CS 110 Lab Sheet 3

Lab cycle starting 16th October 2018, Normal Deadline: two weeks after your lab in that cycle.

General Guidance

Make sure you *save* each program you write *separately*. *Don't* just overwrite an old one with a new one. Having date-stamped files containing your code is helpful in proving you have actually completed the task if, for any reason, there is confusion about it later. (In fact it's always a good idea to keep copies of your work in this way – just in case.)

Some of these tasks are quite fiddly (lots of programming is). So remember to carefully read the hints where there are any, and remember the postgrads are not just there to sign off tasks so *ask for help.*

Car Running Costs Example

The first assessed task on this sheet is to build a car running costs example - you will work out how much it costs to own and drive a car (very simplified) based on some of the costs you can expect.

The program will be built in stages - you need to complete each stage and then combine them to make the final program, which you need to show to me.

Stage 0: Car Running Costs Estimator

Make sure you've read chapter 2 of the notes - and watched the relevant videos.

Stage 1: Car Running Costs Estimator

This example starts to give you practice in turning problems into programs - first you need to turn an example in words into something mathematical (a simple expression). Then you can turn that into a simple program. This is based on the same ideas and concepts as the Swimming Pool estimator in Chapter 2.

You want to find out how much your car costs to run. A simplified model is:

- **Cost to buy** - How much did it cost to buy the car?
- **Cost to service** - A *service* is routine maintenance that is done at fixed intervals for a fixed cost. The typical cost is a few hundred £ or € for a service - more expensive cars cost more - the more expensive the cost of a service, the more expensive the car is to run. Instead of just driving until your car breaks, you get it serviced at regular intervals before something goes wrong to, hopefully, stop that happening. (Understanding this concept caused some people a bit of trouble last year - if you don't get understand it, ask me.)
- **Service interval** - Cars have a fixed recommended distance they should be driven between services. The shorter this distance, the more expensive the car is to own because it means you will need to get a service more often. Typical service intervals are between 15 000 and 36 000 km
- **Km/l** - How many km can it go on average per litre of fuel (I know you don't measure fuel economy in km and litres this way, but it's simpler - if you are used to the 'normal' way using l/100km then you can do that instead, just tell whoever is signing off the work)
- **Fuel cost** - How much does the fuel cost?

Rationale - an old car may be very cheap to buy; but it might cost a lot to service (because older cars could not go so far between services, and more goes wrong so it costs more) and use more fuel (older cars generally do). So if you drive a lot, it may be better to pay more for a newer car, because it will cost you less to run it; and conversely, if you don't drive much, a cheap old one may be better. Similarly, an old car that uses expensive fuel (petrol/gasoline?) will be more expensive if you drive it a long way than a newer one that uses cheap 'fuel' (electricity?) - but that may not matter if you don't go far.

Work out a formula for how much it would cost to run a car for, say, 100 000km in terms of the costs in the list above. Don't do it in Java to start with - just treat it as a mathematical (algebraic) expression. As well as the costs above, your formula will include the total number of miles the car is driven.

Hint: In the past the main mistakes people have made with this are about not thinking the problem through properly. Make sure you completely understand the problem before you start - even though it seems simple.

Hint 2: You might want to think about doing this with a spreadsheet to start with.

Hint 3: *Do it in stages* - just think about how to calculate the fuel costs first; then think about how to calculate the service costs; then think about adding them all up (with the cost of the car as well).

Hint 4: If you are in the least doubt about it, get one of the lab assistants to check your formula.

Stage 2: Turn it Into Java

Take your formula and turn it into Java - you can just use fixed numbers for the values listed below for now.

Car 1 compared to Car 2

	Cost	Fuel Cost per litre	Km/litre	Service Interval	Service Cost
Car1	12000	1.1	20	15000	100
Car2	18000	1.4	35	25000	200

Your program should consist of one line that looks like this:
`System.out.println(`*`formula`*`);`
where *`formula`* is your arithmetic expression representing the cost of car ownership.

Stage 3: Add in Variables

Rewrite your program so that it uses the variables *carCost, serviceCost, serviceInterval, kmPerLitre* and *fuelCostPerLitre* instead of numbers. To do this you will have to declare them as *double variables* and assign them the values you used in the previous task.

At this point you should have a program with variables representing each of the columns in the table above: cost, fuel cost per litre, km/litre, service interval and service; plus a variable representing the total distance you have driven the car. Your program should assign values to each of these variables - you can choose whatever numbers you like to test your program. After your variable declarations, your program should have a variable that computes the formula for the cost of owning the car; and it should then print out that variable.

Assessed Stage 4: Work out the Cost for Two Possible Cars

Look at the table above - obviously, if you buy car 2 and never drive it, it will cost you a lot more than if you bought car 1. Also, fuel and services cost more for car 2. BUT car one can go much further using the same amount of fuel, and also can go much further between services. So, maybe, if you drive it far enough, car 2 may end up being cheaper to own than car 1.

Extend your program so it calculates the cost of owning two cars, and use the values in the table above for the various costs. Run your program using different values for the total mileage the cars are driven - changing the total distance driven each time. Start at 100 000km, then 200 000km, 300 000km, etc. - until you either reach the point at which car 2 become cheaper than car 1; or you reach 1000 000 km. You should then, when you get the task signed off, be able to answer the two questions: (a) does car 2 ever get cheaper to own than car1? (b) if so, at what distance (to the nearest 100 000km) does that happen?

Challenge Task

Print out a table of the cost of ownership for both cars, starting from 50 000 km and going to 300 000 km, increasing in steps of 25 000 km for each row of your table.

Challenge Task

Repeat the first challenge task but use a loop

Challenge Task

Repeat the previous challenge task but give the use an option to enter the minimum and maximum possible mileage, as well as the size of the steps, and the name of the type of car.

Challenge Task

Improve your previous program so it doesn’t crash if a use enters non-integer values.

Challenge Task

Improve your program so it ‘sanity checks’ the input by not allowing implausible values - what counts as ‘implausible’ is up to you, but an obvious example is negative numbers.