# Lab 1. Introduction to Java Programming Using Processing

## Revision:

During your pre-session activity you should have become familiar with the resources at **Processing.org** website (ensure this is bookmarked) and creating and saving various processing sketch files.

Read the webpage on Problem Solving

Support : See top of page for contact details : support tutor, unit tutors

Ask for support while undertaking the following core exercises.

## Learning Objectives

* Familiarity with Processing Environment
  + Writing and saving a program
  + Running a program
  + Correcting syntax errors
* Java syntax
* Simple problem decomposition – break problem into smaller chunks
* Program flow (order of commands)
* Using Variables and expressions
* Program sequence

## Resources:

open these and have them available for reference

* Lecture Notes (moodle)
* Processing.org website (add to favourites) and look at **tutorials**, particularly
  + https://processing.org/tutorials/coordinatesystemandshapes

**Exercise 1 : (Pair Programming). Draw a Car**

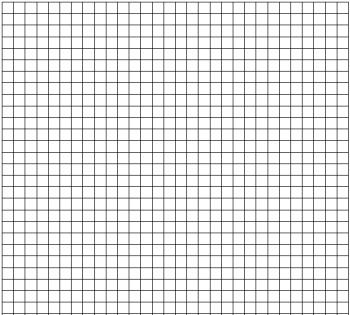
Open ***Processing*** P on your machine. You’ll see a text editor with “sketch” at the top and a text editor below (white background)

Using the **rect** and **ellipse** drawing commands, you must draw a car similar to the diagram below (with the same relative sizes and positions of each part). You can assume that the size of a square is 10 pixels, but you will need to set the size of your canvas to 300 by 300

size(300,300);

Pair programming involves one person writing the code, while the other makes suggestions. The first coder should get the boxes in the correct places, the 2nd will then take over to get the circles in the correct places. Draw a box , run the code to check, draw another box and check (repeat)

Your code should be clearly commented to show what each block of commands is drawing.



Press play to run the code

Save your code in Programming/Week1

**Exercise 2 : Variables**

**Exercise 2.** Enter the code below precisely (you can copy and paste). Press play

//A simple drawing Program

float x;

x = 50;

//screen size 500x500 pixels

size(500,500);

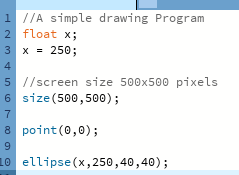
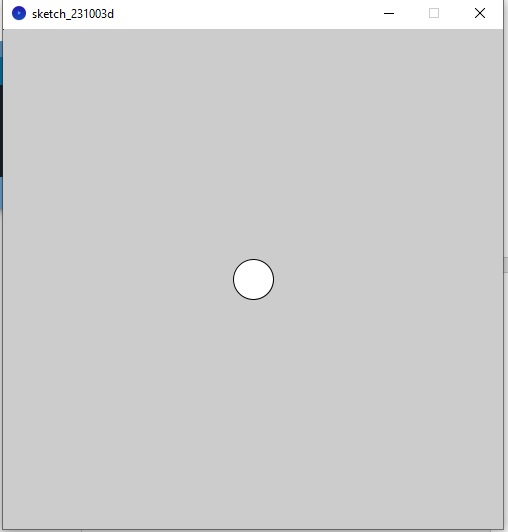
point(0,0);

ellipse(x,250,40,40);

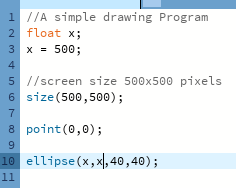
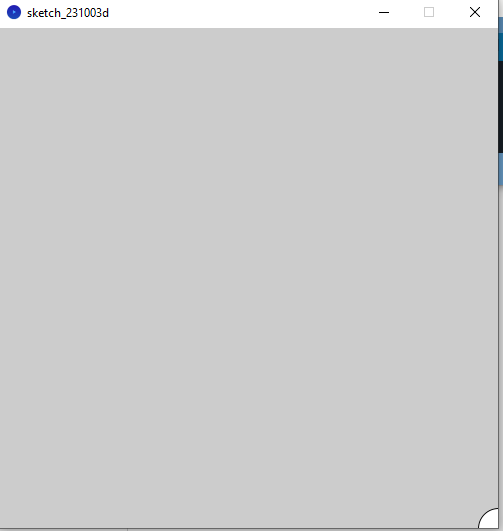
You should have made a ‘point’ (spot) at the origin corner (top left) and a white circle at position 50,250 with width and height of 40 pixels.

