

4.7 ExamManager.java

4.7.1 addStudentExam(String[][] answers, String[] studentData)

No CFG testing needed

4.7.2 public void addStudentExam(String[][] answersFront, String[][] answersBack,String[] studentData)

No CFG testing needed

4.7.3 getGrades()

No CFG testing needed

4.7.4 getExamGrades()

No CFG testing needed

4.7.5 getStats()

No CFG testing needed

4.7.6 getExamLetterGrades()

No CFG testing needed

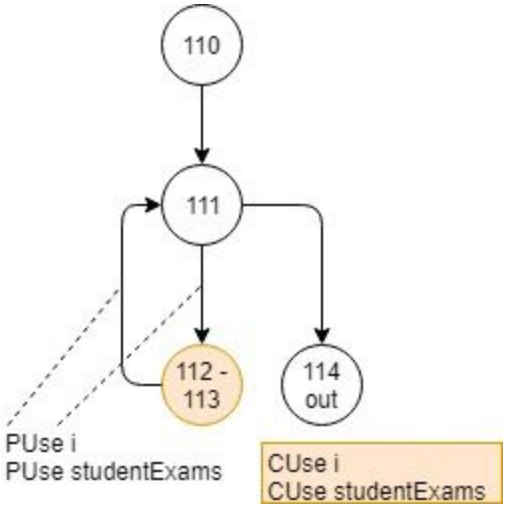
4.7.6 examGradeIntegerConverter()

No CFG testing needed

4.7.7 getExamID()

No CFG testing needed

4.7.8 assignStudentGrades()



4.7.9 getStudents()

