```
00001: /*
00002:  * To change this license header, choose License Headers in Project Properties.
00003:  * To change this template file, choose Tools | Templates
00004:  * and open the template in the editor.
00005:  */
00006: 
00007: import java.awt.Point;
00008: import org.junit.Test;
00009: import static org.junit.Assert.*;
00010: 
00011: /**
00012:  * This is the Java JUnit test for Quadrilateral.java
00013:  *
00014:  * @author WahabEhsan
00015:  */
00016: public class QuadrilateralTest {
00017: 
00018:     private Point p1;
00019:     private Point p2;
00020:     private Point p3;
00021:     private Point p4;
00022: 
00023:     /**
00024:      * Initialize the four points with a square
00025:      */
00026:     private void initPointsWithSquare() {
00027:         //Square
00028:         p1 = new Point(1, 1);
00029:         p2 = new Point(1, 2);
00030:         p3 = new Point(2, 2);
00031:         p4 = new Point(2, 1);
00032:     }
00033: 
```

```
00034:     /**
00035:      * Test isParallelogram with a square as input
00036:      */
00037:     @Test
00038:     public void testIsParallelogramWithSquare() {
00039:         initPointsWithSquare();
00040:
00041:         assertEquals(true, Quadrilateral.isParallelogram(p1, p2, p3, p4));
00042:     }
00043:
00044:     /**
00045:      * Test isRhombus with a square as input
00046:      */
00047:     @Test
00048:     public void testIsRhombusWithSquare() {
00049:         initPointsWithSquare();
00050:         assertEquals(true, Quadrilateral.isRhombus(p1, p2, p3, p4));
00051:     }
00052:
00053:     /**
00054:      * Test isKite with a square as input
00055:      */
00056:     @Test
00057:     public void testIsKiteWithSquare() {
00058:         initPointsWithSquare();
00059:         assertEquals(true, Quadrilateral.isKite(p1, p2, p3, p4));
00060:     }
00061:
00062:     /**
00063:      * Test isRectangle with a square as input
00064:      */
00065:     @Test
00066:     public void testIsRectangleWithSquare() {
```

```
00067:         initPointsWithSquare();
00068:         assertEquals(true, Quadrilateral.isRectangle(p1, p2, p3, p4));
00069:     }
00070:
00071:     /**
00072:      * Test isSquare with a square as input
00073:      */
00074:     @Test
00075:     public void testIsSquareWithSquare() {
00076:         initPointsWithSquare();
00077:         assertEquals(true, Quadrilateral.isSquare(p1, p2, p3, p4));
00078:     }
00079:
00080:     //Rectangle - Add 5 testcases, one for each of the methods as shown above, for a quadrilateral
00081:     //            that is a Rectangle, but not a Square.
00082:     /**
00083:      * Test isRectanlge with parallelogram points
00084:      */
00085:     @Test
00086:     public void testIsRectangleWithParallelogram() {
00087:         Point p1 = new Point(2, 0);
00088:         Point p2 = new Point(2, 4);
00089:         Point p3 = new Point(6, 6);
00090:         Point p4 = new Point(6, 2);
00091:         assertEquals(false, Quadrilateral.isRectangle(p1, p2, p3, p4));
00092:     }
00093:
00094:     /**
00095:      * Test isRectanlge with rhombus points
00096:      */
00097:     @Test
00098:     public void testIsRectangleWithRhombus() {
00099:         Point p1 = new Point(1, 2);
```

```
00100:         Point p2 = new Point(2, 5);
00101:         Point p3 = new Point(5, 6);
00102:         Point p4 = new Point(4, 3);
00103:         assertEquals(false, Quadrilateral.isRectangle(p1, p2, p3, p4));
00104:     }
00105:
00106:     /**
00107:      * Test isRectanlge with kite points
00108:      */
00109:     @Test
00110:     public void testIsRectangleWithKite() {
00111:         Point p1 = new Point(0, 1);
00112:         Point p2 = new Point(1, 2);
00113:         Point p3 = new Point(3, 1);
00114:         Point p4 = new Point(1, 0);
00115:         assertEquals(false, Quadrilateral.isRectangle(p1, p2, p3, p4));
00116:     }
00117:
00118:     /**
00119:      * Test isRectanlge with rectangle points
00120:      */
00121:     @Test
00122:     public void testIsRectangleWithRectangle() {
00123:         Point p1 = new Point(2, 2);
00124:         Point p2 = new Point(2, 4);
00125:         Point p3 = new Point(5, 4);
00126:         Point p4 = new Point(5, 2);
00127:         assertEquals(true, Quadrilateral.isRectangle(p1, p2, p3, p4));
00128:     }
00129:
00130:     /**
00131:      * Test isRectanlge with other points
00132:      */
```

```
00133:     @Test
00134:     public void testIsRectangleWithOther() {
00135:         Point p1 = new Point(0, 1);
00136:         Point p2 = new Point(1, 2);
00137:         Point p3 = new Point(3, 1);
00138:         Point p4 = new Point(5, 3);
00139:         assertEquals(false, Quadrilateral.isRectangle(p1, p2, p3, p4));
00140:     }
00141:
00142:     //Parallelogram - Add 5 testcases, one for each of the methods as shown above, for a quadrilateral
00143:     //                that is a Parallelogram, but not a Rectangle.
