واجب اليوم الرابع

Exercises on Polymorphism:

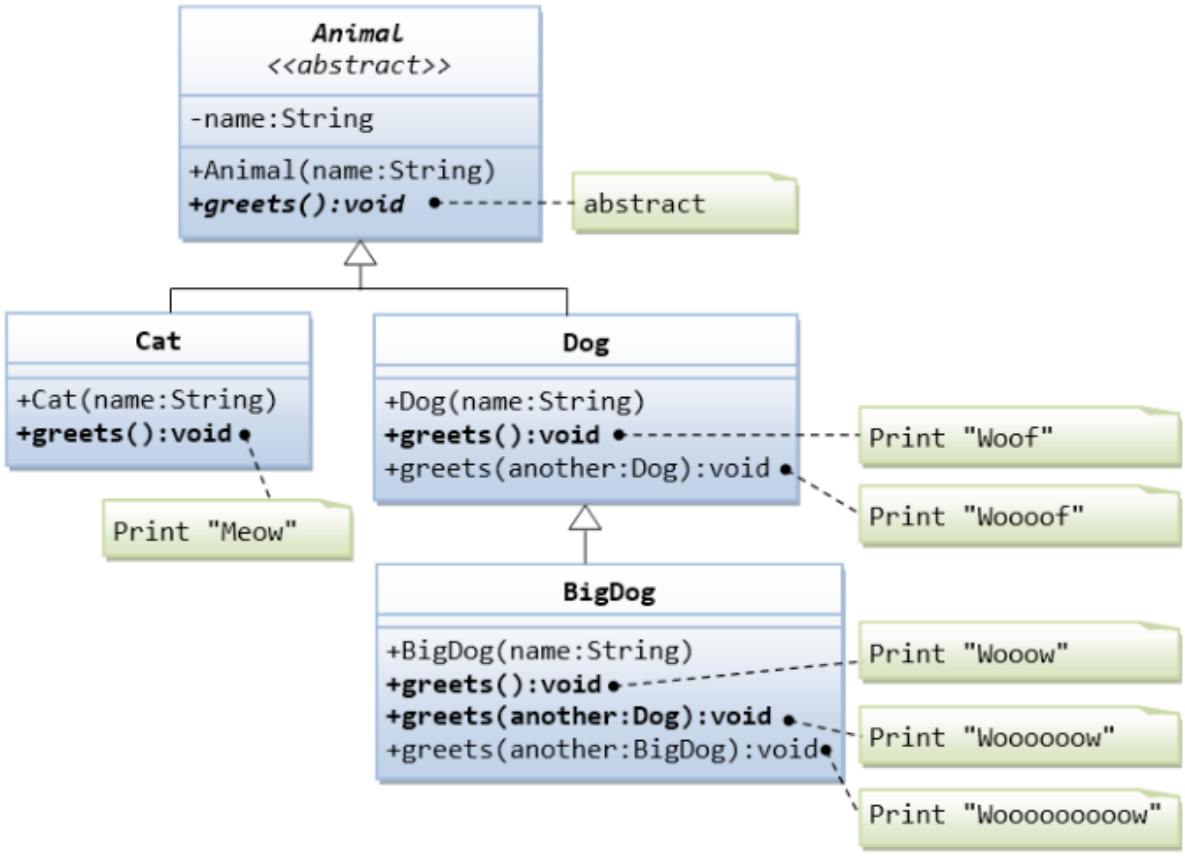
Ex: Abstract Superclass `Shape` and Its Concrete Subclasses

