SOFTWARE DEVELOPMENT LABS

WEEK 2 LAB 4

Program Design Sheets should be completed for all problems except 1.

Create a new package called **Lab4** to store all the java programs you create in this lab class.

ATTEMPT AS MANY AS YOU CAN !

1. Type in and run **Tax3.java** discussed in lectures. Test the program with the data below. Note what each piece of data is testing.

Gross Pay	Why is this being tested?	Expected Outcome	Actual Outcome
£60,000	Gross Pay is MORE THAN the High Tax Point	Tax = (60,000 – 40,000) * 0.40 + (40,000 – 10,000) * 0.20 = 8,000 + 6,000 = 14,000	
£40,000	Gross Pay is the SAME as the High Tax Point		
£30,000	Gross Pay LIES BETWEEN the High Tax Point And the Low Tax Point		
£10,000	Gross Pay is the SAME as the Low Tax Point		
£5,000	Gross Pay is BELOW the Low Tax Point AND more than zero		
£0	Gross Pay = 0		

2. Write a Java application called **OddOrEven.java** that prompts the user to enter an integer, then test whether the number entered was **even or not** and output a suitable message informing the user of the result.

   You can make use of the % operator. (number % 2 == 0) tests whether a number is even. If this evaluates to true, then number currently holds an even value (such as 2, 4, 20). If false, then number currently holds an odd value (such as 1, 15, 99).

   Test your program with the values -10, -5, 0, 35, 60.

Input	Expected Outcome	Actual Outcome
Number = -10		
Number = -5		
Number = 0		
Number = 35		
Number = 60		

3. Write a Java application called **SquareRoot.java** which reads in a whole number from the keyboard. If the number is greater than or equal to zero, the program should calculate and print the square root (to 2 decimal places), otherwise indicate the number has no square root. The Math class has a method to calculate the square root.

   The program should output a suitable message such as, either:

   ```
   The square root of 20 = 4.47
   ```

   or `This number has no square root`

   Test your program with the values 64, 121, 1600, 99, 0, -36, -100.

Input	Expected Outcome	Actual Outcome
Number = 64		
Number = 121		
Number = 1600		
Number = 99		
Number = 0		
Number = -36		
Number = -100		

4. Write a Java application called **GuessNumber.java**. This program should
   - prompt the user to guess a number via the keyboard,
   - compare the user's guess with a pre-set value (stored as a constant),
   - inform the user if their guess was correct.

   Test your program with suitable data.

5. Write a Java application called **TestNumber.java** which reads in an integer value (number) and tests whether the integer has a value **greater than 20** (declared as a CONSTANT). The program should output a suitable message such as, either:

   ```
   The number 45 is greater than 20
   ```

   or `The number 10 is not greater than 20`

Test your program with the numbers 25, 50, 20, 0 and -6.

Input	Expected Outcome	Actual Outcome
Number = 25		
Number = 50		
Number = 20		
Number = 0		
Number =-6		

Edit the program so that the number entered is compared to 0 and test with the same data

Input	Expected Outcome	Actual Outcome
Number = 25		
Number = 50		
Number = 20		
Number = 0		
Number =-6		

5.1 Extend your previous program so that it will test if the number entered is **greater than 0**, **exactly equal to 0** or **less than 0**. Output a suitable message informing the user of the result.

This is slightly more complex. In this case we have 3 potential options so we are not dealing with an either/or situation but a more complex case. Test your program with the numbers 15, 0 and -6.

Input	Expected Outcome	Actual Outcome
Number = 15		
Number = 0		
Number = -6		

6. Write a Java application called **Pay2.java**. This program should read in a person's name and the number of hours they have worked in a week, then calculate and output their total pay for the week (to 2 decimal places). Total pay is calculated as follows:

   Basic hourly rate = £5.95. Overtime hourly rate = £8.50.

   The basic hourly rate is paid for the first 40 hours, then the overtime rate applies.

