Directions for Clock Lab Assignment

A manufacturer of digital desk clocks has produced some hardware to display the current time. This hardware needs software to drive the display. To make sure that all necessary software is properly written by the developer, the hardware manufacturer has produced a Java-based software *interface* called **Clock** which is being provided to you for this assignment.

Your assignment is to implement this Clock interface to create three different classes of clocks:

1. **DeskClock:** This is the base class and represents the simplest of the three clock designs that you must construct. DeskClock displays the time as updated by DeskClockRunner software which is also being provided to you. There are two ways to create a DeskClock - one takes no arguments. This DeskClock is created with 12:00 as its default time for display. The other DeskClock constructor takes two arguments, both integers, which represent the current hour and minute to be displayed. Please look in DeskClockRunner for examples of how DeskClocks are to be created and modified. Note that DeskClock does not distinguish between AM and PM.
2. **AM\_PM\_Clock:** This clock has all the features of DeskClock but also displays AM or PM after the time. So whereas DeskClock would display 07:00, the AM\_PM\_Clock would display 07:00AM or 07:00PM. Note that you must toggle AM to PM and PM to AM whenever time is added to the clock to cause it to roll over past midnight or noon. Please look at AM\_PM\_ClockRunner for examples of how these clocks will be created and used.
3. **MilitaryClock:** This clock has all the features of DeskClock but displays time as is done in the military. The time varies from 00:00 to 23:59. After 23:59, the time rolls over back to 00:00. In all other respects, the MilitaryClock acts like the DeskClock. Please look in MilitaryClock

The test code for all three versions of these clocks are being provided to you (DeskClockRunner, MilitaryClockRunner, and AM\_PM\_ClockRunner). ***You are not allowed to change the code in these tester classes.*** Instead, you have to provide code to create the classes DeskClock, MilitaryClock and AM\_PM\_Clock that demonstrate you have properly implemented the Clock interface and the requirements listed above.

**Hints:**

1. The method normalizeTime() is responsible for how the time will be displayed for each style of clock. Thus, it represents the fundamental method that must be overridden for each clock type.  
  
2. Try to use code that is already written whenever possible instead of writing new code. Thus, if you have written DeskClock properly, the additional code needed to implement MilitaryClock and AM\_PM\_Clock should be minimal. If you find yourself rewriting large portions of code, you probably did not develop/construct the base class (DeskClock) correctly.

3. Do not bury magic numbers in your code.  
  
4. Try to use method calls in place of assignment statements in your code when possible. Your teacher will explain this principle in class before you begin your assignment.