SdPd/java Lab Exam 1

G-Uber offers taxi services to customers. The table hereunder outlines the various fees.

Initial Charge: Standard (off-peak times) Premium (20.00h - 8.00h & Sundays/Public holidays)	**Price:** € 2.10 € 2.90
Passengers: Each additional passenger	 €1.00
Fare charge categories: A - Standard fee for trips up to and including15 km B - Standard fee for trips longer than15 km C - Premium fee for trips up to and including15 km D - Premium fee for trips longer than15 km	**Price per kilometre:** € 1.10 € 1.45 € 1.40 € 1.65
Credit card charge: 3.00, 4.00, or 5.00	**Rates:** Minimum of 3.00% Maximum of 5.00%
Accommodations: Buggies, bikes, luggage, shopping (non-human cargo) **y**es/**n**o	**Price:** **y** - € 10.00 **n** - € 0.00
Discounts: Long distance traveller	**Discount Rates:** Trips over 50 km 5% Trips between 100 km-150 km 6% Trips over150 km 7%
VAT/Tax Rate Vat is charged on all transactions	 23.00 %

Sample Line Input & Explanation:

	Pas	Dis	Fare	CCard	Acc
	(int)	(double)	(char)	(double)	(char)
E.g.	1	16.5	P	3	Y

Pas – Passengers: the number of passengers – integer
Dis – Distance: the distance of the trip in kilometres – double
Fare – Fare: Premium (P) or Standard (S) – char
CCard – Credit Card: credit card charges – double
Acc – Accommodations (Special Requirements) luggage, bikes, buggies – char

Develop a Java program to enable a G-Uber employee to input data for a number of trips made daily. Allow the program to calculate individual fares and additional charges associated with the services on offer.

1. **Download** the LabExam1 **zip** file and extract the folder, **Saved** on the root of the u-drive (**not** on a USB or the desktop).
    - Rename the **LastNameFirstName2017LabEx1** folder & java file as per your own name.
    - For example **AgnewGerry2017LabEx1** folder and **AgnewGerry2017LabEx1**.java program file.
    - To be **verified** by your lab supervisor.
    - Remember to rename the starter **class name** as per your java program file name.

2. Add your **Program Id, Name** & **Program Description** as comments at the top of the program.

3. **10%** of the Lab Exam marks are for the Algorithm sheet (enter your name at the top of the first page) which must be submitted at the end of the lab exam.

4. **Warning:** marks will be deducted for **bad programming practices** such as:
    - Lacking meaningful variable names, white-space, indentation, etc.
    - Ensure redundant code is deleted prior to program submission.
    - Ensure that non-working code is commented out prior to program submission.
    - Otherwise severe penalties will be incurred.

5. **Constants:**
   Declare the 10+ necessary constants as appropriate with meaningful names, data types and values.

6. **Variables:**
   Declare any necessary variables as appropriate with meaningful names and types.

7. **Initialise:**
   Initialise any necessary variables such as counters and totals (not all of the variables).

8. **Preliminary Input**: – see screenshot 1 (page 5)
   Ask the user how many trips he/she will be inputting details for.
   Allow the user to enter the number of trips for which he/she wishes to enter details, via the keyboard, which can vary from day to day.

9. **Multi Item Line Input:** (inside the **for** loop) – see screenshot 1 on page 5.
    - Allow the user to enter trip details, via the keyboard, on the same input line
    - Refer to the Sample Line Input and Explanation on page 1.

10. **Calculations:** – see the table on page 1
    - Calculate the basic fare for each trip based on whether the distance travelled is 15 kilometre or less, or greater than 15 kilometres and whether the fare is based on the standard or premium rates. This should be added to the appropriate initial charge (standard or premium). The basic fare comprises the appropriate kilometres rate multiplied by the distance in kilometres plus the initial fee.

- Calculate the cost of additional passengers (the first passenger does not incur this charge). If/where there are more than one passenger multiply the number of passengers (minus 1) by the passengers' fee in the table on page 1.
- Calculate the cost of accommodations/special requirements. If these are requested account for the associated fee.
- Calculate the full fare: the basic fare plus the charge for additional passengers plus the charge for special requirements.
- Calculate the credit card charge (where applicable). This is the full fare multiplied by the credit card rate (entered by the user) divided by 100.
- Calculate the discount which is determined by the kilometres travelled. The discount rates relevant to kilometres are outlined in the table on page one. The discount amount is achieved by multiplying the full fare by the appropriate discount rate.
- Calculate the net price. This comprises the full fare plus the credit card charge less the discount amount.
- Calculate the VAT. This is the net price less VAT multiplied by the vat rate provided in the table on page 1.
- Calculate the final price. This is the net price plus vat.

11. **Selection: –** basic fare (if /else if /else) with logical operators to determine fare type and distance. Determine the fare category – premium or standard (p, P, s, or S) and whether the distance is 15 kilometres or less or greater than 15 kilometres. Use the selection to assign the appropriate rate and calculate the basic fare.

12. **Selection: –** passengers (**if/else**):
    Determine the number of passengers over 1, calculate and assign the appropriate fee.

13. **Selection: –** discount (**if/else if/else)** with logical operators to designate (or not) the appropriate discount for trips based on the distance of the trip.

14. **Selection: –** accommodations/special requirements **(if/else**):
    Determine whether or not accommodations/special requirements were requested and assign the appropriate fee.

15. **Line Output: –** see screenshot 1 on page 5.
    - Output the sequential trip number, basic fare cost, passenger fee, accommodations, full fare, credit card charge, discount amount, net price, vat cost and final price.
    - Develop unformatted line output initially with a println ( ) statement and then formatted with a printf ( ) statement for each trips details.
    - Both unformatted and formatted versions are required.
    - Comment out the unformatted version when the formatted one is working (do not delete the println() statements).

16. **Header Output: –** see screenshot 1 on page 5.
    - Display the program headers including your name aligned as specified.

— Using a println ( ) rather than a printf ( ) statement.

17. **Footer Output/Totals:** – see screenshot 1 on page 5.
    - — Display program footers aligned as specified using both println() and printf () statements
    - — Initialise, accumulate and output formatted cost totals for the total basic fares, total passenger fees, total accommodations/special requirements requests costs, total for full fare cost, total accumulated for credit card charges, total discount amounts, total net prices, total vat accumulated and the total final prices.

18. **Trip type counters** – see screenshot 1 on page 5.
    - — Use counters to determine the number of trips that occurred in the various 4 categories (A, B, C and D) and output the details.

19. **Largest and smallest:** – see screenshot 1 on page 5.
    - — Determine and output details of both the largest and smallest trip fares and their corresponding trip number.

20. **Average discount**: – see screenshot 1 on page 5.
    - — Determine whether discounts were granted and, if so output details of the average discount amount. Alternatively, should no discounts occur, output a message to state that there were no discounted fares.

20. **Case Insensitive Character Processing:**
    - — Enhance the program to accept and handle both upper and lower case character input.

21. **Compile (routinely)**
    - — Develop your solution incrementally ensuring you have debugged all errors as you progress through the solution.

22. **Save – The End:** When finished Save and Exit TextPad
    - — Zip (R/click: Send ➔ Compressed Zip) your **LastNameFirstName2017LabEx1** folder
    - — Upload your **LastNameFirstName2017LabEx1** zip file to the LearnOnline/Moodle link provided
    - — To be **verified** by your supervisor **before** you **submit** the zip file
    - — Submit your Named Algorithm sheet before you exit the lab
    - — Ensure that you have signed the **attendance sheet** before you exit the lab

Screen shot 1 – Sample output