Now stop the program, this will clear the display.

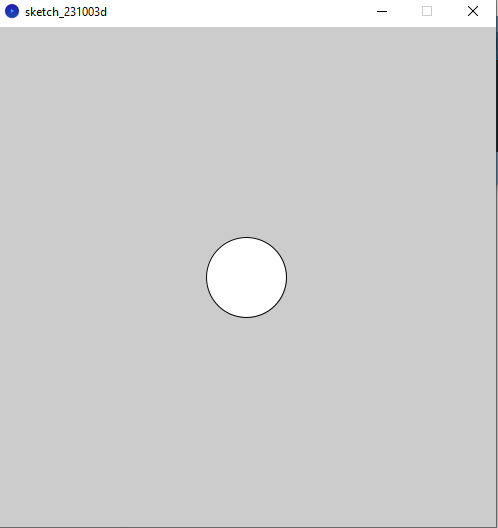
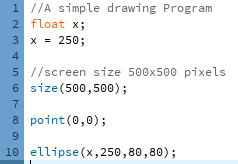
1. Make the circle appear in the middle of the screen (alter the value of variable **x**).

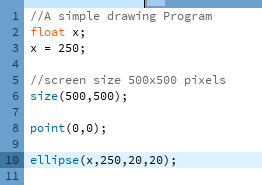
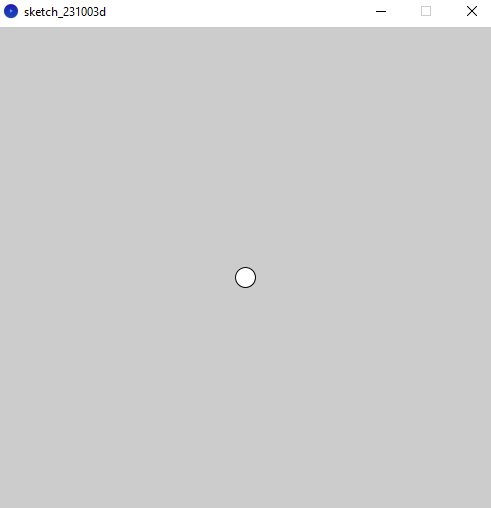


1. Make the circle appear in the bottom right corner of the screen (we should only see ¼ of it)



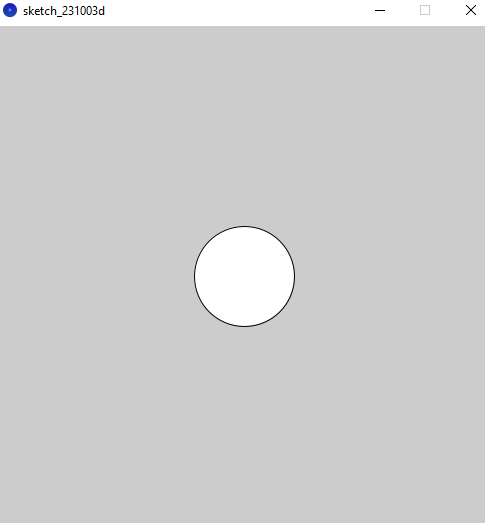
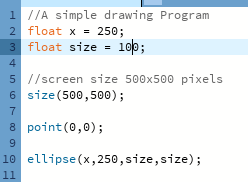
1. Make the circle double its original size, then ½ its original size





1. Introduce a new variable

float size = 50;



1. Use the **size** variable inside the ellipse command, so that, you can more easily alter the size of the circle, only changing one value in the program (rather than 2). Try this out.

