Name: _____

Project 3, Part 1

Black- and White-box Unit Testing

This is a GROUP assignment.

Due date:
This part of the assignment is due on Wednesday, July 20, 2016 at 9:59pm. No submissions will be accepted after the deadline.

The purpose of this assignment is to have you learn and practice the following:

1. Writing software methods and classes that conform with specifications/requirements.
2. Ability to analyze and create black-box unit tests for methods and classes based only upon specifications/requirements.
3. Ability to analyze and create white-box unit tests for methods and classes based upon specifications/requirements and analysis of source code.

Your assignment:
Your group will create a number of related Classes in Java that simulate a basic program to track student course schedules, determine a student's enrollment status, the amount of tuition charged, etc.

Although this is a group project, some of the classes involved in this project will be written INDIVIDUALLY and the source code will NOT be shared with the other members of the group at this time. Instead, individual group members will be assigned to code specific classes, according to the specifications below, and will share ONLY the BYTECODE file (i.e. the .class file and not the .java source file) with group mates, who will then perform black-box unit testing. This will prevent group members for seeing the implementation of the module for which they will design black-box test cases and will more closely simulate a real-world testing experience.

PART	ONE

For the first part of the assignment, you and your group mates will create the following Java classes:

Class Name: Student
Public members: enum StudentType { undergraduate, graduate }
Public methods: Student(String fName, String lName, int ID, Classification classification,
FinancialAid award) // constructor. Creates a new student.
String getFirstName()
String getLastName()
String getName() // returns first and last name, separated by a space
int getID() // returns student ID
StudentType getStudentType() // returns the student type
Boolean isFullTime() // returns true if a full time student (>=9 hours graduate, >=12 undergrad)
FinancialAid getAward() // returns the financial aid award for this student.
Schedule getSchedule() // returns the schedule for this student.
float amountDue() // returns the amount of money owed by this student ,which is equal to the bill
// minus any financial aid award, if eligible. Bill cannot be less than zero.

Class Name: Course
Public members: None
Public methods: Student(String name, String prefix, int number, int section, float hours,
Boolean online, Boolean labComponent) // constructor
String getCourseName() // return course name in format Software Testing and Quality Assurance
String getCourseNumber() // returns course name in the SWE 3643 001
float getHours() // returns the number of hours. Fractional hours are possible, but uncommon.

Class Name: Schedule
Public members: None
Public methods: Schedule() // constructor – creates a schedule with zero courses in it.
int addCourse(Course course) // adds a course to the schedule, unless it's already in there
// and returns number of hours in the schedule. Can throw an
// exception. Only allows up to 18 hours in schedule.
int removeCourse(Course course) // removed a course from the schedule, if it's in there, and
// returns number of hours in the schedule. Does nothing if
// course isn't in there. Can throw an exception.
void clearSchedule() // clears the schedule completely.
Course[] getCourses() // returns an array of courses in this schedule. Array can be empty.
float getHours() // returns the number of hours in this schedule. Fractions possible, uncommon.

Class Name: Bill
Public Members: None
Public Methods: Bill(Schedule schedule) // constructor. Creates a new bill for this student based upon schedule.
float getTuitionCost() // returns the student's tuition according to the following formula:
// $250.00 per undergraduate credit hour (course number <=4999) up
// until 12 credits, after which there is no more charge per hour. Cost is
// $350.00 per credit hour for graduate courses (course number >= 5000)
// up to 12 hours, after which there is no more charge per hour. Online
// courses have a 20% premium added to the cost per hour, regardless of
// whether the schedule is above 12 hours or not. Courses with a lab
// component have a $150.00 lab fee.
// Students are assessed a facilities fee of $295.00 per semester, unless the
// student is ONLY taking online classes, in which case the fee is waived.

ClassName: FinancialAid
Public Members: None
Public Methods: FinancialAid (float amount, Boolean enrollmentContingent) // constructor. Creates a new
// financial aid award in the specified amount and with a condition which
// indicates whether the student must be full time to receive the award
float getAmount() // returns the award amount.
Boolean isContingent() // returns whether the award is contingent on full-time enrollment.

Some of these classes will be created PRIVATELY by INDIVIDUAL team members and the source code for these particular classes WILL NOT be shared with other members at this time. The creation of classes is as follows:

<u>Group member whose last name is FIRST alphabetically:</u>
Student class

<u>Group member whose last name is SECOND alphabetically:</u>
Bill class

<u>Group member whose last name is LAST alphabetically:</u>
Schedule class
(if the group has only two members, the two members can create this class jointly)

<u>All Group Members, jointly:</u>
FinancialAid and Course classes.

PART TWO

For this portion of the project, you will perform black-box unit testing as follows:

1. Give the BYTECODE file (i.e. the .class file, and NOT the .java file) of your class to person whose last name comes AFTER yours alphabetically, or to the first member if your name is last.
2. Perform Black-box unit testing on the Object file that you have received, based upon the descriptions above. You should perform your testing as follows:
    a. Name your test code file CLASSNAME_BlackBoxTest.java.
    b. If you have created any stubs in order to test your class, you may include them in a ZIP file named CLASSNAME_TestStubs.zip.
    c. Create a main method that instantiates an instance of the class you are testing and then executes each of the test cases you have devised, indicating in the console the success or failure of each. It's OKAY to have tests that fail.
    d. During the course of testing, DO NOT have your colleague go back and fix individual errors in his or her class as this time, UNLESS the class is so broken that it cannot even be instantiated in order to test it.
    e. Ensure your name and all relevant information is at the top in comment form.
    f. Include a WRITTEN DESCRIPTION (in paragraph form) at the top of the file in block comment format, (e.g. "/* … */"), explaining the black box texting approach or approaches you have employed and why you think this is appropriate
    g. If, in the course of preparing your tests, you have any supporting written materials, such as decision tables, flow graphs, etc., include them as a PDF named CLASSNAME_BlackBox.pdf
    h. Paste the OUTPUT of your test execution at the BOTTOM of the .java file in block comment format (e.g. "/* … */").

ASSIGNMENT SUBMISSION

Assignment Submission:

You must submit this part of the assignment on D2L as a single ZIP file by the specified date and time. Your ZIP file should contain:

1. All ClassName.java files. YOU SHOULD NOT HAVE SHARED THE SOURCE CODE OF YOUR INDIVIDUAL FILES WITH THE REST OF THE GROUP UNTIL THIS VERY MOMENT.
2. All ClassName.class files. (just in case I can't compile them myself for some reason!)
3. All ClassName_BlackBoxTest.java files.
4. Any supporting ClassName_TestStubs.zip files.

You MUST name your file in the following manner:

`Group#_Project3_Part1.zip`