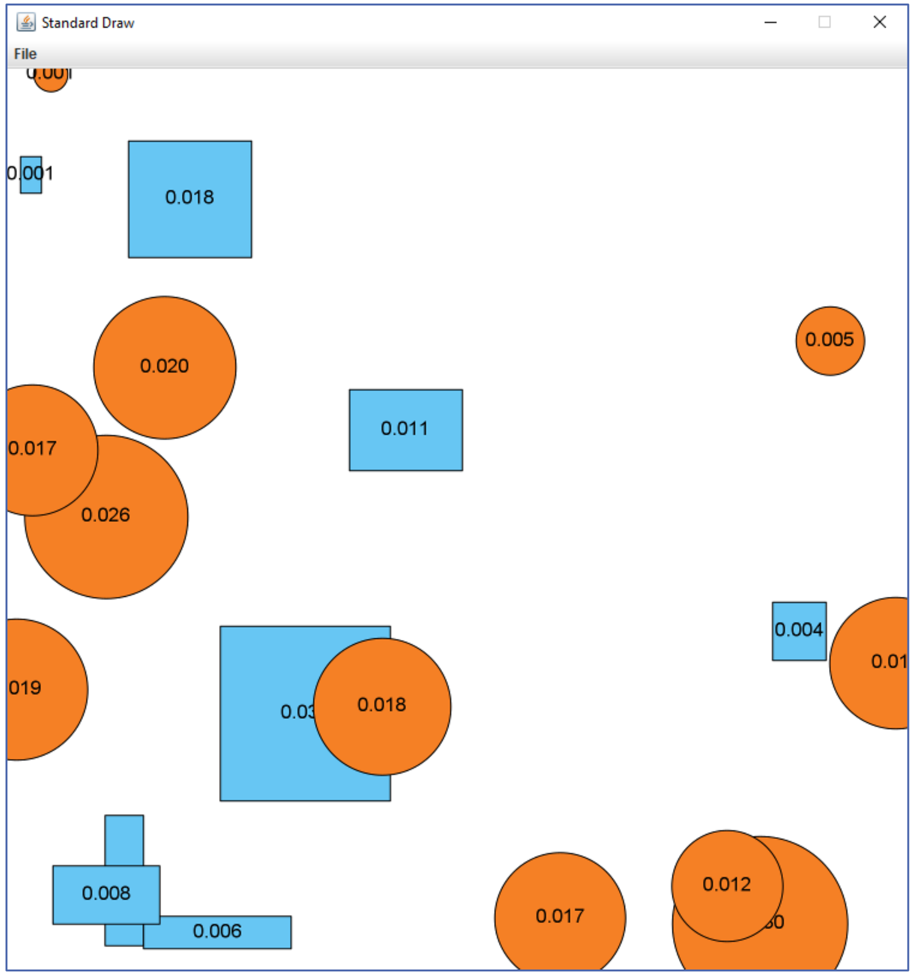
COMP 201 Data Structures and Algorithms
Lab 3 – Abstract Classes and the Comparable Interface