Question : how does the compiler understand the difference between the **size** **variable** and the **size** **command** in our program?

Answer : commands are followed by brackets “( )”. These brackets tell the compiler that we are dealing with a command. We couldn’t use brackets or other symbols in a variable name – actually they should start with a letter, they are case sensitive, they can contain numbers but not spaces. The usual convention is camelCase, where each new word start with a Capital letter e.g.

float centreCircleX = 20;

**Exercise 3. drawing a square**

Draw a square of edge 50 pixels on the screen using only the **line** command, go clockwise. Line takes 4 parameters (2 pairs) as follows

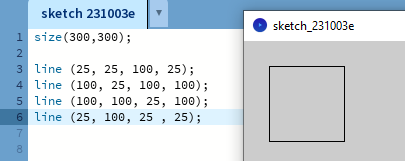
line (startX, startY, endX, endY);

Problem decomposition : square is made up of **4** parts

1. top edge - line
2. right edge – line
3. bottom edge – line
4. left edge – line

edges : 50 pixels long, so either startX and endX differ by 50, or startY, endY differ by 50

Add 4 comments, to your code one for each stage

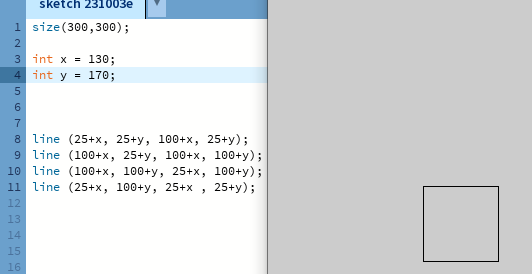


**Exercise 3.B.** Movable square

If we want to move the square… introduce a variable **x** to allow the square to be moved left or right on the canvas

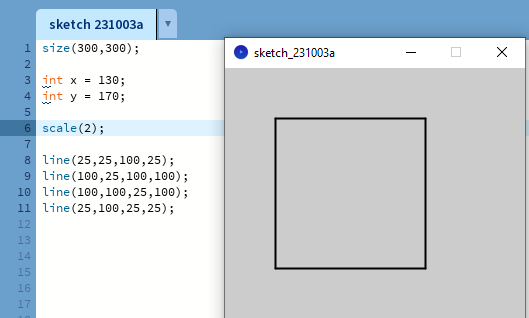
float x = 20; //x can store floating points or integer numbers, x becomes equal to 20

Introduce a variable **y** to allow the square to be moved up and down as well.



**Exercise 3.c** Scalable square

Finally add a variable called **edge** that allows you to vary the size of the square and try different values for edge. Alter x and y as well to ensure your square scales and moves dependent on the variable values.

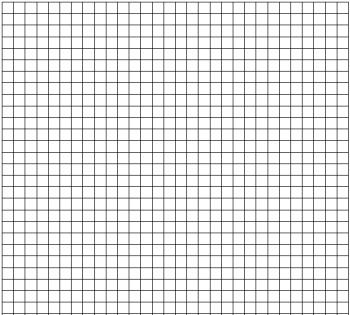


**Exercise 4. Drawing a Stick Person of variable size**

Using the 3 variables **x**,**y**,**size** and only the line and ellipse drawing commands, you must draw a stick person similar to the diagram below (with the same relative sizes and positions of each part) where x,y is the centre of the head and the value of size the length of the torso (centre of head down). Other sizes can be seen by counting the number of graph squares relative to size. You SHOULD add extra variables as required to store suitable starting points for the arms, legs etc.

Your code should be clearly commented to show what each block of commands is drawing.

Experiment with the values of size, x and y to draw different sized people in different places on the screen (remember that processing will allow you to draw off the screen, so if nothing appears alter x and y).



size

size

Save your program.

**Extension exercises** : tutorials at <https://processing.org/tutorials>, covers everything from basic drawings through to 3Danimation and machine learning. The Text tutorials cover animation and other ideas that we’ll meet shortly in the course.

If you are a more advanced programmer, see <https://natureofcode.com/book/>