**CSE 1310 – Fall 2017**

**Introduction to Programming**

**Lab 3**

**Lab 3 parts to complete:**

**Part 1:** Given the code you wrote for ***Lab2Part5c.java***, create a new Java application called ***Lab3Part1.java*** that contains the same code you wrote for Lab2Part5c. For this lab you are going to make the following modifications. [The code of Lab2Part5c.java is given below]

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package lab2part5c;

import java.util.Scanner;

/\*\*

\*

\* @author Indra Bhurtel

\*/

public class Lab2Part5c {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

Scanner input=new Scanner(System.in);

int exactStartTime = 0;

int exactBreakStartTime = 0;

int exactBreakEndTime = 0;

int exactEndTime = 0;

System.out.print("Exact Start Time : ");

exactStartTime=input.nextInt();

System.out.print("Exact Break Start Time : ");

exactBreakStartTime=input.nextInt();

System.out.print("Exact Break End Time : ");

exactBreakEndTime=input.nextInt();

System.out.print("Exact End Time : ");

exactEndTime=input.nextInt();

double recStartTime = change(exactStartTime);

double recBreakStartTime = change(exactBreakStartTime);

double recBreakEndTime = change(exactBreakEndTime);

double recEndTime = change(exactEndTime);

double res = (recEndTime - recStartTime) - (recBreakEndTime - recBreakStartTime);

double hoursToPay = calculateHoursToPay(res);

System.out.printf("%8s%20s%10s\n","Time","Start","End");

System.out.printf("%8s%10s%10s%10s%10s\n","Data:","Start","Break","Break","End");

System.out.printf("%8s%10d%10d%10d%10d\n","Exact",exactStartTime,exactBreakStartTime,exactBreakEndTime,exactEndTime);

System.out.printf("%8s%10.0f%10.0f%10.0f%10.0f\n\n","Recorded",recStartTime,recBreakStartTime,recBreakEndTime,recEndTime);

System.out.printf("%16s\n","Total");

System.out.printf("%8s%10.0f\n","hhmm",res);

System.out.printf("%8s%10d\n","Hours &",((int) res) / 100);

System.out.printf("%8s%10d\n","Minutes",((int) res) % 100);

System.out.printf("%8s%10.2f\n","As hours",hoursToPay);

System.out.printf("%8s%10.2f per hour\n","At rate",8.25f);

System.out.printf("%8s%10.2f\n","Pay $",hoursToPay\*8.25);

}

public static double change(int num) {

int n = num % 100;

if (n <= 7) {

num = num / 100;

num = num \* 100;

} else if (n >= 8 && n <= 22) {

num = num / 100;

num = num \* 100 + 15;

} else if (n >= 23 && n <= 37) {

num = num / 100;

num = num \* 100 + 30;

} else if (n >= 38 && n <= 52) {

num = num / 100;

num = num \* 100 + 45;

} else {

num = num / 100;

num = num \* 100 + 100;

}

return num;

}

private static double calculateHoursToPay(double res) {

int temp = ((int) res) % 100;

switch (temp) {

case 15:

res += 10;

break;

case 30:

res += 20;

break;

case 45:

res += 30;

break;

}

return res/100;

}

}

**1.a)** For the code you wrote for Lab2Part5c, does the code do any type of error checking? (Answer in a complete sentence.) What errors is it checking for? List all of the errors that your code currently checks for AND for each error, tell what the code does if it finds that error. **Put your answers in the answer file.** **(8 points)**

**1.b)** Now, you will add some additional error checking to Lab3Part1 and create **Lab3Part1b.java**. First, for this question, you must error check the four exact time values before calling the method to round them. This means that your code must take the exact values and verify that these values are meaningful military time values. As an example, 1134 is a meaningful value while -1134, 3116, 2399, and 266 are all invalid values.

Your program must check the hours value and the minutes value and determine if they are valid. If the values are not valid, then handle them in the following ways:

1. If the hour value is negative, then set the hour to 0, take the absolute value of the minutes, and tell the user what the error was and how it was handled. Note: since a negative value is handled in the hour, then the exact minute values cannot be negative.
2. If minutes value is greater than 59, then take the minute value divide by 60 and add that number to the hours. Also take the minutes value and modulo by 60 and save that as the new minute value. Give the user a message telling them the number of hours added to the hours and the new value for minutes.
3. If the hour value is equal to 24, then set the hour to 0 and tell the user what the error was and how it was handled.
4. If the hour value is greater than 24, then modulo the hour value by 24, set the hour to this remainder, then tell the user what the error was and how it was handled.

