<<abstract>> Shape

#color:String
#filled:boolean

+Shape()

+Shape(color:String,filled:boolean)

+getColor():String

+setColor(color:String):void

+isFilled():boolean

+setFilled(filled:boolean):void

+getArea():double
+getPerimeter:double
+toString():String

Circle

#radius:double

+Circle()

+Circle(radius:double)

+Circle(radius:double,

color:String,filled:boolean)

+getRadius():double

+setRadius(radius:double):void

+getArea():double

+getPerimeter():double

+toString():String

Rectangle

#width:double
#length:double

+Rectangle()

+Rectangle(width:double,length:double)

+Rectangle(width:double,length:double,

color:String,filled:boolean)

+getWidth():double

+setWidth(width:double):void

+getLength():double

+setLength(legnth:double):void

+getArea():double

+getPerimeter():double

+toString():String

Square

+Square()

+Square(side:double)

+Square(side:double,color:String,

filled:boolean)
+getSide():double

+setSide(side:double):void

+setWidth(side:double):void

+setLength(side:double):void

+toString():String

In this exercise, Shape shall be defined as an abstract class, which contains:

- Two protected instance variables color(String) and filled(boolean).

 The protected variables can be accessed by its subclasses and classes in the same package. They are denoted with a '#' sign in the class diagram.
- Getter and setter for all the instance variables, and toString().
- Two abstract methods getArea() and getPerimeter() (shown in italics in the class diagram).

-Subclasses Circle and Rectangle shall *override* the abstract methods getArea() and getPerime ter() and provide the proper implementation. They also *override* the toString().