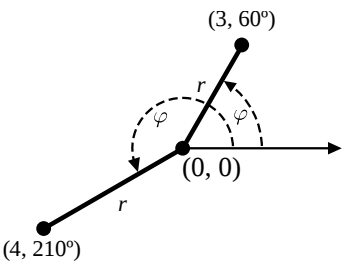
3. **Object Oriented Programming**

(a) With reference to Object Oriented programming, what is meant by the terms *Encapsulation* and *Data Hiding*? [3]

(b) The Polar Coordinate System is a two-dimensional coordinate system in which each point on a plane is determined by a distance from a fixed point and an angle from a fixed direction.

i. Design a class, `PolarPoint` that stores a fixed origin `Point` (that is shared by all instances of the `PolarPoint` class and is initialed using `Point origin = new Point(0,0);`) and a distance and angle from the reference point. You can assume that the angle is always relative to an infinite line extending to the right of the point (see diagram). [5]



ii. Converting between polar and cartesian coordinate systems can be done using the sine and cosine operations.

$$x = r\cos\varphi + x_{\text{origin}}$$

$$y = r\sin\varphi + y_{\text{origin}}$$

Write a method that returns a `Point` in the cartesian coordinate system. You may find the `Math.cos(double a)` and `Math.sin(double a)` methods useful. [3]

iii. Cartesian coordinates can be converted back to polar coordinates using

$$r = \sqrt{(x - x_{\text{origin}})^2 + (y - y_{\text{origin}})^2}$$

$$\varphi = \text{atan2}((y - y_{\text{origin}}), (x - x_{\text{origin}}))$$

Write a constructor method that creates a `PolarPoint` object from a `Point` object. For this question you may need to use the `Math.sqrt(double a)` and `Math.atan2(double y, double x)` methods. [4]

iv. Write a function that calculates the shortest straight-line distance between two `PolarPoint` objects (**Hint**: This may be easier using the cartesian coordinate system). [5]