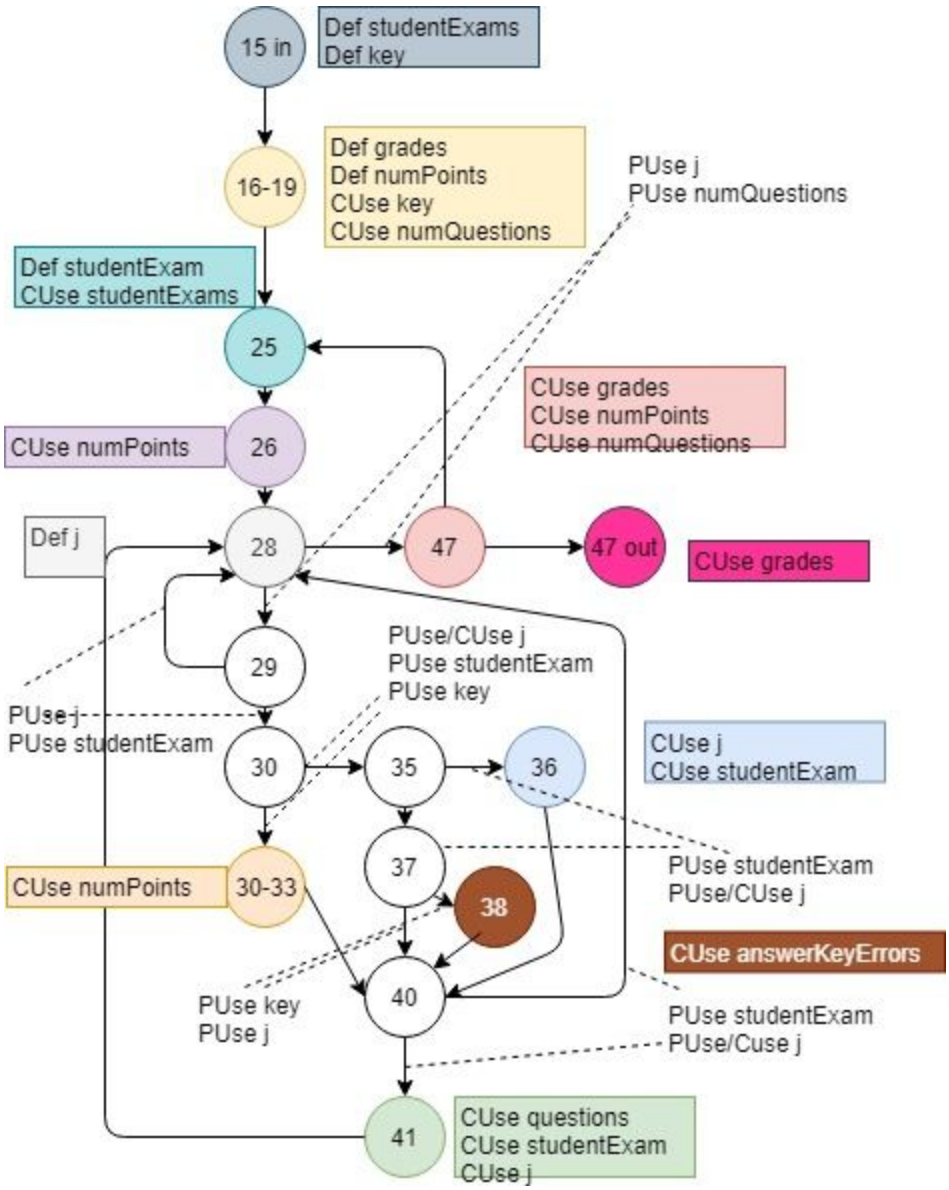
No CFG testing needed

4.7.10 getKey()

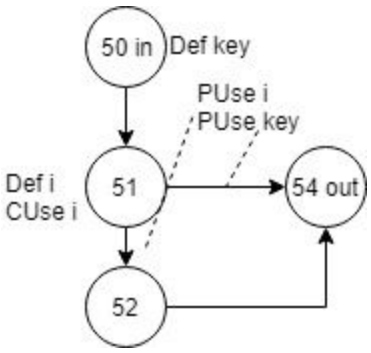
No CFG testing needed

4.8 Grader.java

4.8.1 getGrades(List<Student> studentExams, Student key)

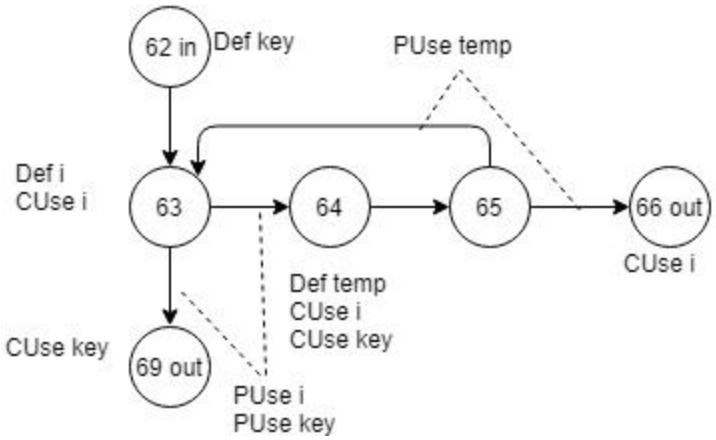


4.8.2 initializeStatsByQuestion(List<String> key)



4.8.3 getStatsByQuestion()

4.8.4 findKeyLength(Student key)



4.8.5 getExamLength()

4.9 JSONBuilder.java

4.9.1 setExamCode(String examID)

No CFG Testing is needed

4.9.1 setExamName(String examName)

No CFG Testing is needed

4.9.1 setExamCode(String examID)

inProgress

4.9.1 buildAnswerKeyJSON(Student key)

