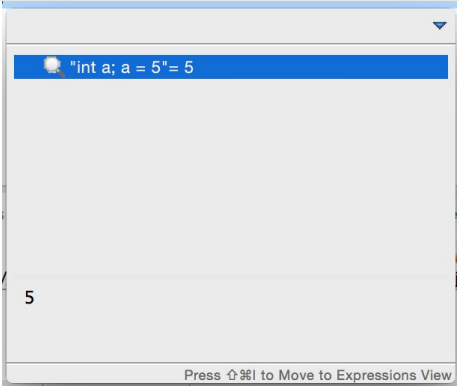

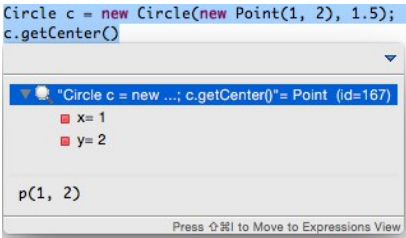
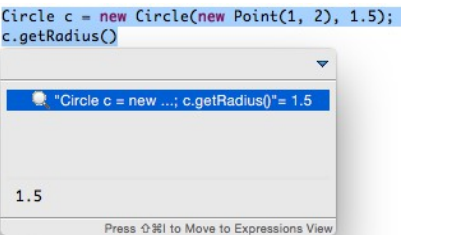
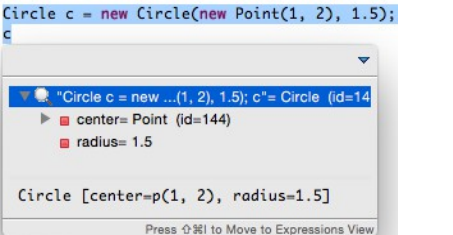
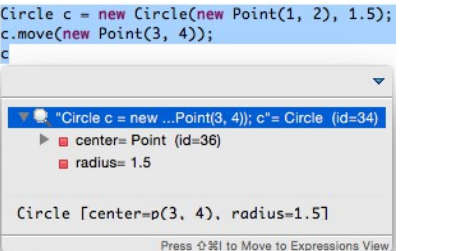
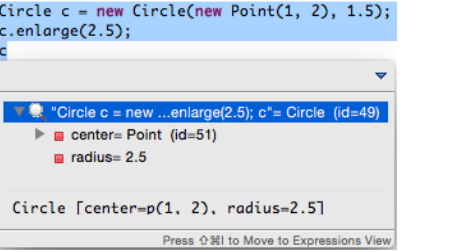


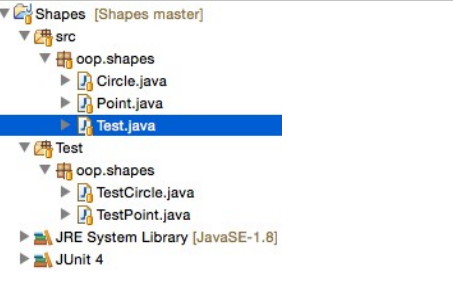
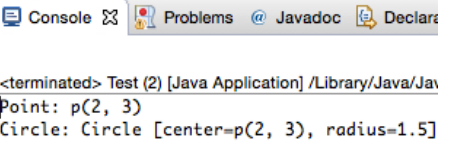
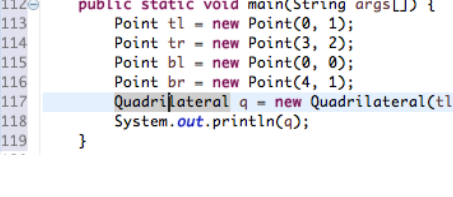
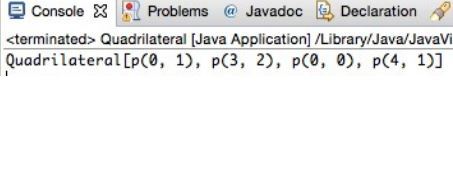
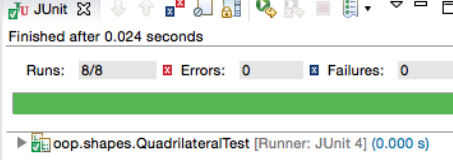
Sheet1

