Data Structures and Algorithms 1

Worksheet 1 – JUnit Testing

Create an abstract Java class called StudentGrade, which can be used to store marks/grades for students. Properties of StudentGrade should include student name and module name (keeping it simple, so no need for separate e.g. Student and Module classes in this instance). Add getter/setter methods for these properties. StudentGrade should also include an abstract method classifyGrade() that eventually returns a string indicating the category the grade is in, namely “Distinction”, “Merit”, “Pass” or “Fail”. It should also include an abstract method getGrade() that returns an integer indicating the actual grade.

Two subclasses of StudentGrade should be created: PercentageGrade and LetterGrade. PercentageGrade should store grades as integer values between 0 and 100 only. The grade categories for PercentageGrade are: 70-100 for Distinction, 50-69 for Merit, 40-49 for Pass, and 0-39 for Fail.

LetterGrade should store grades as letters A, B, C, D and E only. The getGrade() method should return integer values 1 for A, 2 for B, 3 for C, 4 for D and 5 for E on for LetterGrade objects. The grade categories for LetterGrade are: A or B for Distinction, C for Merit, D for Pass, and E for Fail.

Appropriate setter methods named setGrade() should be provided for both subclasses, with an integer parameter for PercentageGrade and a case-insensitive character or string parameter for LetterGrade. The setGrade() methods should throw appropriate exceptions (e.g. IllegalArgumentException) if invalid arguments are used (e.g. 110 for PercentageGrade, or ‘Q’ for LetterGrade).

Once all the above have been created, use the facilities in either IntelliJ or Eclipse to create a number of JUnit tests to test the classes to ensure that they are working correctly. Each @Test method should include at least one assert statement (e.g. to ensure that a grade stored in a PercentageGrade equals the subsequently retrieved one, that invalid arguments throw exceptions as required, etc.). Use assertEquals(), assertTrue(), assertFalse(), assertNull(), assertNotNull(), etc. as appropriate in your assert statements. You should end up with at least 8-10 separate and useful JUnit tests that successfully run/pass by the end of the worksheet. You should also incorporate @Before and @After methods to manage any setup and tear down operations. You are free to add additional properties and getter/setter methods to the above classes as you experiment with JUnit testing.

Note:
The objective of this worksheet is to ensure that you are familiar with JUnit testing in Java, and can use the JUnit facilities in either IntelliJ or Eclipse for test-driven development (TDD). The CA in this module will require you to show evidence of TDD, so make sure that you are sufficiently familiar with JUnit testing to this end. This worksheet does not have to be submitted, and it does not count towards your final mark. It should ideally be completed in the Week 1 lab sessions.