1. It was my software development / programming experience with Java so I had to not only Java from scratch but also learn testing and UML diagrams. I realized that testing is given prime place in Java judging from the availability and support of testing frameworks.

2. Inheritance helped to reduce code thus reducing chances and points of error and the amount of tests that need to be written. It also allows easy maintainability.

3. There was a conundrum deciding the order of inheritance for square and rectangle as well as for ellipse and circle.

4. There were errors in test code themselves while the working logic of the classes being perfect. This implies even when classes are well built their tests code can have faults which are easily discovered because test and main source code must agree anyway.

5. Makes making change to code base easy after the tests are written because you would know if your changes broke the system or not depending on the output of tests. That's how I removed moveX() and moveY() functions in favor of single move() function.

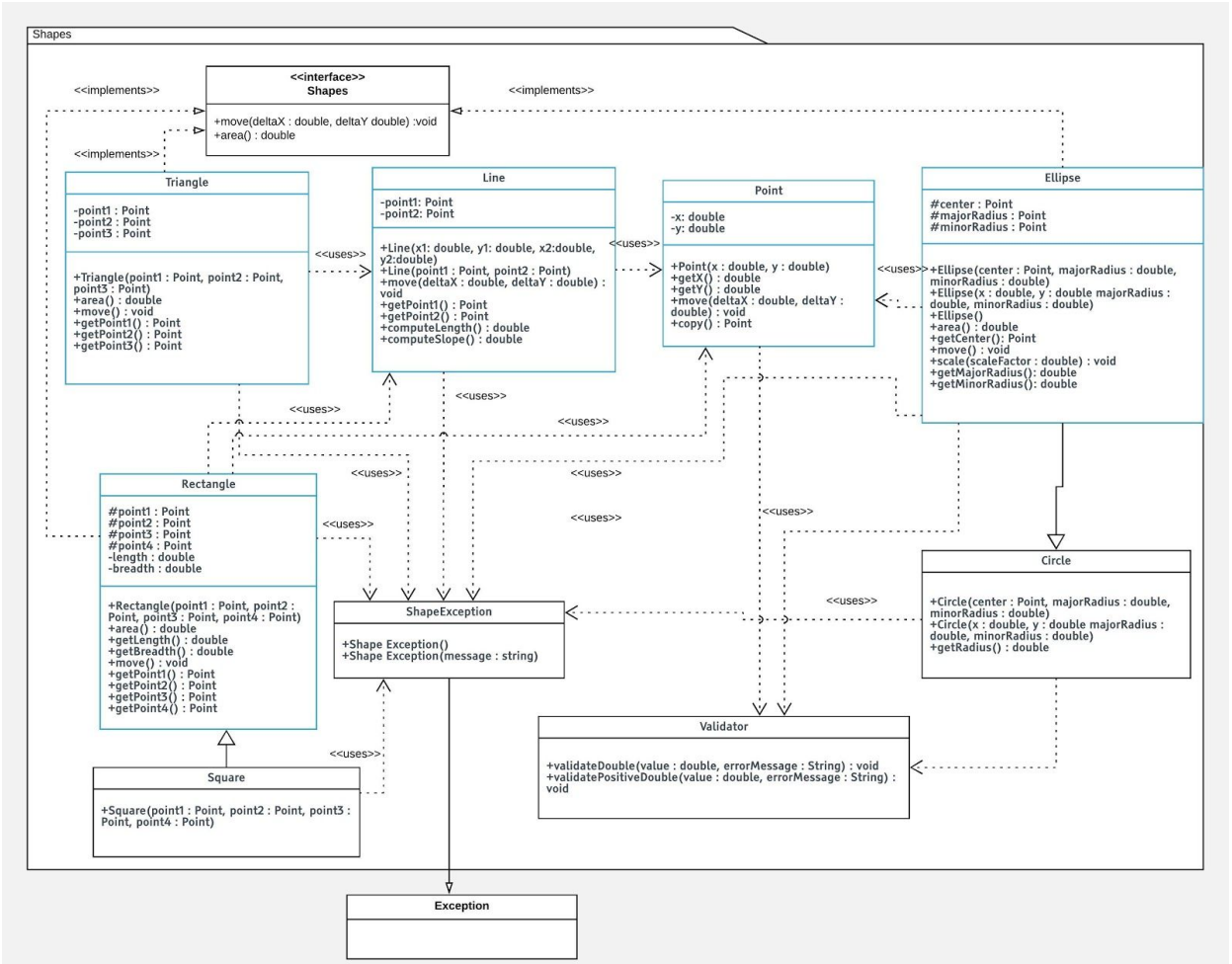
6. Even with all the tests and hundred percent accuracy there could still be glitches or technical errors in your code because you did not cover some test cases. Writing tests and choosing test examples is a rigorous task.