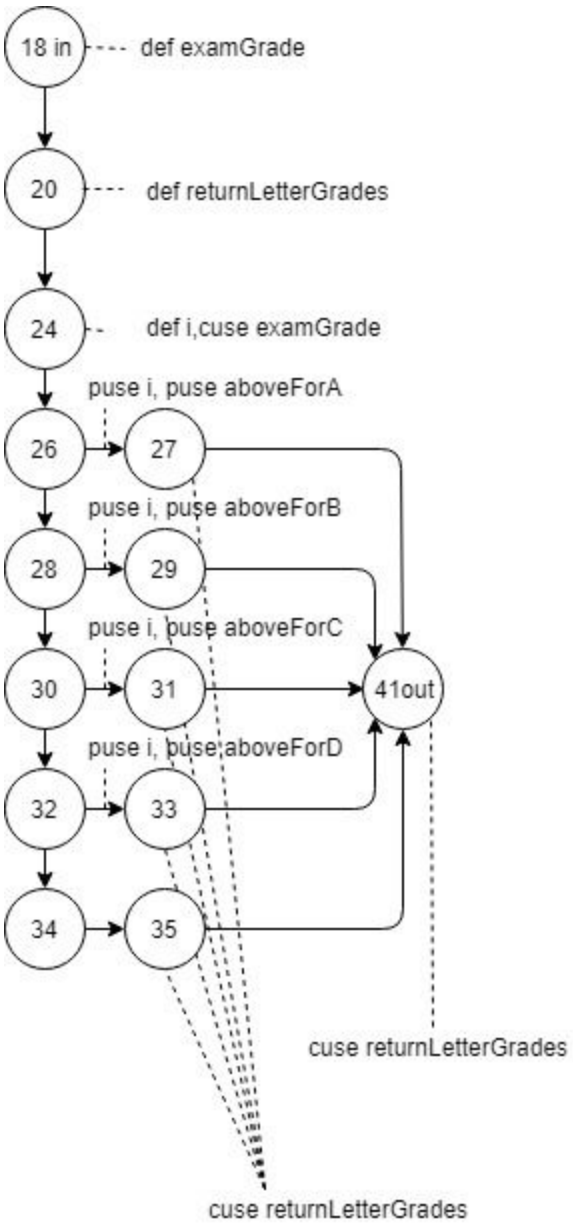
inProgress

4.9.1 buildByStudentJSON(List<Student> studentExams)

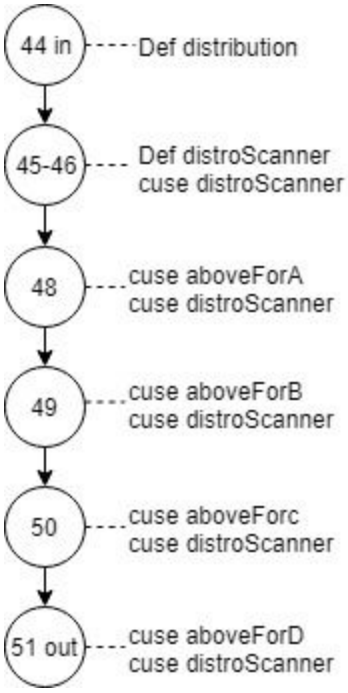
inProgress

4.10 LetterConverter.java

4.10.1 genLetterGrade(List<Float> examGrade)



4.10.2 letterDistribution(String distribution)



4.10.3 getAboveForA()

No CFG testing needed

4.10.4 getAboveForB()

No CFG testing needed

4.10.5 getAboveForC()

No CFG testing needed

4.10.6 getAboveForD()

No CFG testing needed

4.12 MiddlewareInterface.java

4.12.1 addStudentExam(String StudentData)

No CFG testing needed

4.12.2 getGrades()

No CFG testing needed

4.12.3 getExamGrades()

No CFG testing needed

4.12.4 getStats()

No CFG testing needed

4.12.5 getExamLetterGrades()

No CFG testing needed

4.12.6 examGradeIntegerConverter()

4.12.7 getExamID()

No CFG testing needed

4.12.8 setExamName(String examName)

No CFG testing needed

4.12.9 assignStudentGrades()

4.12.10 makeJSON()