Base Code: Inheritance and Polymorphism with Shapes

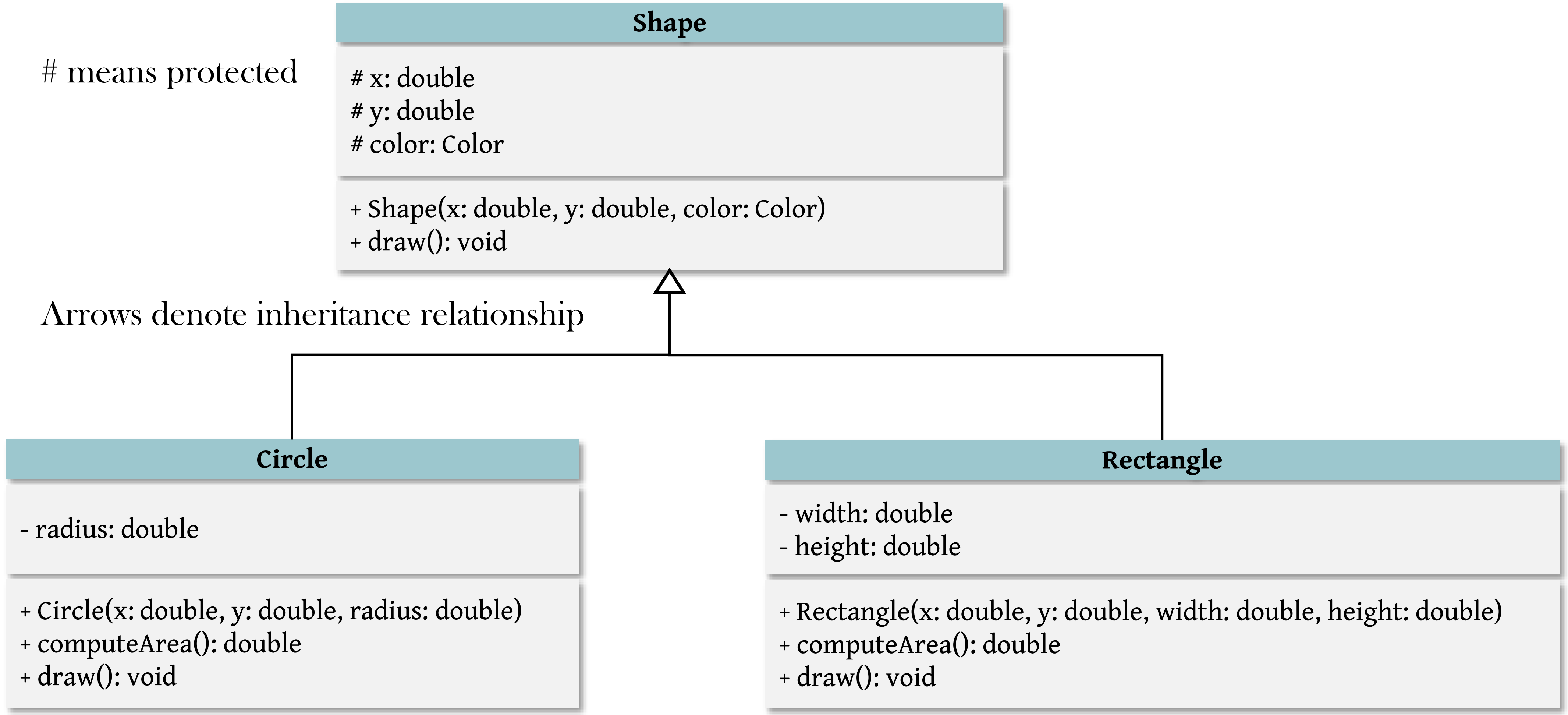
- Circle and Rectangle subclasses share some common properties, so they are derived (inherited) from a Shape superclass.
- A Shape class is written with the following common data fields:
  - x (double) and y (double): center x and y coordinates
  - color (Color type, e.g., StdDraw.ORANGE): color of the shape
  - An empty draw method that is actually implemented in subclasses.
- A Circle class is written by extending the Shape class with:
  - radius (double): radius of the circle
  - A computeArea method that computes and returns the area of the circle
  - An overridden draw method that draws the circle using StdDraw
- A Rectangle class is written by extending the Shape class with:
  - width (double) and height (double): width and height of the rectangle
  - A computeArea method that computes and returns the area of the rectangle
  - An overridden draw method that draws the rectangle using StdDraw