S.No	Prio.	Story	Test	Example	Notes
1	1	Evaluate expressions in a Scrapbook page.	<ol style="list-style-type: none"> <li>1. Declare an integer.</li> <li>2. Set variable.</li> <li>3. Declare an integer array</li> <li>4. Write a loop to initialize array</li> <li>5. Write a loop to calculate average</li> <li>6. Write a loop to revers an array.</li> <li>7. Write a lop to calculate first 10 Fibonocci numbers</li> <li>8. Write a loop that will put the first 50 integers divisible by the first five integers in five arrays.</li> </ol>	 <p>The screenshot shows a variable declaration and assignment: <code>"int a; a = 5" = 5</code>. Below the code, the value <code>5</code> is displayed. At the bottom, there is a prompt: <code>Press ⌘⇧I to Move to Expressions View</code>.</p>	Lab 0 gives instructions on setting up a Scrapbook page.
2	2	Create a Point class as described in Lecture. Member variables x and y. Methods: Point(int int), move(), getX(), getY(). ToString produces p(n, m) where n is the value of x and m is the value of y.	<ol style="list-style-type: none"> <li>1. Point p = Point(3, 5);</li> <li>2. p</li> <li>3. See p(3, 5)</li> <li>4. p.move(4, 6);</li> <li>5. p</li> <li>6. See p(4, 6)</li> <li>7. System.out.println( p.getX() + " " + p.getY());</li> <li>8 see 4 6</li> </ol>	 <p>The first screenshot shows the creation of a Point object: <code>"Point p = new Point(3, 5); p" = Point (id=108)</code>. The object's state is shown as <code>x= 3</code> and <code>y= 5</code>. Below, the expression <code>p(3, 5)</code> is evaluated. The second screenshot shows the object after a move operation: <code>"Point p = new P..., p.move(4,6); p" = Point (id=129)</code>. The state is now <code>x= 4</code> and <code>y= 6</code>. Below, the expression <code>p(4, 6)</code> is evaluated. Both screenshots include the prompt: <code>Press ⌘⇧I to Move to Expressions View</code>.</p>	
3	3	Create a Circle class that uses your Point class as a center point and include a floating point radius. Add a getCenter() method.	<ol style="list-style-type: none"> <li>1. Point p = new Point(1, 2);</li> <li>2. Circle c = new Circle(p, 1.5);</li> <li>3. c.getCenter()</li> <li>4. See p(1, 2)</li> <li>5. c.getRadius()</li> <li>6. See 1.5</li> </ol>	 <p>The screenshot shows the creation of a Circle object: <code>Circle c = new Circle(new Point(1, 2), 1.5); c.getCenter()</code>. Below, the object's state is shown as <code>"Circle c = new ...; c.getCenter()" = Point (id=167)</code> with <code>x= 1</code> and <code>y= 2</code>. Below, the expression <code>p(1, 2)</code> is evaluated. At the bottom, there is a prompt: <code>Press ⌘⇧I to Move to Expressions View</code>.</p>	

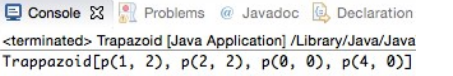
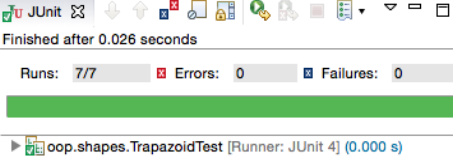
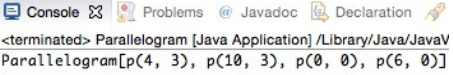
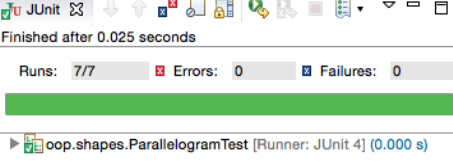
Sheet1

4	4	Add a getRadius method to your Circle Class	<ol style="list-style-type: none"> <li>1. Point p = new Point(1, 2);</li> <li>2. Circle c = new Circle(p, 1.5);</li> <li>3. c.getCenter()</li> <li>4. See p(1, 2)</li> <li>5. c.getRadius()</li> <li>6. See 1.5</li> </ol>		
5	5	Add a toString method to your Circle class that prints a c followed by parentheses followed by the center point, followed by a comma and the radius, followed by a close parenthesis.	<ol style="list-style-type: none"> <li>1. Point p = new Point(1, 2);</li> <li>2. Circle c = new Circle(p, 1.5);</li> <li>3. c.toString()</li> <li>4. See Circle [center=p(1,2), Radius=1.5]</li> </ol>		
6	6	Add a move method to your Circle class.	<ol style="list-style-type: none"> <li>1. Circle c = new Circle(new Point(1, 2), 1.5);</li> <li>2. c.move(new Point(3, 4));</li> <li>3. c</li> <li>4. See Circle [center=p(3,4), Radius=1.5]</li> </ol>		
7	7	Add an enlarge method to your Circle class	<ol style="list-style-type: none"> <li>1. Circle c = new Circle(new Point(1, 2), 1.5);</li> <li>2. c.enlarge(2.5);</li> <li>3. c</li> <li>4. See Circle [center=p(3,4), Radius=2.5]</li> </ol>		

