**CSE 1310 – Fall 2017**

**Introduction to Programming**

**Lab 2**

**Lab 2 parts to complete:**

**Part 1:** Given the code below, create a new Java application.

public static void main(String[] args)

{ /\* \*\*\*\*\*

Declare the needed input object here using the names shown below in the program.

Also add any other needed lines to make the input object function.

\*\*\*\*\*\* \*/

// When running the program, first input should be "SE" entered

// without the quote marks

System.out.print("Please enter the abbreviation for your major "

+ "(like CS or SE or CpE, etc.): ");

abbrev = input.next();

// Input here should be "2021"

System.out.print("Please enter an expected graduation year: ");

gradYear = input.nextInt();

String sem = "Spr";

char dot = '.';

char comma = ',';

String month = "May";

String sp = " ";

String gradDate;

// should be semester (with a period), then month

// then year with the words between as given

gradDate = "As a "+abbrev+" major"+comma+" you should graduate in "

+sem+dot+" semester"+comma+sp+month+sp+gradYear+dot;

System.out.println(gradDate); }

}

**1.a)** Add the needed code to the program above to make it run. Save the program as *Lab2Part1a.java* ***.* (5 pts)**

**1.b)** If the user can enter only *CS*, *CpE*, or *SE* for the major and only *2021* or *2022* for the graduation year, then in the answers file, show two of the six possible outputs that could be produced by the program, excluding the one that is directed by the comments in the code itself. **(4 pts)**

**1.c)** What is different about variables *comma* and *dot* from the other variables in the program and why can they be used with the other variables? Put your answer in the answers file. **(4 pts)**

**1.d)** Rewrite the program so that it accepts the same inputs and produces the same output but variable *gradDate* is removed from the program. Do not create a new variable to replace *gradDate*. **Save this modified program as *Lab2Part1d.java .* (5 pts)**

**Part 2:** Create a NEW Java application Lab1Part2. Do not reuse the Lab1Part1 program. In Lab1Part2, enter the code below inside the curly braces of the *main* method. Remember, if you create this in NetBeans, be careful about cutting and pasting. Things like quotation marks can mess up when copied. Also note that when you start a new program, i.e. make a new Java application, in NetBeans that you will need to set a new Main Project before you can run your new application. Go to the Run menu, choose Set Main Project, then select the name of the new application you just created.

int ival1 = 5;

int ival2;

double fval1 = 3.0;

double fval2;

ival2 = 12;

fval2 = 60.0;

int tnum = ival2;

System.out.println("Initial values are:");

System.out.println("Integer ival1 = "+ival1);

System.out.printf("Integer ival2 = %d\n",ival2); // note how it is printed

System.out.println("Floating point fval1 = "+fval1);

System.out.printf("Floating point fval2 = %5.2f\n\n",fval2);

ival2 = ival2 / ival1;

System.out.println( "orig ival2/ival1 = "+tnum+"/"+ival1+" = "+ival2 );

fval2 = tnum / (double)ival1;

System.out.println( "orig ival2/(double)ival1 = "+tnum+"/"+(double)ival1+" = "+ fval2 +"\n");

ival2 = tnum;

//System.out.println("resetting ival2 value = "+ival2+"\n");

fval2 = ival2 / fval1;

System.out.println( "ival2/fval1 = "+ival2+"/"+fval1+" = fval2 = "+fval2 );

ival1 = ival2 % (int)fval1;

System.out.println( "ival2 % (int) fval1 "+ival2+"%"+(int)fval1+" = 1val1 = "+ival1 +"\n");

ival2++;

fval1 += ival2;

System.out.println("after ival2++: ival2 value = "+ival2);

System.out.println("after fval1 += ival2: fval1 value = "+fval1+"\n");

System.out.print("ival1 = "+ival1+"\t");

System.out.printf("& ival2 = %d\n",ival2); // note how it is printed

System.out.print("fval1 = "+fval1+"\t");

System.out.printf("& fval2 = %5.2f\n\n",fval2);

System.out.println("fval1 >= ival1 is "+( fval1 >= ival1));

System.out.println( "fval2 != ival2 is "+(fval2 != ival2));

**2.a)** If you execute this program, what will be printed in the output window? Show all of the output in your answer including the final Java message. **(5 points)**

**2.b)** The operator *+=* is an abbreviation for what actions? Write this in words. **(5 pts)**

**2.c)** Rewrite the statement *fval1 += ival2;* to use only assignment and addition operators. **(5 pts)**

**Put your answers for 2.a - 2.c in your answers file.**

**Part 3:** The code below does not work. Create a new Java program called ***Lab2Part3.java*** starting with this code and then modify the code as directed in question 3.c) and with the restrictions given.