Base Code: UML Class Diagrams

# means protected

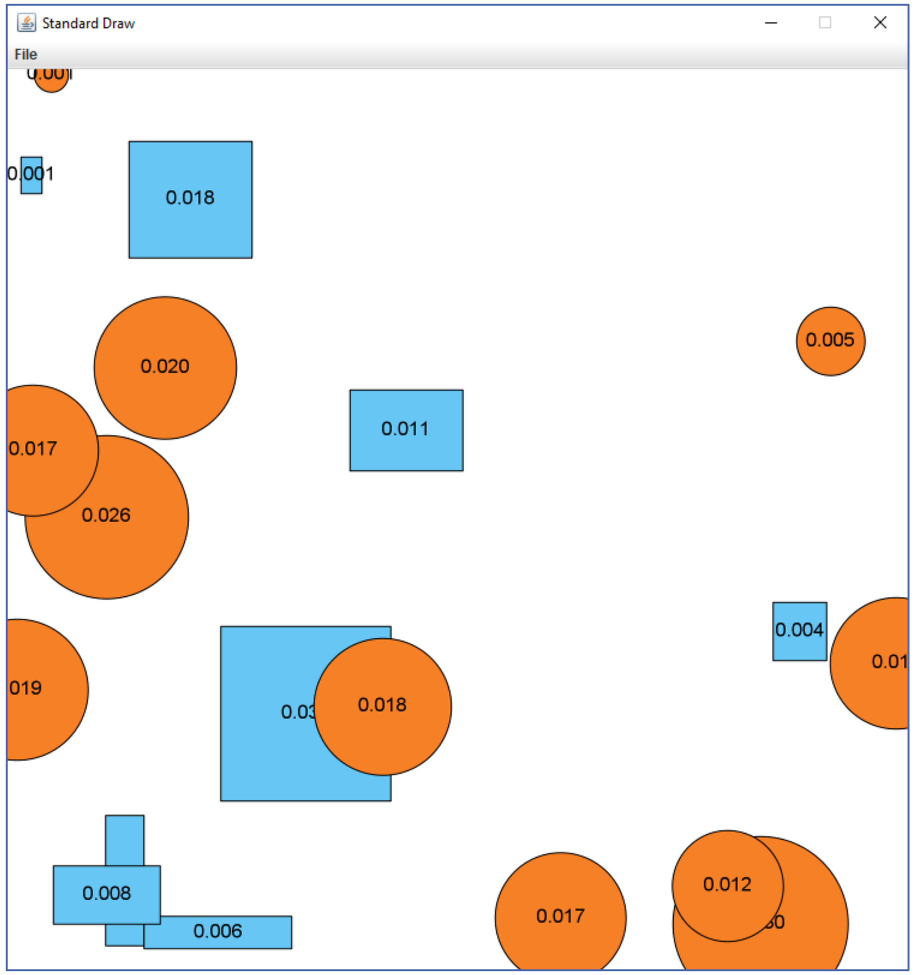
Arrows denote inheritance relationship



Drawing Random Shapes by Using **BaseCode.java**

- N random shapes are generated as follows.
  - With 0.65 probability, the shape should be a circle.
  - With 0.35 probability, the shape should be a rectangle.
  - Center x and y coordinates and sizes (radius or width&height) of the shapes are randomly generated.
- Each shape is drawn with its area written at the center with 3 decimal precision.
- Circles and rectangles are drawn in different colors.
- All created circles and rectangles are stored in a single array list.
  - ArrayList<Shape> shapes = new ArrayList<Shape>();
- The following settings are used in the program.
  - canvasWidth = canvasHeight = 750 and x-scale = y-scale = [0, 1]
  - N = 20 shapes in total and each shape has an area ≥ 0.0005 (the smallest value suitable for 3 decimal precision, 0.000 is displayed otherwise)
  - random center x and y coordinates in the range [0, 1)
  - random circle radius values in the range [0, 0.1)
  - random rectangle width and height values in the range [0, 0.2)

A Sample Drawing (60% of the Original Size):



There are more circles than rectangles.
Each shape is drawn with its area at the center.
Circles and rectangles are in different colors.

Tasks

- Modify **Shape**, **Circle** and **Rectangle** classes in **TestCode.java** as described below and shown in the UML class diagrams on the next page.
  - Task 1: Make the Shape class abstract, make the draw method in the Shape class abstract and add an abstract computeArea method to the Shape class.
  - Task 2: Implement the Comparable interface for the Shape class and add a compareTo method for comparing shapes based on their areas.
  - Task 3: Override the toString method (of the Object class) within both Circle and Rectangle classes such that you can get a similar console output with the ones given on the next pages.
  - Task 4: Implement a highlight method in the Shape class. The method should change the color of the shape to StdDraw.GREEN and then draw the shape.