00144:     /**
00145:      * Test isParallelogram with a other as input
00146:      */
00147:     @Test
00148:     public void testIsParallelogramWithOther() {
00149:         Point p1 = new Point(0, 1);
00150:         Point p2 = new Point(1, 2);
00151:         Point p3 = new Point(3, 1);
00152:         Point p4 = new Point(5, 3);
00153:         assertEquals(false, Quadrilateral.isParallelogram(p1, p2, p3, p4));
00154:     }
00155:
00156:     /**
00157:      * Test isParallelogram with a Square as input
00158:      */
00159:     @Test
00160:     public void testIsParallelogramSquare() {
00161:         Point p1 = new Point(0, 0);
00162:         Point p2 = new Point(0, 1);
00163:         Point p3 = new Point(1, 1);
00164:         Point p4 = new Point(1, 0);
00165:         assertEquals(true, Quadrilateral.isParallelogram(p1, p2, p3, p4));
```

```
00166:     }
00167:
00168:     /**
00169:      * Test isParallelogram with a Kite as input
00170:      */
00171:     @Test
00172:     public void testIsParallelogramWithKite() {
00173:         Point p1 = new Point(0, 1);
00174:         Point p2 = new Point(1, 2);
00175:         Point p3 = new Point(3, 1);
00176:         Point p4 = new Point(1, 0);
00177:         assertEquals(false, Quadrilateral.isParallelogram(p1, p2, p3, p4));
00178:     }
00179:
00180:     /**
00181:      * Test isParallelogram with a Rhombus as input
00182:      */
00183:     @Test
00184:     public void testIsParallelogramRhombus() {
00185:         Point p1 = new Point(1, 2);
00186:         Point p2 = new Point(2, 5);
00187:         Point p3 = new Point(5, 6);
00188:         Point p4 = new Point(4, 3);
00189:         assertEquals(true, Quadrilateral.isParallelogram(p1, p2, p3, p4));
00190:     }
00191:
00192:     /**
00193:      * Test isParallelogram with a Rectangle as input
00194:      */
00195:     @Test
00196:     public void testIsParallelogramRectangle() {
00197:         Point p1 = new Point(2, 2);
00198:         Point p2 = new Point(2, 4);
```

```
00199:         Point p3 = new Point(5, 4);
00200:         Point p4 = new Point(5, 2);
00201:         assertEquals(true, Quadrilateral.isParallelogram(p1, p2, p3, p4));
00202:     }
00203:
00204:     //Rhombus - Add 5 testcases, one for each of the methods as shown above, for a quadrilateral
00205:     //          that is a Rhombus, but not a Square.
00206:     /**
00207:      * This isRhombus test with Rhombus input
00208:      */
00209:     @Test
00210:     public void testIsRhombusWithRhombus() {
00211:         Point p1 = new Point(1, 2);
00212:         Point p2 = new Point(2, 5);
00213:         Point p3 = new Point(5, 6);
00214:         Point p4 = new Point(4, 3);
00215:         assertEquals(true, Quadrilateral.isRhombus(p1, p2, p3, p4));
00216:     }
00217:
00218:     /**
00219:      * This isRhombus test with kite input
00220:      */
00221:     @Test
00222:     public void testIsRhombusWithKite() {
00223:         Point p1 = new Point(0, 1);
00224:         Point p2 = new Point(1, 2);
00225:         Point p3 = new Point(3, 1);
00226:         Point p4 = new Point(1, 0);
00227:         assertEquals(false, Quadrilateral.isRhombus(p1, p2, p3, p4));
00228:     }
00229:
00230:     /**
00231:      * This isRhombus test with parallelogram input
```

```
00232:      */
00233:     @Test
00234:     public void testIsRhombusWithParallelogram() {
00235:         Point p1 = new Point(2, 0);
00236:         Point p2 = new Point(2, 4);
00237:         Point p3 = new Point(6, 6);
00238:         Point p4 = new Point(6, 2);
00239:         assertEquals(false, Quadrilateral.isRhombus(p1, p2, p3, p4));
00240:     }
00241:
00242:     /**
00243:      * This isRhombus test with rectangle input
00244:      */
00245:     @Test
00246:     public void testIsRhombusWithRectanlge() {
00247:         Point p1 = new Point(2, 2);
00248:         Point p2 = new Point(2, 4);
00249:         Point p3 = new Point(5, 4);
00250:         Point p4 = new Point(5, 2);
00251:         assertEquals(false, Quadrilateral.isRhombus(p1, p2, p3, p4));
00252:     }
00253:
00254:     /**
00255:      * This isRhombus test with other input
00256:      */
00257:     @Test
00258:     public void testIsRhombusWithOther() {
00259:         Point p1 = new Point(0, 1);
00260:         Point p2 = new Point(1, 2);
00261:         Point p3 = new Point(3, 1);
00262:         Point p4 = new Point(5, 3);
00263:         assertEquals(false, Quadrilateral.isRhombus(p1, p2, p3, p4));
00264:     }
```

```
00265:
00266:     //Kite - Add 5 testcases, one for each of the methods as shown above, for a quadrilateral
00267:     //       that is a Kite, but not a Square.