// code starting here should be put in main

double x1 = 8;

double y1 = 2;

double x2 = 2;

double y2 = 10;

double x3 = x2, y3 = y1;

int base = Math.abs(x1 - x2); // Line 1

int height = Math.abs(y1 - y2); // Line 2

double hypotenuse = Math.sqrt(Math.pow(base,2)+ Math.pow(height,2));

double area;

area = ; // Line 3 - Look up the formula for area and insert it here

System.out.printf("The base is length %d and the height is %3d \n", base, height);

System.out.print("The distance between ("+x1+","+y1+") and" );

System.out.printf("(%.0f,%2.0f) is the hypotenuse: %4.1f \n\n", x2, y2, hypotenuse);

System.out.print("The vertices of the triangle are : ");

System.out.print(" ("+x1+","+y1+"), ");

System.out.print(" ("+x2+","+y2+"), and ");

System.out.print(" ("+x3+","+y3+")\n\n");

System.out.println("The area of the right triangle is "+area+"\n");

double perimeter = ; // Line 4 - Insert correct calculation here and other as needed

System.out.println("The perimeter of the right triangle is "+ perimeter);

**3.a)** List each different type of error, giving the NetBeans error message, that you find after typing this program into Netbeans exactly as given. (Do not copy and paste.) Put these in your answer file. **(4 pts)**

**3.b)** Also, in your answers file, for each error, explain exactly (only a few words are needed) why the code, as given above, does not work, i.e. why did each error occur? **(4 pts)**

**3.c)** Now correct all the errors and run the program, debugging until it works properly. Only four lines of the program need to be modified to remove the compiling errors including one of the output statements. Be sure to follow any restrictions specified below and in the comments.

Restrictions:

* + You may not ADD any lines (if there are needed *import* statements, they may be added)
  + You may ONLY modify Lines 1, 2, 3, and 4 as labeled below
  + you may NOT change the data type of variable base in Line 1
  + you may NOT change the data type of height in Line 2
  + you must add the area formula for Line 3
  + you must add the perimeter formula Line 4 so that it calculates the perimeter
  + you may remove comments that are not needed after you have corrected the program

Once the program works**, save it as Lab2Part3.java . (8 pts)**

**3.d)** In your answers file, show the EXACT output that is produced by the program after it is fixed. **(6 pts)**

**3.e)** Fill in the missing variable names in the Java statement below that would allow it to print the lengths of the sides using variables that are already calculated in the code above. **(3 pts)**

System.out.println("The lengths of the sides of the triangle are "+

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ +", "+\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_+", and "+\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_);

**Put the answers for 3.a, 3.b, 3.d, and 3.e in your answers file.**

**Part 4:** Given the information below, write a new Java program called ***Lab2Part4.java*** to implement this algorithm.

This program is going to take time values in military time format, round those times to the closest quarter hour, and then determine how many total hours and quarter hours elapsed based on those times. The times will represent the times a person started work for a day, when they took their “dinner” break, when they returned from that break, and when they ended work that day. The military time format is a 24-hour format instead of a 12-hour AM/PM format. Military time is commonly written as four digits with no punctuation representing HHMM. The beginning of the day (12:00 midnight) is represented as 0000. 12:01am is represented as 0001. 1:00am is 0100. 11:59am is represented as 1159. 12:00noon is 1200 and 12:01pm is 1201. 1:00pm is 1300 and 1:59pm is 1359. 6:15pm is 1815. 11:59pm is 2359 and then 12:00midnight resets back to 0000. Therefore, while a 12-hour clock runs from 1200midnight to 11:59pm, a 24-hour clock runs from 0000 to 2359. When reading a military time, the 00 hour is typically read “Oh oh”, 06 hour is “Oh six”, 09 hour is “Oh nine” then 10 is “ten” and the other hours are read normally, i.e. 21 is “twenty-one”. Also when reading military time, the minutes are read normally except for minute 00 which is read as “hundred”. Therefore 0600 is “Oh six hundred”, 0935 is “Oh nine thirty-five” (or sometimes just “Nine thirty-five”), and 2246 is “Twenty-two forty-six”.

For this program the four variables labeled *exact*… will be the exact times that the user input for the day. We will start with the times as given above to do the calculations. The program will take the exact times and then round them up OR down to the closest quarter hour, i.e. if the exact time is 1048, then the closest quarter hour is 1045. The ranges for rounding are based on the minutes digits from HHMM as follows:

If MM is 00 minutes to 07 minutes then round down to 00

08 minutes to 22 minutes round to 15

23 minutes to 37 minutes round to 30

38 minutes to 52 minutes round to 45

53 minutes to 59 minutes round up to 00

Remember that when you are rounding from 53 to 59 minutes that the hour will also increase.