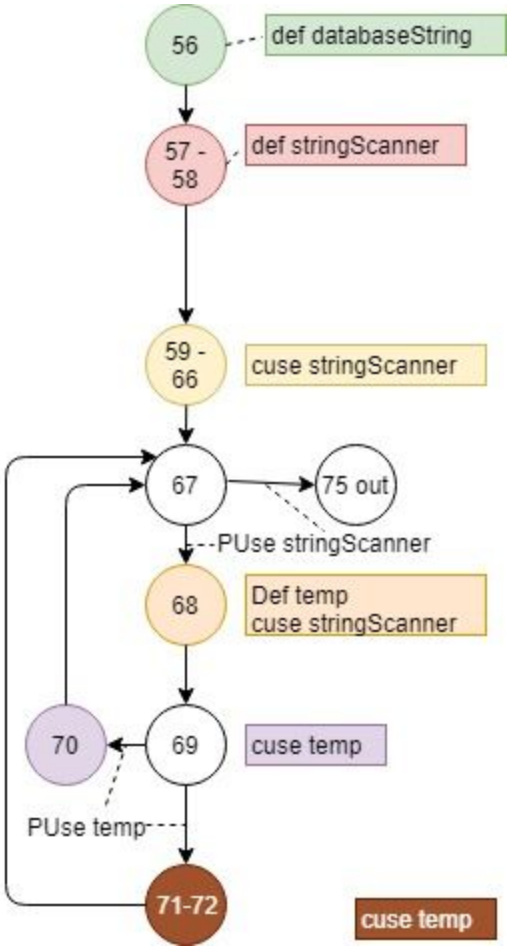
4.16 Student.java

4.16.1 initializeStudentFromStudentData(List<String> studentData)

No CFG testing needed

4.16.2 initializeStudentFromDatabaseString(String databaseString)

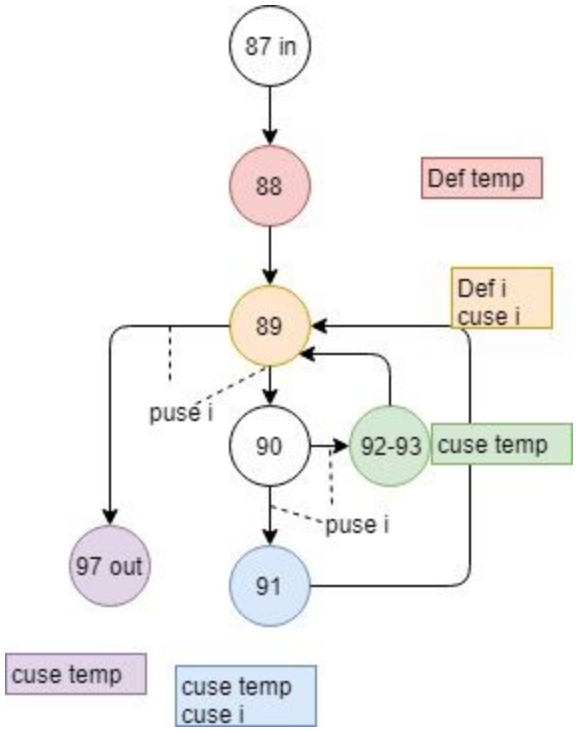
**Node 56 in



4.16.3 initializeAnswers(List<String> answers)

No CFG testing needed

4.16.4 findName()



4.16.5 findSex()

4.16.6 findGrade()

4.16.7 findBirthday()

4.16.8 findId()

4.16.9 findCode()

4.16.10 genJsonData()

No CFG testing needed

4.16.11 alphaConverter(String entry)

4.16.12 setExamGrade(float grade)

No CFG testing needed

4.16.13 setLetterGrade(String letterGrade)

No CFG testing needed

4.16.14 getName()

No CFG testing needed

4.16.15 getSex()

No CFG testing needed

4.16.16 getBirthday()

No CFG testing needed

4.16.17 getGrade()

No CFG testing needed

4.16.18 getId()

No CFG testing needed

4.16.19 getCode()

No CFG testing needed

4.16.20 getAnswers()

No CFG testing needed

4.16.21 geLetterGrade()

No CFG testing needed

4.16.22 getExamGrade()

No CFG testing needed

4.22 AnswerKey.java

No CFG testing needed

4.23 Exam.java

No CFG testing needed

4.24 Professor.java

No CFG testing needed

Statement Testing

(not all fragments of code can do MMC, so statement is necessary)

Minimal Multiple Condition Testing

(This test subsumes branch testing which subsumes statement testing, making those tests unnecessary if this is done.)