   The output should have the following format:

   ```
   Janet, your total pay for this week is: £323.00
   ```

   Test your program with the following data:

Input	Expected Outcome	Actual Outcome
Hours = 50	40 * 5.95 = 238.00 10 * 8.50 = 85.00 Total = £323.00	
Hours = 40		
Hours = 30		
Hours = 10		
Hours = 0		

7. Write a Java application called **CheckLetterShape.java**. This program reads a single character from the keyboard (uppercase) and outputs information according to the following rules:

A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z	**A STRAIGHT letter**
C, O, S	**A CURVED letter**
B, D, G, J, P, Q, R, U	**A MIXTURE letter**

   Test your program with suitable data. How would you amend the program so that regardless of which letter is typed in, it would always inspect the uppercase version of that letter?

8. Write a Java application called **CheckLetter.java** which prompts the user for a character, checks if the character is the letter ‘g’, then prints out an appropriate message e.g.

```
The character entered is 'g'
```

8.1 Amend the program to check if the character entered is a lowercase letter (ie between ‘a’ and ‘z’), then prints out an appropriate message e.g.

```
The character 'g' is a lowercase letter

The character 'G' is not a lowercase letter
```

9. Write a Java application called **DaysInMonth1.java** which reads in the year (e.g. 2003) and the month (e.g. 3) and prints out a message indicating how many days there are in that month (e.g. There are 31 days in month 3, 2003). Make sure you research the rules for days in February (month 2) in leap years. Do not use the built-in Java method to check for a leap year.

Test your program with the following data:

Input	Expected Outcome	Actual Outcome
month = 9 year = 2005		
month = 7 year = 2013		
month = 15 year = 2012		
month = 2 year = 2012		
month = 2 year = 2000		
month = 2 year = 1900		

9.1 Create a new Java application called **DaysInMonth2.java**. Copy your code from **DaysInMonth1.java** but use the Java built-in method to check for a leap year.

Test your program with the above data.

CASE STUDY

Do this in Groups!

10 Write a Java application called **StudentMarks1.java** that will, for a single student:

- Prompt the student to enter their name (and then read in the student's name).
- Prompt the user to enter 2 values representing **Coursework** performance and **Examination** performance as percentages (and then read in these 2 marks). Assume the marks entered are in the range 0..100.
- From this, calculate the student's overall **module mark** as the average of these 2 marks.

The user should then be informed of his/her profile in terms of:

Name	**Coursework**	**Examination**	**Overall**
student name	*mark*	*mark*	*overall module mark*

Example output (note the heading line is printed out first and this is followed by the relevant data):

Name	**Coursework**	**Examination**	**Overall**
Fred Smith	50	16	33

10.1 Amend the program to include the student's grade.

Assume for now that there are just two Grades – Pass or Fail. A Pass grade is awarded whenever a student gets an overall mark of at least 40. Example output:

Name	**Coursework**	**Examination**	**Overall**	**Grade**
Fred Smith	50	16	33	Fail

OR

Name	**Coursework**	**Examination**	**Overall**	**Grade**
Mary White	50	59	55	Pass

10.2 Amend the program written to introduce a third grade – Distinction.

A Distinction is awarded when a student has an overall module mark of 70% or above.

Name	Coursework	Examination	Overall	Grade
Fred Smith	50	16	33	Fail

OR

Name	Coursework	Examination	Overall	Grade
Mary White	50	59	55	Pass

OR

Name	Coursework	Examination	Overall	Grade
Ali Brown	70	90	80	Distinction

10.3 Students must achieve at least **40%** in **each component** – otherwise the module is deemed to be **failed** and the student is awarded a **Fail** grade. If a student does not fail the module then their grade is determined **solely** by the **overall module mark** according to the following:

Overall Mark (Range)	Grade
40% – 69% (inclusive)	Pass
70% – 100% (inclusive)	Distinction

Hence the following examples would apply:

Student	Coursework	Examination	Module Mark	Grade
Sean Allen	55	45	50	Pass
Mary Briggs	40	40	40	Pass
Alan Chambers	90	60	75	Distinction
Alison Dooley	30	25	28	Fail
Patrick Evans	40	39	40	Fail
Colin Fisher	30	90	60	Fail

Amend the Java Application so that the student is also informed of the Grade achieved according to these rules.

HINT:

- Establish the student's overall **mark**

  You need to output this value regardless of the grade decision

- You now have to establish the grade

  Ensure that both marks are at least 40%

  If they are **not** then the grade is a **Fail**

  If they are, further tests on the overall mark are required to establish Pass or Distinction

Test your program with data in the table above.