Second, your program must insure that the four exact error-checked values are in order from earliest to latest time. The smallest valid time value is 0000 and the largest valid time value is 2399 within one day. Check to make sure that the four values are in order. If the four values are not in order from smallest to largest, then ignore the four time values for that day and give the user a message telling them the data was not valid because the values were not in time order.

Test your program by setting the input values to invalid numbers and out of order numbers to make sure these numbers are handled correctly. Once the program works**, save it as Lab3Part1b.java . (16 pts)**

**1.b.i)** In your **Lab3Part1b** code, how did you check that the four time values for a day were in order? Give the Java code from your program that does this testing AND explain in words why you did the testing this way. **(4 pts)**

**1.b.ii)** Come up with an alternate way to check that the four time values for a day were in order. Write a few lines of Java code to show this different method for testing AND explain in words why you did NOT do the testing this way. **(4 pts)**

**1.b.iii)** In the error checking for part I in **Lab3Part1b** described above, you are checking for negative inputs, setting hours to zero (0) and then taking the absolute value of the minutes as the new time. If you were a manager of these employees and you saw a negative time value, what is another approach that the program could take in terms of “error-handling” for negative times? Write down your approach AND explain why your approach would be a reasonable way to handle this kind of error in the real world**. (4 pts)**

**Put your answers for 1.b.i) – 1.b.iii) in your answers file.**

**1.c)** Now, you will copy **Lab3Part1b.java** into a new project and create **Lab3Part1c.java**. For this question, you must allow the user to enter their times as standard times rather than military times and your program must convert the standard times to military times then go through the rest of the program.

This means that you must allow the user to enter a value in the form of HH:MM and that you must let them indicate AM or PM for the time. You can have them enter this info all as one input or you can get the information as two inputs, the hour and minute as one input, then the am/pm info. You CANNOT have them enter hours as one input and minutes as another input and am/pm as a third input. You MUST have them enter the time as HH:MM . Your program must read and interpret the user input and then must convert the standard time value to a military time value for the program to use. NOTE: You MUST use a delimiter with your scanner to get the hour value from the input string. HINT: If you remove the hour value and the delimiter symbol from the input string, you can then get the minutes (and am/pm in single inputs) by using the substring command.

Your program must take in the user inputs for the four exact times and then print the inputs the user gave before you create the military time. These inputs should be added as a row in the output “Time Data:” table. These inputs must be on the line immediately below the table titles, and the row title should be “Inputs”. Each input value should be printed in the form HH:MMxm where xm is either am or pm. The modified table will have the additional row as shown below.

Time Start End

Data: Start Break Break End

Inputs 9:19am 10:58am 12:14pm 6:32pm

Exact 919 1058 1214 1832

Once the program works**, save it as Lab3Part1c.java . (10 pts)**

**1.c.i)** If your program reads the time inputs as strings, will you ever get an *InputMismatchException*? Explain why or why not? **(4 pts)**

**1.c.ii)** Clock values are typically written as HH:MM which then requires us to deal with the formatting and punctuation to get to the hour and minute values. List at least two other kinds of data that are normally written with some type of format and punctuation and then suggest (in words) how you would read the values in a Java program so that you could capture the information without the punctuation. **(6 pts)**

**Put your answers for 1.c.i) – 1.c.ii) in your answers file.**

**1.d)** Now, you will copy **Lab3Part1c.java** into a new project and create **Lab3Part1d.java**. For this question, you will be reading data from an input file and calculating total pay for multiple days. We will do examples of reading from files in class to help you start. The input file is given with this assignment and is called InputLab3Part1.dat . This is a plain text file that will have five input values on each line. The first four data items on each line in the file will be values in the form of HH:MMxm and the last value will be a money amount for pay per hour. Two example lines from the file could be:

7:54am 11:36am 1:14pm 3:58pm 8.25

9:19am 10:28am 11:51am 6:32pm 10.00

where each value is separated by one space from the next value. Each line of the file will represent the data for one day. There will be multiple lines in the file and your program will not know how many lines are in the input file. Your program will need to have a loop to read until there are no more lines of data in the file. As each line is read, it should be counted. For each line of data the four time values will be read, the rate will be read and then all data is processed to find the total pay for the day. This data for each day will be printed out before the next line of data is read from the file. The days will be labeled as “Day # Data” in the output table, as opposed to “Time Data”, with a change to the table contents as follows:

Day 1 Start End Hourly

Start Break Break End Rate

Input 7:54am 11:36am 1:14pm 3:58pm $ 8.25

Exact 754 1136 1314 1558

Recorded 800 1130 1315 1600

Once the program works**, save it as Lab3Part1d.java . (12 pts)**

**1.d.i)** Explain in 3 sentences or less, why you chose the loop control structure that you chose for your program, what test you are using to control the loop and why you chose that test. Put your answer in your answer file. **(6 points)**

**1.d.ii)** Why would we want to read data from an input file rather than from a user directly? Give at least three benefits to reading from files. Also give one drawback of reading from a file. Write a complete sentence explaining each benefit and drawback. **(8 pts)**

**Put your answer for 1.d.i) – 1.d.ii) in your answers file.**

**1.e)** Now, you will copy **Lab3Part1d.java** into a new project and create file **Lab3Part1e.java**. For this question, you will be revising the section of your code that handles the rounding. In your current code, you have four sections of code that do very similar things for the four exact input variables. You will now write a method (function) that will do the rounding task by taking in a time (exact time) and returning a rounded time (recorded time). The rounded time returned by the function should then be saved as the recorded time that corresponds to the input exact time.

The rounding method will take in a military time value and return the closest 15 minutes increment to that value. The rounding method will use the ranges given in Lab 2 to determine which values round to which increments. The method should use nested if-else structures to handle the rounding. Remember that the last range (53 to 59) will round to the next higher hour unless the current hour is 23 in which case it will round to hour 0. Once the correct rounded hour is found, that value is returned from the method. If the minutes value is invalid, then the minutes should be set to 0.

The code from Lab 2 that does rounding for the four input exact values should be removed and replaced with calls to the rounding method for each exact input value. The results from calling the method should be the same as the code from Lab 2. Once the program with the new method works**, save it as Lab3Part1e.java . (10 pts)**

**1.e.i)** What is the benefit of writing a method for rounding the exact time inputs in this program? Describe at least two benefits that the method gives you as the programmer. **(4 points)**

**1.e.ii)** Is there any other section or sections of code that you feel would make a good method for this program? If yes, describe the section(s) of code and explain why a method would be good. In no, explain why all the code should stay in the main method of the program.  **(4 pts)**

**Put your answers for 1.e.i) – 1.e.ii) in your answers file.**

--------------------------------------------------------------------------------------------------------

**Suggestions**

Pay close attention to all requirements on this page, including file names and submission format. Even in cases where the program works correctly, points will be taken off for not following the instructions given on this page (such as wrong file names, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming. Contact the instructor or TA if you have any questions.

**How to prepare your assignment to submit**

Create a folder outside of the Netbeans projects folded and label it with LabX, your lastname, then your first initial. So if your name is Happy Camper, then your Lab 1 folder name is Lab1CamperH. Save your CamperHLab1answers file in this folder. Next save all the .java files for the assignment in the folder. Below is how to do that. Once you have put all the needed .java files in your folder, zip the folder to compress it before submitting it (see info further down with submission details).

**Saving .java files out of NetBeans**

Go to the .java file (the program) in NetBeans that you want to save. Make sure the program is open in the main window and is the program you are looking at. You should see the window tab highlighted for the program you want. For example, if you are saving the .java file for Lab1Part5 you will see the window tab for that file will say lab1part5/Lab1Part5. Make sure your cursor clicks in the editing window of that file.

From that window, go to the File menu and select Save As. This will bring up the directory where the file is currently stored (lab1part5 in NetBeansProjects). Navigate from the current directory to your new folder and save the file in the new folder. This will make a copy of the program in your new folder. You will notice now in NetBeans that the tab on the window of the program has changed. The tab will now say Lab1CamperH/Lab1Part5. This shows that you have made a copy and stored it in the new directory. You should close this window in NetBeans.

NOTE: If you have closed a project in NetBeans and want to go back to it to edit it some more, you can go to the project by going to the Projects tab either in the pane or on the left margin of the window. Then select the folder of the project you want. Open the folder and select Source Packages, then select the name of the class (this is the same as the file) you want, then click on the .java file. This will open the program again in the editing window for you to keep working on.

TIP: If you open the .java file that you saved in your folder (Lab1CamperH/Lab1Part5) you can edit it in NetBeans but NetBeans cannot run it. This is because NetBeans expects all the files to be in its own folder. What will happen is that you will make changes but when NetBeans runs, it will run the copy that is stored in NetBeans not the copy from your folder so none of your edits will be executed. This is why you should always go back to the files in NetBeans to keep working.

JAVA FILE NOTE: Only the .java files and the answers file should be included in your zipped folder. Do NOT include the entire NetBeans project. You will be penalized if you save the project instead of just the .java file.