UML Class Diagrams

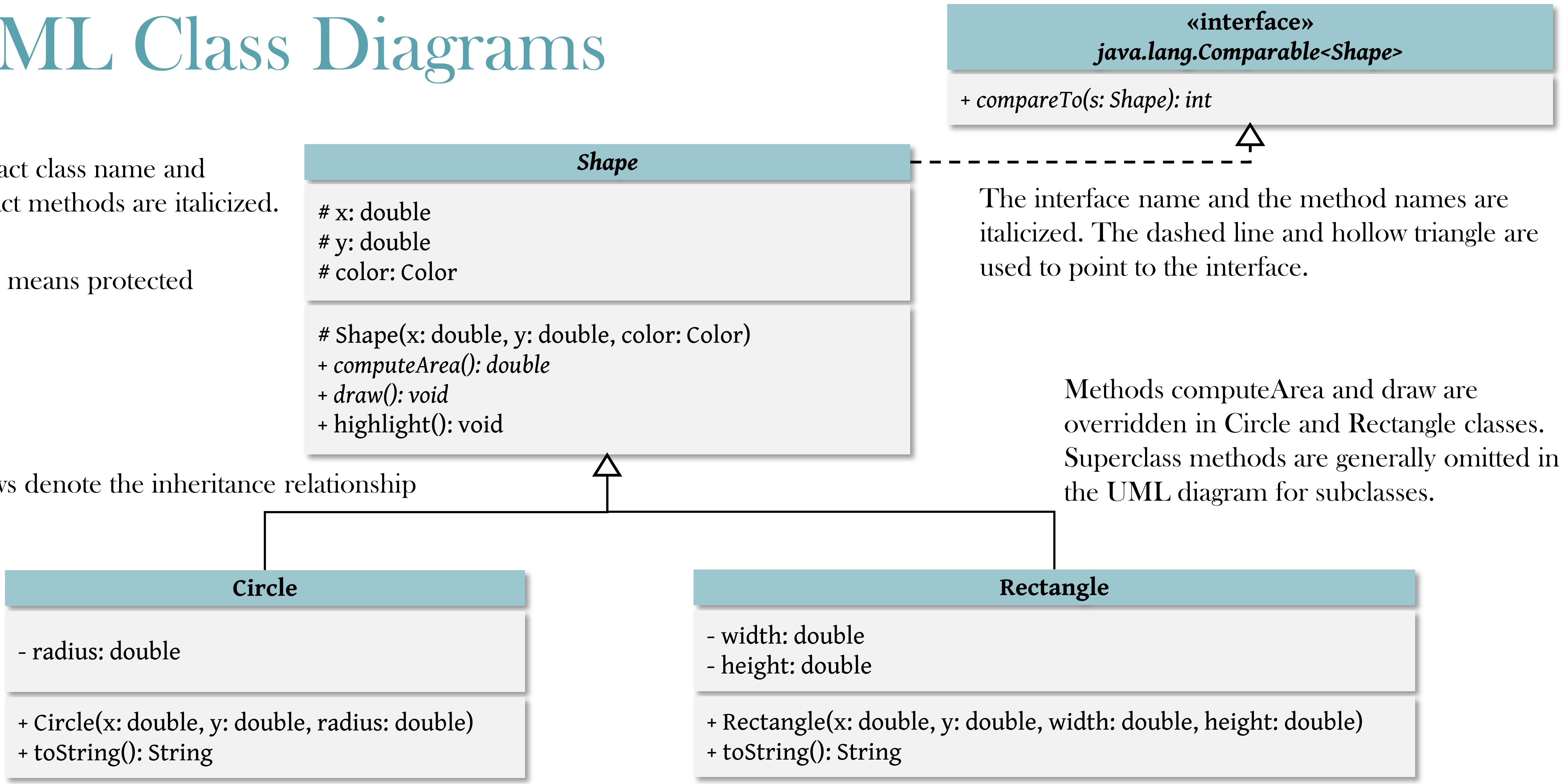
Abstract class name and abstract methods are italicized.