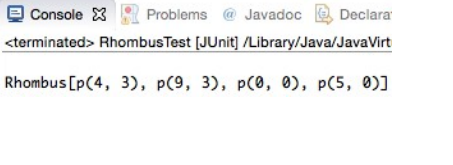
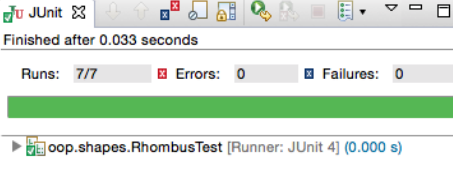
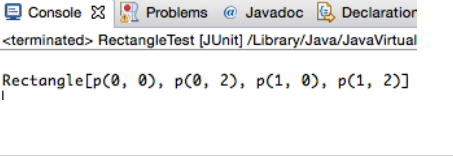
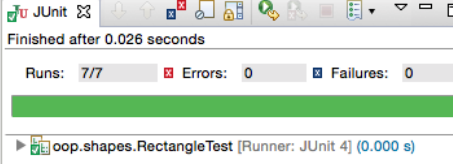
Sheet1

8	8	Save your work in a Git repository on the class AWS server.	<ol style="list-style-type: none"> <li>1. Create a local Git Repository</li> <li>2. Add your project to the local Git repository.</li> <li>3. Create a remote git repository</li> <li>4. Store your materials in the remote git repository.</li> <li>5. See the stored files</li> </ol>		
9	9	Create a new class called Test that contains only a main() method that will make a Point and Circle object, then print those objects on the consol.	<ol style="list-style-type: none"> <li>1. Select Test</li> <li>2. Choose Run as from the context menu (the one you get when you right click)</li> <li>3. Choose Java Class</li> </ol>		
10	10	Create a class, Quadrilateral, that stores four points.	<ol style="list-style-type: none"> <li>1. Create a main() method in Quadrilateral that creates a Quadrilateral.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>		
11	11	Overload toString() to print the four points of the Quadrilateral.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Quadrilateral[p(0, 1), p(3, 2), p(0, 0), p(4, 1)]</li> </ol>		
12	12	Create a test called isRightShape() that returns true when the shape is a convex quadrilateral, false otherwise.	The JUnit test, QuadrilateralTest will run correctly.		

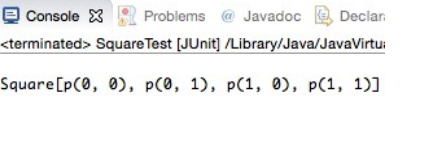
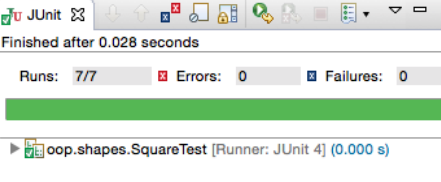
Sheet1

13	13	Create a class, Trapezoid, that is a subclass of Quadrilateral	<ol style="list-style-type: none"> <li>1. Create a main() method in Trapezoid that creates a Trapezoid.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>	<pre> 29 public static void main(String args[]) { 30     Point tl = new Point(1, 2); 31     Point tr = new Point(2, 2); 32     Point bl = new Point(0, 0); 33     Point br = new Point(4, 0); 34     Trapezoid q = new Trapezoid(tl, tr, bl, br); 35     System.out.println(q); 36 }                 </pre>
14	14	Overload toString() to print the four points of the Trapezoid.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Trappazoid[p(1, 2), p(2, 2), p(0, 0), p(4, 0)]</li> </ol>	
15	15	Create a test called isRightShape() that returns true when a shape is a trapezoid (trapezium), false otherwise.	The JUnit test, TrapazoidTest test will run correctly.	
16	16	Create a class, Parallelogram that is a subclass of Trapezoid.	<ol style="list-style-type: none"> <li>1. Create a main() method in Parallelogram that creates a Parallelogram.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>	<pre> 29 public static void main(String args[]) { 30     Point tl = new Point(4, 3); 31     Point tr = new Point(10, 3); 32     Point bl = new Point(0, 0); 33     Point br = new Point(6, 0); 34     Parallelogram q = 35     new Parallelogram(tl, tr, bl, br); 36     System.out.println(q); 37 }                 </pre>
17	17	Overload toString() to print the four points of the Parallelogram.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Parallelogram[p(4, 3), p(10, 3), p(0, 0), p(6, 0)]</li> </ol>	
18	18	Create a test called isRightShape() that returns true when a shape is a parallelogram, false otherwise.	The JUnit test, ParallelogramTest test will run correctly.	

