# 04 Inheritance

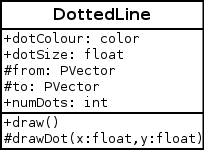
## Part 1

For this part we will be using the BlobExample from the 04 Examples.zip file on moodle. Extract and run the program. You should also have a read of the lerpColor() documentation and here: <http://www.processing.org/reference/lerpColor_.html>  
as this will be useful for you in Step 5 later.

1. This program defines two classes, Blob and RotatingBlob. The relationship between them inheritance, which is created by the statement:  
   class RotatingBlob extends Blob { /\* … details… \*/ }  
   Delete the extends Blob to remove the inheritance relationship and see if the program still works. Explain why you get any errors that you get. Restore the extends Blob when you are satisfied that you understand what’s happening.
2. The Blob class has a field called center that specifies the position of the center of the blob. At the moment, the blobs do not move at all so center always has the same values. Add a velocity field to the Blob class. This will involve the following steps:  
   (i) add a new field of type PVector to the Blob class called velocity  
   (ii) initialize velocity to a random 2D unit vector, which can be achieved using this line of code:  
   velocity = PVector.random2D();
3. Add a move() method to the Blob class. The move() method should simply add the velocity vector to the center vector in order to update the blob’s position. Note: do *not* add a move() method directly to the RotatingBlob class. Instead, we want to make RotatingBlob inherit move() instead.
4. Modify the draw() method so that it now calls the move() on each of the blobs after jitter() is called. Confirm by running your program that the move() method is inherited by the RotatingBlob class.
5. Create a new class, ShimmeringBlob, that inherits from Blob but instead of rotating, the blob slowly changes colour from its original colour to white then back to its original colour again. If you examine the draw() method of the Blob class, you should notice that the field fill is used to set the colour of the blob; you should therefore override the draw() method in your new class to adjust the value of fill and then call super.draw() to draw the blob. Don’t duplicate any drawing code! See the RotatingBlob class for an example of an overridden draw() method.

## Part 2

Although it is possible to draw a continuous straight line in Processing between any two points (*x1,y1*) and (*x2,y2*) using the line() function, there is currently no facility to draw *dotted* lines.

1. The following diagram describes a DottedLine class:  
     
   Implement this class and demonstrate it by drawing several dotted lines with different colours, dot sizes and numbers of dots on the screen. (Note that the modifiers ‘+’ indicates a public field/method while ‘#’ indicates a protected field/method.)   
     
   The class constructor should take values for from and to using parameters provided, with the rest of the parameters being set after the object is created , e.g.  
   DottedLine line = new DottedLine(100,150,300,300);  
   line.dotColour = #A6439A;  
   line.numDots = 10;  
     
   In order to calculate the position of a dot along the line, one can divide the difference between start and end by the number of dots minus 1, and then multiply that by the index of the dot to draw, and then add back the starting from position, i.e.  
   PVector dotPosition = to.get(); // copy ‘to’  
   dotPosition.sub(from); // subtract ‘from’  
   dotPosition.div(numDots-1); // gap between dots  
   dotPosition.mult(dotIndex); // mult. by dot num.  
   dotPosition.add(from); // add back ‘from’
2. Create a new class StarryLine that is just like DottedLine, except that it draws small stars instead of dots. Make StarryLine inherit from DottedLine, but override the drawDot() method. Do not duplicate any code from the DottedLine! Here is a class diagram depicting the situation:  
    