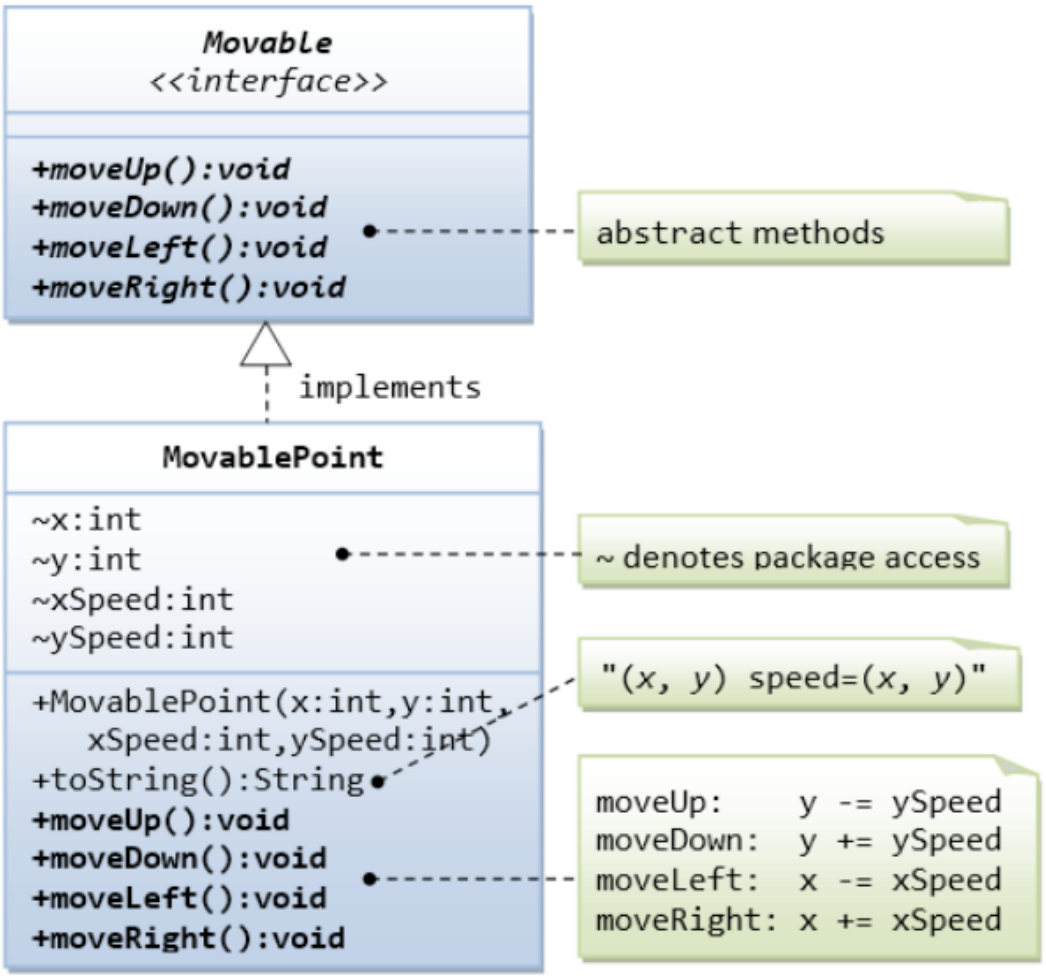
Rewrite the superclass `Shape` and its subclasses `Circle`, `Rectangle` and `Square`, as shown in the class diagram.

`Shape` is an `abstract` class containing 2 `abstract` methods: `getArea()` and `getPerimeter()`, where its concrete subclasses must provide its implementation. All instance variables shall have `protected` access, i.e., accessible by its subclasses and classes in the same package. Mark all the overridden methods with annotation `@Override`.

Ex: Abstract Superclass `Animal` and its Implementation Subclasses

Write the codes for all the classes shown in the class diagram. Mark all the overridden methods with annotation @Override.




Movable
<<interface>>
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
abstract methods
implements
MovablePoint
~x:int
~y:int
~xSpeed:int
~ySpeed:int
~ denotes package access
+MovablePoint(x:int,y:int,
xSpeed:int,ySpeed:int)
+toString():String
"(x, y) speed=(x, y)"
+moveUp():void
+moveDown():void
+moveLeft():void
+moveRight():void
moveUp: y -= ySpeed
moveDown: y += ySpeed
moveLeft: x -= xSpeed
moveRight: x += xSpeed