Path Coverage Testing

Data Flow Testing

5. Class Testing

5.1 Category Partition Test

5.2 Source Code Test

5.3 Polymorphism Test

//probably will be able to skip this

6. Class Scope Test

6.1 State Transition Matrix

6.2 All Events Criterion

6.3 Flattened Class Scope Test

6.4 Class Interaction Test

7. Unit Tests

7.1 Engine Code:

7.2 Database:

7.3 GUI:

7.3.1 Middleware:

7.3.2 Selenium:

Tests from May 8th.
Starting ChromeDriver 2.46.628402 (536cd7adbad73a3783fdc2cab92ab2ba7ec361e1) on port 16099
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
May 08, 2019 11:45:10 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
1 Incorrect Login Test - has Passed!

2 TestLogin - has Passed!
3 Email display - has Passed!
4 Welcome username should contain comp science - has Passed!
5 Scan Code displayed - has Passed!
6 New exam Displayed - has Passed!
7 Name Change Test - has Passed!
8 Results page buttons displayed correctly - has Passed!
9 Statistics Report Table displayed - has Passed!
10 Statistics Report Table 2 displayed - has Passed!
11 Statistics Report Graph displayed correctly - has Failed!
//For test 11, the chart was removed and should fail.
12 By Question Table displays - has Passed!
13 By Question Graph displayed correctly - has Failed!
14 By Students table displayed correctly - has Passed!
15 Scan Code displayed - has Passed!
16 New exam Displayed - has Passed!
17 Newly created exam has correct amount of questions in answerKey - has Failed!
18 Newly created exam stays on home page - has Failed!
19 Scan Code displayed - has Passed!
20 New exam Displayed - has Passed!
21 Answer Key Page is Navigated to correctly - has Passed!
22 Test key displays - has Failed!
23 Log Out Test - has Passed!
-18- tests passed and -5- tests failed out of -23- total tests. Giving us -78%- have passed.

8. Integration Tests

9. System Tests

10. Acceptance Test

11. Results

11.1 Scan results

Test scan from Wednesday 4-17-19 results -

10 perfect scans - including both answer keys
19 imperfect scans - 2 notable imperfects, others would cause flags for manual review but 2 were incorrectly scanned for no noticeable reason. This was researched farther and fixed by engine, and will be tested again.

Test scan 2 from Wednesday DATE results -

11.2 GenericDao - depreciated class

Tests fail due to the variable “list” being a global variable in the class leading to any select statement to also return any past select statements undergone. If no previous successful select statements are undergone then NULL gets returned instead of a 0 length list for a select that finds nothing.

11.3 InstructorDao - fixed

InstructorDao fails for the same reason as GenericDao. This is fixed in the newer code, however the current tests will fail until future integration.

11.4 Stats

These four images are of the math done for the 4 different testing sets I used in many test cases.

	A	B	C	D	E	F	G	H	I	J	K
1	students	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	total
2	1	1	1	1	1	1	1	1	1	1	9
3	2	1	0	1	1	0	1	1	1	1	7
4	3	0	1	1	0	1	1	1	1	1	7
5	4	1	0	1	1	1	1	0	0	1	6
6	total	3	2	4	3	3	4	3	3	4	29
7											
8	p	0.75	0.5	1	0.75	0.75	1	0.75	0.75		
9	q	0.25	0.5	0	0.25	0.25	0	0.25	0.25		
10	pq	0.1875	0.25	0	0.1875	0.1875	0	0.1875	0.1875	1.1875	
11											
12	k	9									
13	sum(pq)	1.1875		Mean	7.25						
14	var	1.1875									
15	kr20	0		kr21	-0.21053		SumVar	1.1875			
16	rounded	0		rounded	0		C.Alpha	0			
17	Rounded to 0 if negative, and 1 if higher than 1										