means protected

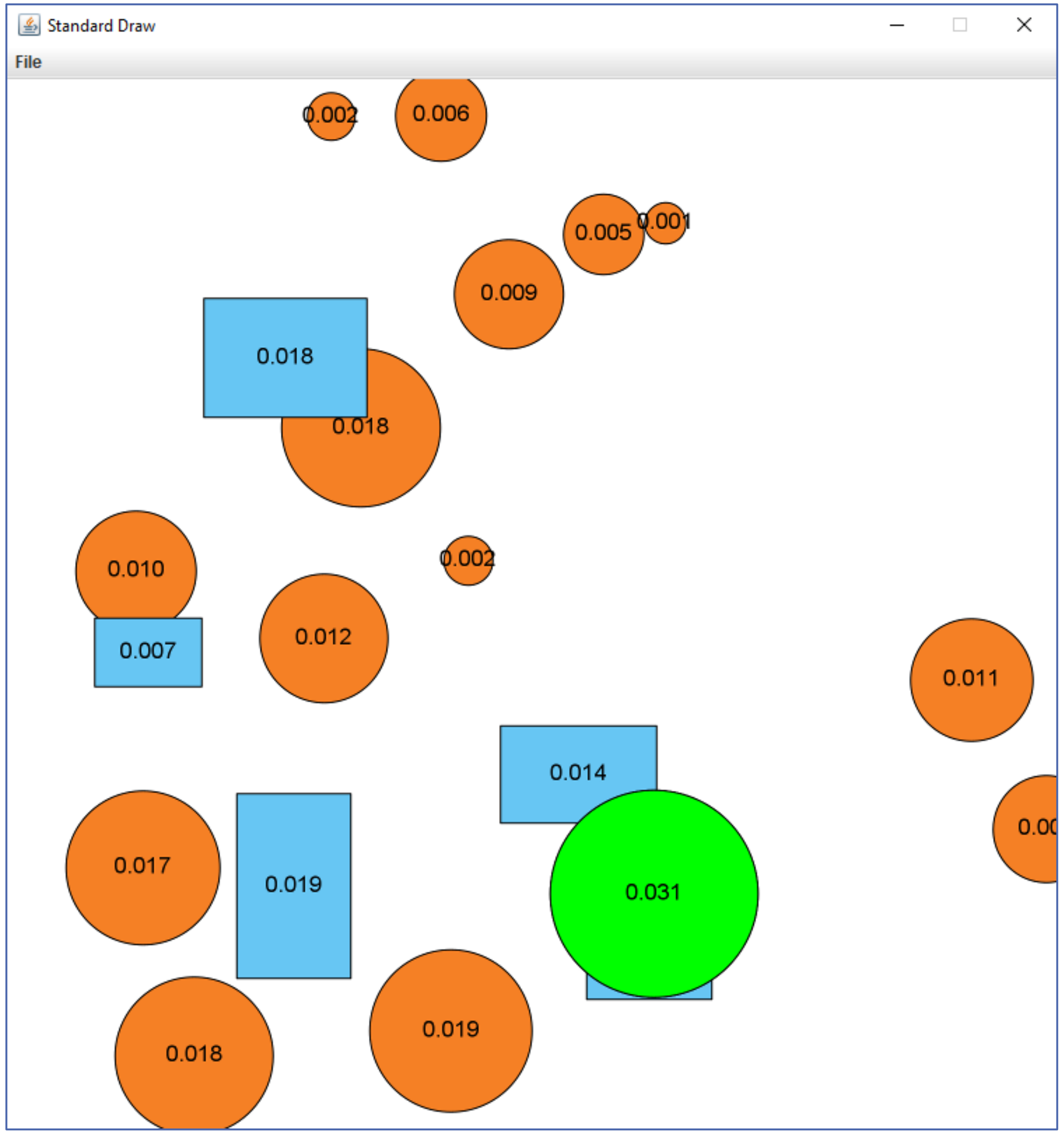
Arrows denote the inheritance relationship

The interface name and the method names are italicized. The dashed line and hollow triangle are used to point to the interface.

Methods computeArea and draw are overridden in Circle and Rectangle classes. Superclass methods are generally omitted in the UML diagram for subclasses.


«interface»
java.lang.Comparable<Shape>
+ *compareTo(s: Shape): int*
Shape
# x: double
# y: double
# color: Color
# Shape(x: double, y: double, color: Color)
+ *computeArea(): double*
+ *draw(): void*
+ highlight(): void
Circle
- radius: double
+ Circle(x: double, y: double, radius: double)
+ toString(): String
Rectangle
- width: double
- height: double
+ Rectangle(x: double, y: double, width: double, height: double)
+ toString(): String