7. I tested multiple formulas to compute area of triangle. They offered trade-off in space, time and accuracy. I choose to go with accuracy and space. Again since the tests cases had been implemented it was easy to test different formulas for the area computation. Ultimately, this package is going to offer different performance to different types of user depending on their usage of the package and its various features. There will always be trade offs involved.

8. I also discovered that floating point calculations can be lossy while computing the area of triangle. Using multiple steps to compute area can be lossy, so I decided to compute area using coordinate of vertices using single statement.

9. It's amazing how we can use asserts to verify that our code works without using a single print statements. It's also wonderful that interfaces are standardizations enforced by compiler.

10. For sometime I struggled with changing the exception that is thrown for construction of child class. I thought I should just pass a general error in the parent class and be done with it. But I decided to make error messages more helpful and specific and found way to override parent class exception messages.


Shapes
<<interface>>
Shapes
+move(deltaX : double, deltaY double) :void
+area() : double
<<implements>>
<<implements>>
<<implements>>
Triangle
-point1 : Point
-point2 : Point
-point3 : Point
+Triangle(point1 : Point, point2 : Point, point3 : Point)
+area() : double
+move() : void
+getPoint1() : Point
+getPoint2() : Point
+getPoint3() : Point
Line
-point1: Point
-point2: Point
+Line(x1: double, y1: double, x2:double, y2:double)
+Line(point1 : Point, point2 : Point)
+move(deltaX : double, deltaY : double) : void
+getPoint1() : Point
+getPoint2() : Point
+computeLength() : double
+computeSlope() : double
Point
-x: double
-y: double
+Point(x : double, y : double)
+getX() : double
+getY() : double
+move(deltaX : double, deltaY : double) : void
+copy() : Point
Ellipse
#center : Point
#majorRadius : Point
#minorRadius : Point
+Ellipse(center : Point, majorRadius : double, minorRadius : double)
+Ellipse(x : double, y : double majorRadius : double, minorRadius : double)
+Ellipse()
+area() : double
+getCenter(): Point
+move() : void
+scale(scaleFactor : double) : void
+getMajorRadius(): double
+getMinorRadius(): double
<<uses>>
Rectangle
#point1 : Point
#point2 : Point
#point3 : Point
#point4 : Point
-length : double
-breadth : double
+Rectangle(point1 : Point, point2 : Point, point3 : Point, point4 : Point)
+area() : double
+getLength() : double
+getBreadth() : double
+move() : void
+getPoint1() : Point
+getPoint2() : Point
+getPoint3() : Point
+getPoint4() : Point
Circle
+Circle(center : Point, majorRadius : double, minorRadius : double)
+Circle(x : double, y : double majorRadius : double, minorRadius : double)
+getRadius() : double
ShapeException
+Shape Exception()
+Shape Exception(message : string)
Validator
+validateDouble(value : double, errorMessage : String) : void
+validatePositiveDouble(value : double, errorMessage : String) : void
Square
+Square(point1 : Point, point2 : Point, point3 : Point, point4 : Point)
Exception