	A	B	C	D	E	F	G	H	I	J	K
1	students	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	total
2	1	0	0	0	0	0	0	0	0	1	1
3	2	0	0	0	0	0	0	0	0	1	1
4	3	0	0	0	0	0	0	0	0	1	1
5	4	0	0	0	0	0	1	0	0	1	2
6	total	0	0	0	0	0	1	0	0	4	5
7											
8	p	0	0	0	0	0	0.25	0	0		
9	q	1	1	1	1	1	0.75	1	1		
10	pq	0	0	0	0	0	0.1875	0	0	0.1875	
11											
12	k	9									
13	sum(pq)	0.1875		Mean	1.25						
14	var	0.1875									
15	kr20	0		kr21	-5.33333		SumVar	0.1875			
16	rounded	0		rounded	0		C.Alpha	0			
17	Rounded to 0 if negative, and 1 if higher than 1										

	A	B	C	D	E	F	G	H	I	J	K
1	students	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	total
2	1	0	0	0	0	0	0	0	0	1	1
3	2	1	0	0	0	1	0	0	1	1	4
4	3	0	0	0	0	0	0	0	0	1	1
5	4	0	0	0	0	0	1	0	0	1	2
6	total	1	0	0	0	1	1	0	1	4	8
7											
8	p	0.25	0	0	0	0.25	0.25	0	0.25		
9	q	0.75	1	1	1	0.75	0.75	1	0.75		
10	pq	0.1875	0	0	0	0.1875	0.1875	0	0.1875	0.75	
11											
12	k	9									
13	sum(pq)	0.75		Mean	2						
14	var	1.5									
15	kr20	0.5625		kr21	-0.04167		SumVar	0.75			
16	rounded	N/A		rounded	0		C.Alpha	0.5625			
17	Rounded to 0 if negative, and 1 if higher than 1										

	A	B	C	D	E	F	G	H	I	J	K
1	students	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	total
2	1	0	0	1	1	1	1	1	0	1	6
3	2	1	1	0	0	1	0	0	0	1	4
4	3	1	0	1	1	1	0	0	0	1	5
5	4	1	1	1	1	1	1	1	1	1	9
6	total	3	2	3	3	4	2	2	1	4	24
7											
8	p	0.75	0.5	0.75	0.75	1	0.5	0.5	0.25		
9	q	0.25	0.5	0.25	0.25	0	0.5	0.5	0.75		
10	pq	0.1875	0.25	0.1875	0.1875	0	0.25	0.25	0.1875	1.5	
11											
12	k	9									
13	sum(pq)	1.5		Mean	6						
14	var	3.5									
15	kr20	0.642857		kr21	0.482143		SumVar	1.5			
16	rounded	N/A		rounded	0		C.Alpha	0.642857			
17	Rounded to 0 if negative, and 1 if higher than 1										

3.14.30(TC# 2), 3.14.31(TC# 2), 3.14.32(TC#'s 1&2), and 3.14.33(TC#'s 2&3)

Test #2 for highestScore and lowestScore fail due to the methods returning int's. If a grade ever has a decimal the result for these methods is rounded and therefore incorrect resulting in an accuracy below standards. Ex: 85.5% is now 85%
Since highestScore and lowestScore fail then rangeOfScores which uses both of them fails as well, however it also returns an int and therefore should be fixed as well.

Median fails for the same reason as it returns an int and a decimal answer is possible even if the input has no decimals. Median also fails because the method is not actually calculating the median, it is instead grabbing the value in the middle of the Integer array which will fail if the array is not in numerical order from lowest to highest, or if the array has an odd number of inputs.

Recommendations:

1. To use a different return datatype such as Integer or double for these four methods. Especially Integer since the input for these methods are a list of Integers.
2. To redo the median method to actually calculate the median.

There are multiple classes in Stats.java that are never used. overallStandardDeviation, babyloniansqrt, quartiles, and weightGenerator.
It is recommended that either a use be found or they be removed.
Quartiles is in the requirement documentation, so either the requirements needs to change or it needs to be put into use.

The Class takes exam scores as an input, and yet decides to grade the students exams again over the inputted answer key for some reason? This functionality makes no sense to me as it should be and is a part of the functionality of the grader class and should be removed.

There seems to be a lot of leftover functionality meant for weighting of answers from before it was scratched out of the project that can and should be removed. Not urgent.