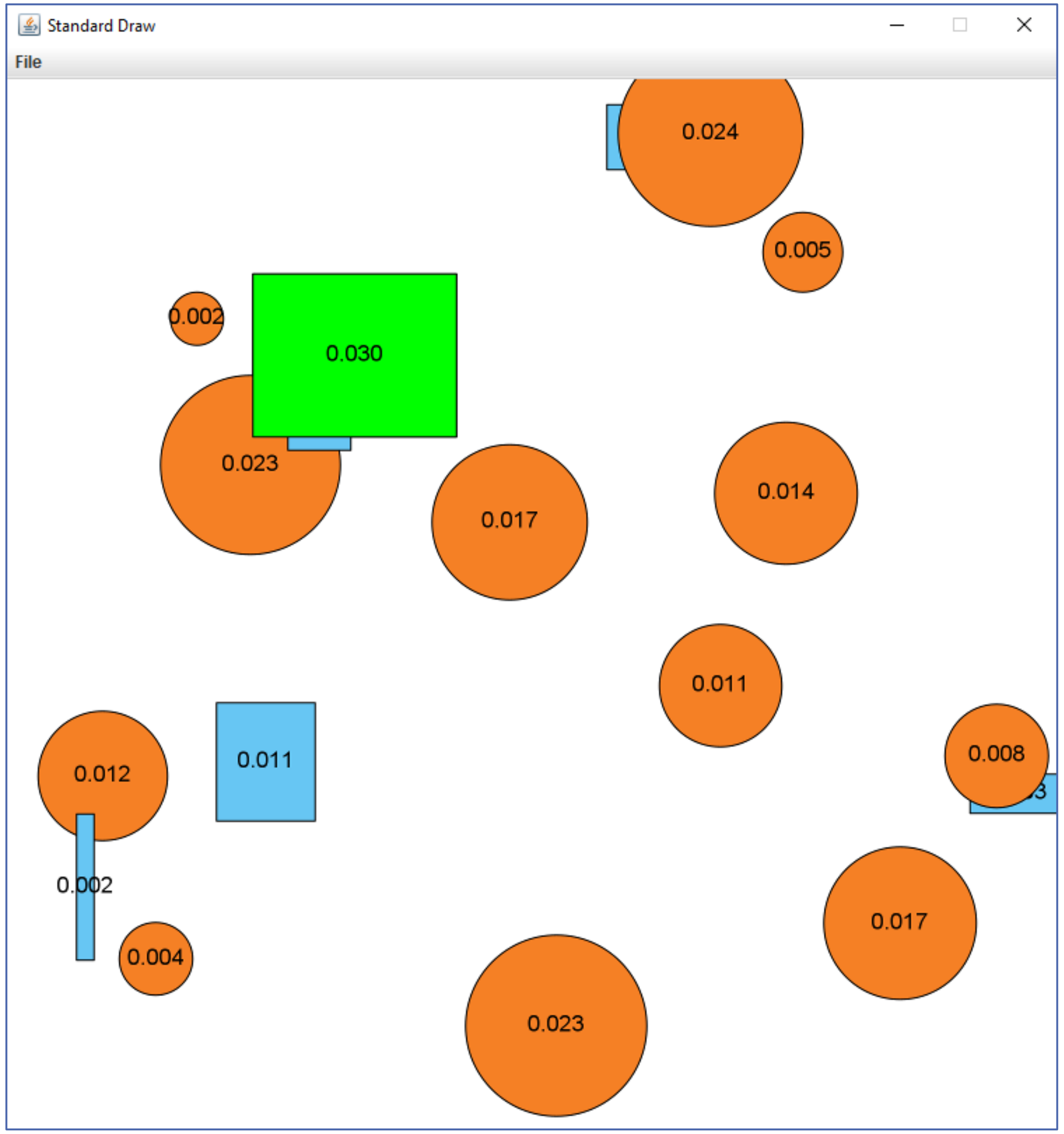
Test Code: Sample Expected Output 1



Console Output

```
Circle: 0.0011841858012981603
Circle: 0.0016500204189220818
Circle: 0.0016960760171194618
Circle: 0.004599809959461269
Circle: 0.005918208365666569
Rectangle: 0.0065344524372687O3
Rectangle: 0.0066977013163257l4
Circle: 0.008121373293027822
Circle: 0.00850006126146245
Circle: 0.010346461269087358
Circle: 0.010741803070370091
Circle: 0.011725256604163488
Rectangle: 0.013828246435163586
Circle: 0.01690134420782095
Rectangle: 0.017518031610136663
Circle: 0.017752882210025793
Circle: 0.01790847774676l2
Circle: 0.018766336744852403
Rectangle: 0.019124244154867O3
Circle: 0.030740950907656876
Program finished.
```

Test Code: Sample Expected Output 2



Console Output

```
Circle: 6.587282382318719E-4
Circle: 0.0020377057173910245
Rectangle: 0.002238231966482051
Rectangle: 0.0023583960363093813
Rectangle: 0.0034597654359662254
Circle: 0.0038008484565574805
Circle: 0.004388629739046904
Circle: 0.0045749549745403
Rectangle: 0.006645906024234093
Circle: 0.007669476711611014
Rectangle: 0.010615542595432077
Circle: 0.010673429038734868
Circle: 0.011950264581904414
Circle: 0.014479405239690862
Circle: 0.016609313538360863
Circle: 0.01728367475022856
Circle: 0.022980611786330978
Circle: 0.023424654333146427
Circle: 0.024390066407803983
Rectangle: 0.030179334215661855
Program finished.
```

Submission of Lab Work

- Submit your **Java code file(s)** (no report required) to Blackboard.
- You can work as a team of 2 students, if you like. Each team member should upload his/her code to Blackboard individually.

Grading
Your lab work will be graded on a scale of: 0: Incorrect/NA, 1: Partially correct, 2: Correct