Shape and its Subclasses

`Shape` is a public interface defined by the Java API that defines the methods required by all inheriting classes. These include `contains()`, `getBounds()`, `getBounds2D()` (a variation of `getBounds()`), `getPathIterator()`, and `intersects()`. These are the four (five including `getBounds2D()`) main methods that must be implemented by every Shape subclass. Each method is overloaded for different variations of parameters.

- `contains()` is a method to determine whether the coordinates or rectangular region defined by the parameters of the method are completely contained within the shape. The Java API documentation has details regarding how this should be defined to return true.
- `getBounds()` is a method that should return integer rectangle that completely contains the shape. `getBounds2D()` is a variation where instead of integers, the rectangle returned uses floating point numbers.
- `getPathIterator()` is a method that returns the `PathIterator` object which dictates the outline (i.e. the actual shape) of the Shape object.
- `intersects()` is a method that must tell whether the interior of the Shape intersects the interior of the rectangular region specified by the parameters.

See the Java API documentation for more information about how these methods are overloaded, and how they should be used.

Here are some of the abstract classes that implement Shape: `Arc2D`, `CubicCurve2D`, `Ellipse2D`, `Line2D`, `QuadCurve2D`, `Rectangle`/`Rectangle2D`, and `RoundRectangle2D`. These abstract classes each have `.Double` and `.Float` subclasses, which are concrete (fully implemented) classes that are constructed with doubles or floats, respectively. An example would be:

```
Ellipse2D circle = new Ellipse2D.Double(x, y, 50, 50);
```

Where `x` and `y` are variables of type double. The diameter of the circle is `50`.

In order to draw a Shape, one must use a Graphics2D object (if you have worked with graphics before, this is the "context" with which drawing shall take place). How such an object is created or obtained is outside of the scope of this summary, but an example of how one is used is not. To draw the circle we created earlier we would call the `draw()` method of our Graphics2D object (in this case named `context`):

```
context.draw(circle);
```

References:

http://download.oracle.com/javase/1.5.0/docs/api/java/awt/Shape.html

http://download.oracle.com/javase/1.5.0/docs/api/java/awt/Graphics2D.html