Title: Intro to Java lab

The purpose of this lab is to get used to your Java IDE, get working classes code working etc.

Open up Eclipse or IntelliJ whatever your IDE of choice is on your machine. on your machine. The instructions below are for Eclipse – but just find the equivalent if you are using a different IDE. There is also an Intro to Eclipse video in the lab directory showing steps to set up a java project.

(Note: Eclipse will ask you to set the **workspace** (this is the directory where you want your code will be stored).

1. Creating a Control java class to run your java code.

This part is setting up a class with the "main"method as discussed in class

Create a java **project**: In Eclipse – do File/New – java project. Give it a name. Create a **package** within the project: Right click on "src" folder and create a new package. Name the package **`com.lab1.test`**

Create a java **class**: From the package, create a new class. Call it "Control" – as we're going to use it to control /run things – it isn't a typical class with attributes.. etc.

First put a comment header block into your new java class at the top to explain what it is e.g.

```
/**************************************
*   Control: the purpose of this class.. etc
*  Author:
*  Date:
**************************************/
```

Write java code in your Control class to simply print a sentence e,g, "first lab" out to the console –

- You will need the "main" method – as discussed in class in your code as an entry point to your code. (this method can also be auto-generated for you in Eclipse as an option when you are creating the class).

- note: `System.out.println(" ");` is the java command for printing to the console.

Save your code into the src/package directory of your project.

Run your java class : by selecting Run/run from the menu or clicking on green run button arrow.

Check the console in the bottom pane to see your sentence printed out.

2. Create a Java Class

Create a new java class (Using File/ New/ Class..) in the same directory as the others. Call this class "Car"

In your "Car" class, add attributes - including owner name, registration number, maximum speed, colour, automatic (or not), number of wheels.

Add a **constructor** to your Car class to set up new Car objects, and that sets up the "owner name" attribute with a value and "registration" with a value.

Change the **main method in your Control class** so that it creates a new Car object (i.e. *instantiates* an object).

You don't "see" any result - as creating objects isn't producing any visible output.

3. Multiple constructors

Add **a second** constructor to your Car class – that sets up all attributes with initial values.

Now instantiate more Car objects, this time using the 2nd constructor you created.

4 toString()/Printing objects

From your main method in your Control class, try
`System.out.println (whateveryourobjectnameis)`
from the main `Control` class – to "print out" your object. What do you see? How useful is it as a representation of your object?

Note down what it is.

Now:
Add a method called "toString" to your Car class. Let's use this method to return a String result that shows a readable version of the attributes.
– by concatenating* the values of the attributes of the class, formatted with text e.g. "This Car has owner name X , has a registration plate of A, is of colour Y etc".

*Example of concatenation in java : `String summary = "This patient room number is name" + this.roomNumber;`

In your main method , instantiate a new Car object and add in code to System.out.println(whateveryourobjectnameis) , where object name if the object you just created. What do you see? Why?