AP Computer Science Java
Mr. Clausen
Program 5A

Program 5A LastNameFirstNameP5A (Military Time: 50 points)

This program will require two java files. You will create a class to work with Military Time, and a client program to test your class file. To see a model for this type of program, double click on My Computer, then these folders, APCSFileJava, 3rd Edition Textbook Source Codes, Chapter05, Case Study, and open the files: Student.java and StudentApp.java.
DO NOT USE a break statement to escape from any type of loop in this class regardless of what the book says or models.

1) Begin with the class file (server file) Military Time.

2) Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Don't forget to change the information from program 2A to include the new information for this program.**

3) There are no packages to import for this file.

4) Declare the class name in the format:
   public class **LastNameFirstNameMilitaryTime**.
   Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, **LastNameFirstNameMilitaryTime.java**.

5) The class file does not have a main method

6) There are no constants for the class file.

7) You do not instantiate an object of the Scanner class or of any other classes in the Military Time class.

8) Declare all of the variables necessary for this class. You will need **integers** to store hours, minutes, and seconds. Don't forget to declare these variables as **private**. Make sure that you use descriptive variable names to create readable, self-documenting code.

9) Type the following comment:
   //--------------------------Constructors--------------------------
   Don't forget that constructors are **public** methods, do NOT have a return type, and **MUST have the same name as the name of the class**. You will need three constructors. A default constructor that initializes all of the private class variables to 00:00:00, a parameterized constructor that allows the client to instantiate and initialize a time to whatever they wish. (This will be set in your client program.) The third constructor will

allow us to copy a time to another instance of the Military Time class. ("Chain" these constructors, see Pages 166 – 167 for examples of parameterized constructors and chaining constructors.)

10) Type the following comment:
//----------------------------------Mutators---------------------------------
Don't forget that Mutator methods usually are **void** methods and therefore do NOT use a return statement. They will also need at least one parameter in order to change the values of the private variables. In this section, you will create the mutator methods setHour, setMinute, and setSecond so the client can set the time after the time objects have been instantiated. You will need another mutator method incrementTime, which will add one second to the time. If the number of seconds is greater than 59, set the seconds to zero and increment the minute by one minute. If the minutes are greater than 59, set the minutes to zero and increment the hours by one. If the number of hours are greater than 23, set the hours to zero, and the minutes to zero

11) Type the following comment:
//----------------------------------Accessors--------------------------------
Remember that Accessor methods usually do not have parameters, and only "access" the private variables. Therefore, they do need a return type and must include at least one return statement in order to return the value of the private variables to the client program. In this section, you will create the accessor methods, getHour, getMinute, and getSecond which should return the values of the private class variables: hours, minutes, and seconds. Also include the accessor method toString, which will display the hours, minutes, and seconds in military time. If the hours, minutes, or seconds, are less than 10, display a zero and then the number of hours, minutes, or seconds. The display should look similar to: 01:05:00 where there are always two digits to represent the hours, minutes, and seconds. (Note: Real Military Time does not use the colons (:) to separate the hours from the minutes from the seconds. We will include them for readability.)

12) If you haven't done so already, compile this file, and debug all errors so that your class file is ready for the client program.

Use blank lines to separate each of the program sections listed in all of the steps above. Now it's time (no pun intended) to create the client program, **LastNameFirstNameP5A.java**.

13) Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Don't forget to change the information from program 2A to include the new information for this program.**

14) Import the package:
import java.util.Scanner;

You do not need to import your Military Time class file, but both of the .java files and their corresponding .class files need to be in the same folder.

15) Declare the class name in the format **LastNameFirstNameP5A**. Don't forget that the filename needs to be the same when you save your program. This would be a good time to save your program if you haven't done so already, **LastNameFirstNameP5A.java**.

16) Declare the main method:
public static void main(String [] args)

17) There are no constants necessary for this program.

18) Instantiate an object of the Scanner class in order to use input from the keyboard. In addition, instantiate three objects of the Military Time class. Use another_time, which implicitly invokes the default constructor and sets the time to 00:00:00, a second instance, current_time (set it to 23:59:00) that implicitly invokes the parameterized constructor, and a third copy_time, which will use the copy constructor later in this client program.

19) Declare all of the variables necessary for this program. You will need **integers** for hours, minutes, and seconds. Use different variations of these names from the private variables in the Military Class file, but they can be the same name as the parameters in the Military Class file. Make sure that you use descriptive variable names to create readable, self-documenting code.

20) Type the following comment:
//---------------------------Display My Information--------------------------
Follow this comment with println statements to display your name and period output just like those used for program 2A. **Don't forget to change the information from program 2A to include the new information for this program.**

21) For the Input section, type the following comment:
//-----------------------------------Input-----------------------------------
Ask the user to enter the hours, then the minutes, and then the seconds. Using the mutator methods, change the another_time object according to what the user entered for hours, minutes, and seconds. In order to practice calling our accessor methods, after you ask the user for the hours, echo this value back on the monitor screen using the accessor, getHour. Do the same for the minutes and seconds also.

22) For the Calculations section, type the following comment:
//----------------------------Calculations & Output---------------------------
Copy the current_time to the instance variable copy_time, so we can test our copy constructor. Next, create a "for loop" to count from 1 to 100. Inside the loop have the instance current_time invoke the member method incrementTime and toString. This will display 100 times incremented by one second each. This will test your incrementTime method to make sure it is working correctly. The variable current_time object should start at 23:59:00. Declare the loop counter inside the loop header; call this variable "counter". Display these times in a column format displaying one time in each row.

23) For the Output section, type the following comment:
//----------------------------------Output----------------------------------
For the output portion of your program, display the values of copy_time, current_time and another_time after the 100 iterations of the current_time that was displayed in the "for loop". Make this output easy to read and formatted nicely with blank lines between each part of your output (except for the 100 times that the loop displayed).

Use blank lines to separate each of the program sections listed in all of the steps above. When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, in the Program 5A folder.
After you are finished, you will need to turn in (copy and paste) the three data files:

1) **LastNameFirstNameMilitaryTime.java**
2) **LastNameFirstNameMilitaryTime.class**
3) **LastNameFirstNameP5A.java**