# Object Oriented Programming 4: Week beginning 19/09/2016

Using NetBeans IDE. See

<http://www.netbeans.org/kb/docs/java/quickstart.html>

for short tutorial.

This takes 10 minutes, so take this tutorial before trying the exercises below.

Writing and debugging your own classes. See how to debug below.

Q.1. Implement a class **Circle** that has a property **radius** and methods **getArea()** and **getPerimeter()**. These methods should calculate the area and perimeter. Area = ∏\* radius2. Perimeter = 2 \* ∏\* radius. In the constructor, supply the radius of the circle. Write a main method to test the methods of this class.

Q.2. Implement a class **Car** with the following properties. A car has a certain fuel efficiency (measured in miles/gallon or km/litre – pick one) and a certain amount of fuel in the tank. The efficiency is specified in the constructor and the initial fuel level is 0. Supply a method **drive** that simulates driving the car for a certain distance, reducing the fuel level in the gas tank, and methods **getFuel**, returning the current fuel level, and **addFuel**, to tank up.

Sample usage:

Car myCar = new Car(30); //30 miles per gallon

myCar.addFuel(20); // tank 20 gallons

myCar.drive(90); // drive 90 miles

System.out.println(myCar.getFuel()); //Print remaining fuel – what output do you expect here?

Q.3 Simple algorithm to search a list of strings for a particular string: Linear search

Write method to search (Linear search) a list of strings for a particular string. It should have the following header:

public static int search(String [] stringList, String searchString)

The method returns the subscript in the array where searchString is found or -1 if it is not found.

Write a main method to test this method.

Q.5 Arrays has a binarySeach method. Write a main method to test this on an array of Strings.

### To debug your code:

1. Then press Ctrl-F8 to set a breakpoint.
2. Choose Run > Debug Main Project (Ctrl-F5). The IDE opens the Debugger windows and runs the project in the debugger until the breakpoint is reached.
3. Select the Local Variables window to see values of variables or Call Stack to see nested method calls.
4. In debug mode:

Step Over (F8). Executes one source line.

Step Over Expression (Shift-F8). Executes one method call in an expression

Step Into (F7). Executes one source line. If the source line contains a call, the IDE stops just before executing the first statement of the routine.

Step Out (Ctrl-F7). Executes one source line. If the source line is part of a routine, executes the remaining lines of the routine and returns control to the caller of the routine.

5. When the program reaches the end, the debugger windows close.