**COMP 132 - Homework # 3**

**Conditionals, Booleans and Composition**

**Part 1 (uses Employee.java)**

**Note: You may use Eclipse to write and run your code for each question. The completed code for each question can be copied from Eclipse and pasted here.**

Examine the code for class Employee.java. This class will be used to calculate weekly paycheck amounts, with hours worked above and beyond the normal work week being paid a higher rate. The fields of the class, the constructor and four accessor methods are provided for you:

* the name field holds the name of the employee
* the rate field holds the hourly pay rate (in dollars) for this employee
* the standardHours field holds the minimum number of hours worked per week needed to qualify for overtime pay (i.e. any hours worked beyond this will be paid at the overtime rate)
* the multiplier field holds the amount to multiply the hourly wage by for overtime hours. For example, to pay "time and a half" for overtime hours, this field should be set to 1.5.

Complete the following tasks:

1. Complete the implementation of the giveRaise method, which increases the employee's pay rate by the specified amount.
2. Complete the implementation of the calculateWeeklyPay method that takes the number of hours worked for the week as a parameter, and returns the paycheck amount for the week. For example, suppose we are using using an Employee with a 40 hour work week, an $8/hr pay rate and a pay multiplier of 1.5. If this method is passed 45 (hours) when called using this employee, it should return 380.0, which is 40 \* 8 + (45 - 40) \* 8 \* 1.5. On the other hand, if the method is passed 30 (hours) in this situation, it should return 240.0, which is just 30 \* 8 (i.e., no overtime hours were worked).

*Hint: you will need to use an if statement in this method.*

1. Add a test class for class Employee that tests the constructor and covers all statements of the giveRaise and calculateWeeklyPay methods.

**Part 2 (uses GradeCalculator.java)**

1. Write test cases that achieve statement coverage for the grade method.
2. Change the first line of the grade method to:

 int pct = (score / numPoints) \* 100;

and run your unit tests for this method. Explain what happens and why. **Before continuing be sure to undo this change.**

1. Now change the order of the if statements in the grade method as shown below:

 if (pct >= 90) {

 letGrade = "A";

 } else if (pct >= 70) {

 letGrade = "C";

 } else if (pct >= 80) {

 letGrade = "B";

 } else {

 letGrade = "other";

 }

Run your tests from # 4 again. Which of your test methods fail when you run your tests? Explain why each of those tests failed. **Be sure to undo this change before continuing.**

1. Complete the implementation of the passingGrade method.
2. Write test cases that achieve statement coverage for the passingGrade method that you implemented in the previous exercise.

**Part 3 (Booleans)**

1. Consider the boolean expression: (x < 10) && (x > 4). What values of x make this expression false? Justify your answer.
2. Consider the boolean expression: !(x < 10). What values of x make this expression true? Justify your answer.
3. Consider the boolean expression: (year % 400 == 0) && (year % 100 != 0). What values of year make this expression true? Justify your answer.

**Part 4 (Composition)**

1. (**uses Car.java from the BadCarDesign folder** ) Discuss the problems with the given design of the Car class in terms of the concepts discussed in class *(modularization, abstraction, code reuse).*
	1. The Car class in this project is not well designed and would be better implemented using composition. Improve the design of the Car class using Composition in a way similar to what we did in class to improve the design of the Student class.
		1. Create new classes to represent objects for the wheels, radio and engine.
		2. Define the appropriate fields in each of your new classes.
		3. Update the definition of the Car class using composition with your new classes.

Your classes do not need to contain any constructors or methods, but you should check that your class definitions will compile correctly. Copy and paste each class definition into your homework submission as your answer for this question.

1. (**uses Bicycle.java, BicycleSeat.java, and Tire.java from the BicycleDesign folder** )
	1. complete the definitions of the BicycleSeat and Tire classes according to the specifications given in those classes
	2. create test classes for the BicycleSeat and Tire classes, and write test cases that achieve statement coverage for those classes
	3. Complete the definition of the  Bicycle class according to the specifications given in that class.
	4. create a test class for the Bicycle class, and write a test case that checks that the constructor sets the manufacturer and number of gears correctly for a Bicycle object. You do not need to check the values of any other fields set by the constructor.
2. Using the *same classes (BiclycleDesign)* that you worked on in the previous exercise:
	1. Draw a *class diagram* for the classes in this project. [It’s fine to draw it by hand on paper then paste in a scan or photo.]
	2. Draw an *object diagram* for an instance of the Bicycle class. Use any values for the fields that are of the appropriate type and that could reasonably represent a real bicycle. [It’s fine to draw it by hand on paper then paste in a scan or photo.]