The rounding section will take in each military time value and return the closest 15 minutes increment to that value. The rounding will use the ranges given above to determine which values round to which increments. The code 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 saved as the recorded time that corresponds to the input exact time. If the minutes value is invalid, then the minutes should be set to 0. Since there are four exact times then there will be four sections of code to do the rounding.

To start this lab, in your main routine, do the following:

Define a variable called *exactStartTime* and give it a value of 754 . //\*\*not 0754

Define a variable called *exactBreakStartTime* and give it a value of 1136 .

Define a variable called *exactBreakEndTime* and give it a value of 1214 .

Define a variable called *exactEndTime* and give it a value of 1558 .

Create four new double variables called *recStartTime*, *recBreakStartTime*, *recBreakEndTime*, and *recEndTime* where *rec* stands for “recorded”.

\*\* Note: Cannot put 0 on front of number in code or it is interpreted as OCTAL not DECIMAL

Each of the four exact times should be rounded and then saved as the corresponding recorded time. Once the four exact times are rounded, the program will take the recorded times and calculate the total time worked by finding the time worked from start time to break start and then the time from break end to end time and summing these two values.

Think about how to find the amount of time worked from the state time to the break start time to get the first half of the day. If you have to do subtraction then how do you adjust the difference if the minutes value is negative? Make sure that the difference will be the total hours and minutes worked in 15 minute increments not greater than 45.

Once you have the first half of the day and the second half of the day calculated, then sum these values to get the total hours and minutes worked in 15 minute increments not greater than 45. Make sure to handle the cases where the minutes’ sum is greater than 45 so that the final value represents hours and minutes, where minute is 45 or less.

Once you have the total hours and minutes worked, then the minute increment (00, 15, 30, or 45) will need to be changed into a fraction or decimal representing the portion of the hour it represents, i.e. 30 minutes is ½ an hour. Once your program has the number of hours and the portion of an hour, this number will be multiplied times an hourly rate to calculate the pay for a single day.

As output the program should print the exact input times, the recorded times, the total hours and minutes worked, the total hours and fractions of hours to be paid, the hourly rate, and the final daily pay. This information should be printed in understandable lines and sentences like the ones shown below so that the user of the program understands the meaning of the output values.

Example output could be (don’t worry about the indentation level of the output) :

Exact Start Time is 754

Exact Break Start Time is 1136

Exact Break End Time is 1214

Exact End Time is 1558

Recorded Start Time is 800

Recorded Break Start Time is 1130

Recorded Break End Time is 1215

Recorded End Time is 1600

Total recorded is 7 hours and 15 minutes.

Total hours to pay is 7.25

Total pay for this day is $59.81 at $8.25 per hour.

Or something like this

Exact Start Time is 754 and Recorded Start Time is 800

Exact Break Start Time is 1136 and Recorded Break Start Time is 1130

Exact Break End Time is 1214 and Recorded Break End Time is 1215

Exact End Time is 1558 and Recorded End Time is 1600

Total hours recorded in the form of hhmm is 715

Etc…

**4)** Save your working program as ***Lab2Part4.java* . (20 pts)**

**Part 5:** Given the program you created for Part 4, modify that program in the following ways.

**5.a)** Produce formatted output that has the same titles and is spaced EXACTLY like the output below. Your font does NOT have to be the same as the one below but your font MUST be a monospaced fixed-width font. The first title word “Time “ (with the blank space at end) is in an 8 char space, then there is an empty 8 char space, then the word “Start” in 8 char, etc. . You must use *printf* statements to get exact spacing to make this table. All of the table fields are 8 spaces in size for both the strings and the numbers. **Save your working program as *Lab2Part5a.java* . (8 pts)**

Time Start End

Data: Start Break Break End

Exact 754 1136 1214 1558

Recorded 800 1130 1215 1600

Total

hhmm 715

Hours & 7

Minutes 15

As hours 7.25

At rate 8.25 per hour

Pay $ 59.81

**5.b)** Modify your code from part 5.a such that the output below is produced exactly. This means that the values for the first 4 variables are changed in the code such that the program produces the table below: **Save your working program as *Lab2Part5b.java* . (4 pts)**

Time Start End

Data: Start Break Break End

Exact 919 1058 1214 1832

Recorded 915 1100 1215 1830

Total

hhmm 800

Hours & 8

Minutes 0

As hours 8.00

At rate 8.25 per hour

Pay $ 66.00

**5.c)** Accept user input values for the original input times instead of hard coding the values into the program. Initialize the variables in the program with 0 then take user input to get the values. **Save your working program as *Lab2Part5c.java* . (6 pts)**

**5.d.i)** Give one different set of exact input values that will produce a total pay of $82.50 and

**5.d.ii)** Give one additional different set of exact input values that will produce a total pay of $76.31. **Put your two sets of input values in your answers file. (4 pts)**

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

**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.