Sheet1

19	19	Create a class, Rhombus that is a subclass of a Trapezoid.	<ol style="list-style-type: none"> <li>1. Create a main() method in Rhombus that creates a Rhombus.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>	<pre> 28 public static void main(String args[]) { 29     Point tl = new Point(4, 3); 30     Point tr = new Point(9, 3); 31     Point bl = new Point(0, 0); 32     Point br = new Point(5, 0); 33     Rhombus r = 34         new Rhombus(tl, tr, bl, br); 35     System.out.println(r); 36 }                 </pre>	
20	20	Overload toString() to print the four points of the Rhombus.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Rhombus[p(4, 3), p(9, 3), p(0, 0), p(5, 0)]</li> </ol>	 <pre> &lt;terminated&gt; RhombusTest [JUnit] /Library/Java/JavaVirt Rhombus[p(4, 3), p(9, 3), p(0, 0), p(5, 0)]                 </pre>	
21	21	Create a test called isRightShape() that returns true when a shape is a rhombus, false otherwise.	The JUnit test, RhombusTest test will run correctly.	 <pre> JUnit Finished after 0.033 seconds Runs: 7/7 Errors: 0 Failures: 0 oop.shapes.RhombusTest [Runner: JUnit 4] (0.000 s)                 </pre>	
22	22	Create a class, Rectangle that is a subclass of a Parallelogram.	<ol style="list-style-type: none"> <li>1. Create a main() method in Rectangle that creates a Rectangle.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>	<pre> 28 public static void main(String args[]) { 29     Point tl = new Point(0, 0); 30     Point tr = new Point(0, 2); 31     Point bl = new Point(1, 0); 32     Point br = new Point(1, 2); 33     Rectangle r = 34         new Rectangle(tl, tr, bl, br); 35     System.out.println(r); 36 }                 </pre>	
23	23	Overload toString() to print the four points of the Rectangle.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Rectangle[p(0, 0), p(0, 2), p(1, 0), p(1, 2)]</li> </ol>	 <pre> &lt;terminated&gt; RectangleTest [JUnit] /Library/Java/JavaVirtual Rectangle[p(0, 0), p(0, 2), p(1, 0), p(1, 2)]                 </pre>	
24	24	Create a test called isRightShape() that returns true when a shape is a rectangle, false otherwise.	The JUnit test, RectangleTest test will run correctly.	 <pre> JUnit Finished after 0.026 seconds Runs: 7/7 Errors: 0 Failures: 0 oop.shapes.RectangleTest [Runner: JUnit 4] (0.000 s)                 </pre>	

Sheet1

25	25	Create a class, Square that is a subclass of Rhombus.	<ol style="list-style-type: none"> <li>1. Create a main() method in Square that creates a Square.</li> <li>2. Pass the constructor four points.</li> <li>3. See the method compile and run.</li> </ol>	<pre> 29 public static void main(String args[]) { 30     Point tl = new Point(0, 0); 31     Point tr = new Point(0, 1); 32     Point bl = new Point(1, 0); 33     Point br = new Point(1, 1); 34     Square r = 35         new Square(tl, tr, bl, br); 36     System.out.println(r); 37 }                 </pre>	
26	26	Overload toString() to print the four points.	<ol style="list-style-type: none"> <li>1. in the main() method call the toString() method.</li> <li>2. See Square[p(0, 0), p(0, 1), p(1, 0), p(1, 1)]</li> </ol>		
27	27	Create a test called isRightShape() that returns true when a shape is a Square, false otherwise.	The JUnit test, SquareTest will run correctly.		
28	28				