# 02 Encapsulation Lab

## Part 1

For this part, we will be using an existing class called PShape. A simple example showing how to use a PShape object is given in by PShapeExample in the 02 Examples.zip file on moodle which you should open, run and examine. You should also refer to the PShape documentation: <http://processing.org/reference/PShape.html>

1. Open a new Processing sketch and create a function called createStar(). The form of the function should be:  
   PShape createStar(){  
   /\* … create the star… \*/  
   }

Note: do not draw the star inside the createStar() function. The function should just create a PShape, call methods on the shape such as fill and vertex, and then return the PShape. All points should be drawn around the origin (0,0), as PShapes are later translated when they are drawn using the shape() function.

1. Test your createStar() function by creating several PShapes in the setup() method and then drawing them using the draw() method of your sketch.
2. Modify your createStar() function so that it assigns a random fill colour to stars. You can generate a random colour any way you want (see the previous lab for an example).
3. Investigate the rotate() method of the PShape object (<http://processing.org/reference/PShape_rotate_.html>). Inside the createStar() method, randomly rotate the star so that the stars do not all have the same orientation. Note that angles for the rotate function are in radians, so the range is between 0 to TWO\_PI.
4. Investigate the scale() method of the PShape object (<http://processing.org/reference/PShape_scale_.html>). Modify createStar() so that the star is scaled randomly (e.g. make the star twice its size with probability 50%).
5. Finish off your program so that it creates and draws five stars at different positions on the screen, each star having a random colour, a random size, and a random orientation.

Show the program to your tutor for marking.

## Part 2

For this part, you will create your own new class. You can study the StickyNoteExample file from 02 Examples.zip as an illustration of something similar to what we will do in this part of the lab.

1. Create a new sketch, adding setup() and draw() methods to it. Also add an empty class called Tree to the sketch.
2. Add some properties to the tree class: trunk colour, foliage colour, width, height, and position. Use appropriate types and give the properties appropriate default values.
3. Add a draw() method to the Tree class. Test your method by creating three trees with different properties (especially different heights) inside setup() and drawing them inside draw(), just as the StickyNoteExample does for objects of the class StickyNote. Note: a simple tree can be drawn using a rectangle for the trunk and an ellipse for the foliage.
4. Create an arraylist of trees called forest. Generate twenty trees with different random positions on the screen, and put them all into the arraylist. Draw the trees by iterating over the forest list. (For an example of how to create and iterate over an arraylist of objects, see the InvadersExample from 02 Examples.zip).
5. Add a grow() method to the Tree class. The grow() method should increase the height of the tree by one pixel.
6. Inside the draw() method of your main program, after the forest is drawn, pick a random element of the list and call the grow() method on it. This can be done with the following statements:  
     
   int randomIndex = (int) random(forest.size());  
   forest.get(randomIndex).grow();  
     
   The result should be that a random tree is grown sixty times per second; so you should see the forest “growing”.

Demonstrate your forest animation and show your code to the tutor for marking.