Comments are out of date all over the place(nice!)… Many places reference byte arrays that are never in place...

kuderRichardson21(important when fails)

Tested in section 3.14.26

In line **418** calculates the mean weight, seeing as weights are no longer in the functionality this can be changed or defaulted to 1, which is what is happening anyway since all weights are set to 1.

When calculating the number of questions, a 0 is subtracted which quite literally does nothing and can be removed. Makes my IDE mad.

As noted in the comments, a negative result is possible. Through this I can gather that a result greater than 1 is also possible. This goes against the usage of the class and as such it is recommended to set results lower than 0 to 0 and higher than 1 to 1.

Scores is a global variable of grades for the current exam and yet this method does not use them at all and instead re-grades the exams. Usage of the global scores variable instead of the examGrader method is recommended as well as the removal of the answerKey and exam parameters. Exams is also a global variable that should be used in this way.

Test Cases #'s 2,3,4 and 9 all fail due to a lack of exceptions being thrown in some situations.

Test Case #1 Fail:

```
Expected :0

Actual   :-0.80000000
```

Test Case #10 Fail:

```
Expected :0

Actual   :-0.33333334
```

Test Case #12 Fail:

```
Expected :0.482142857

Actual   :0.61764706
```

1 and 10 can possibly be due to not rounding to 0, but 12 is not so there might be a problem elsewhere.

kuderRichardson20(important when fails)

Tested in section 3.14.37

A 0 is subtracted same as in KR21 and can be removed. Still makes my IDE mad. ツ
Mean weight also still calculated.

Test Cases #'s 2,3,4, and 9 all fail for the same reasons as KR21.

Test Case #1 Fail:

```
Expected :0

Actual   :0.22500000
```

Test Case #11 Fail:

```
Expected :0.5625

Actual   :0.20454545
```

Test Case #12 Fail:

```
Expected :0.642857

Actual   :0.61764706
```

cronbachsAlpha(important)

Tested in section 3.14.29
Is done in an over complicated manner. Difference from mean is just 1-mean, there is no need to have a method just for it.
Individual variance doesn't seem to follow the formula of the variance of a population:

$$\frac{\sum(x-\bar{x})^2}{n}$$

Test Cases #'s 2,3,4, and 9 all fail for the same reasons as KR21.

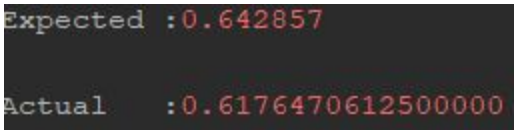
Test Case #1 Fail:

```
Expected :0

Actual   :0.2250000000000000
```

Test Case #11 Fail:

```
Expected :0.5625

Actual   :0.2045454525000000
```

Test Case #12 Fail:



gradesByQuestion

Tested in section 3.14.22

Test Case #'s 2,3,4, and 9 all fail for the same reasons as KR21.

Test Case #'s 1, 10, 11, 12, and 13 ALL fail. The method assumes there are 200 questions without actually CHECKING how many questions there ACTUALLY are meaning there are a large number of 0's for each question that doesn't exist. This may screw over all methods that use this by skewing the results towards the fail's with the extra number of 0's.

Further testing can continue when this is fixed.

proportionPassingByQuestion

Tested in section 3.14.34

Test Case #'s 2,3,4, and 9 all fail for the same reasons as KR21.

Test Case #'s 1,10,11, and 12 fail. Not only are the results wrong, there are a bunch of extra 0's at the end of the result.

proportionFailingByQuestion

Tested in section 3.14.35

Same problems as proportionPassingByQuestion above.

questionFrequency

Tested in section 3.14.38

This method iterates 200 times instead of the number of questions there are. No impact because 0's are removed after. Shouldn't have to remove the trailing nothingness if you never put it there in the first place.

Exceptions not thrown for Test Case #’s 2,3,4, and 9

meanByQuestion and differenceFromMean

Tested in nowhere because why would I.

This is literally the same exact thing as proportionPassingByQuestion and proportionFailingByQuestion, but done better and would have saved me loads of time if the other two did not exist. People who wrote this why did you do this to me you’re killing me.

12. Conclusion