00268:     /**
00269:      * Test of isKite with kite points
00270:      */
00271:     @Test
00272:     public void testIsKiteWithKite() {
00273:         Point p1 = new Point(0, 1);
00274:         Point p2 = new Point(1, 2);
00275:         Point p3 = new Point(3, 1);
00276:         Point p4 = new Point(1, 0);
00277:         assertEquals(true, Quadrilateral.isKite(p1, p2, p3, p4));
00278:     }
00279:
00280:     /**
00281:      * Test of isKite with Rhombus points
00282:      */
00283:     @Test
00284:     public void testIsKiteWithRhombus() {
00285:         Point p1 = new Point(1, 2);
00286:         Point p2 = new Point(2, 5);
00287:         Point p3 = new Point(5, 6);
00288:         Point p4 = new Point(4, 3);
00289:         assertEquals(true, Quadrilateral.isKite(p1, p2, p3, p4));
00290:     }
00291:
00292:     /**
00293:      * Test of isKite with rectangle points
00294:      */
00295:     @Test
00296:     public void testIsKiteWithRectangle() {
00297:         Point p1 = new Point(2, 2);
```

```
00298:         Point p2 = new Point(2, 4);
00299:         Point p3 = new Point(5, 4);
00300:         Point p4 = new Point(5, 2);
00301:         assertEquals(false, Quadrilateral.isKite(p1, p2, p3, p4));
00302:     }
00303:
00304:     /**
00305:      * Test of isKite with parallelogram points
00306:      */
00307:     @Test
00308:     public void testIsKiteWithParallelogram() {
00309:         Point p1 = new Point(2, 0);
00310:         Point p2 = new Point(2, 4);
00311:         Point p3 = new Point(6, 6);
00312:         Point p4 = new Point(6, 2);
00313:         assertEquals(false, Quadrilateral.isKite(p1, p2, p3, p4));
00314:     }
00315:
00316:     /**
00317:      * Test of isKite with other points
00318:      */
00319:     @Test
00320:     public void testIsKiteWithOther() {
00321:         Point p1 = new Point(0, 1);
00322:         Point p2 = new Point(1, 2);
00323:         Point p3 = new Point(3, 1);
00324:         Point p4 = new Point(5, 3);
00325:         assertEquals(false, Quadrilateral.isKite(p1, p2, p3, p4));
00326:     }
00327:
00328:     //Other - Add 5 testcases, one for each of the methods as shown above, for a quadrilateral
00329:     //        that is not a Kite nor a Parallelogram.
00330:     /**
```

```
00331:      * This test the isOther with square
00332:      */
00333:     @Test
00334:     public void testIsOtherWithSquare() {
00335:         Point p1 = new Point(0, 0);
00336:         Point p2 = new Point(0, 1);
00337:         Point p3 = new Point(1, 1);
00338:         Point p4 = new Point(1, 0);
00339:         assertEquals(true, Quadrilateral.isKite(p1, p2, p3, p4));
00340:     }
00341:
00342:     /**
00343:      * This test the isOther with rectangle
00344:      */
00345:     @Test
00346:     public void testIsOtherWithRectangle() {
00347:         Point p1 = new Point(2, 2);
00348:         Point p2 = new Point(2, 4);
00349:         Point p3 = new Point(5, 4);
00350:         Point p4 = new Point(5, 2);
00351:         assertEquals(false, Quadrilateral.isKite(p1, p2, p3, p4));
00352:     }
00353:
00354:     /**
00355:      * This test the isOther with kite
00356:      */
00357:     @Test
00358:     public void testIsOtherWithKite() {
00359:         Point p1 = new Point(0, 1);
00360:         Point p2 = new Point(1, 2);
00361:         Point p3 = new Point(3, 1);
00362:         Point p4 = new Point(1, 0);
00363:         assertEquals(false, Quadrilateral.isSquare(p1, p2, p3, p4));
```

```
00364:     }
00365:
00366:     /**
00367:      * This test the isOther with parallelogram
00368:      */
00369:     @Test
00370:     public void testIsOtherWithParallelogram() {
00371:         Point p1 = new Point(2, 0);
00372:         Point p2 = new Point(2, 4);
00373:         Point p3 = new Point(6, 6);
00374:         Point p4 = new Point(6, 2);
00375:         assertEquals(false, Quadrilateral.isKite(p1, p2, p3, p4));
00376:     }
00377:
00378:     /**
00379:      * This test the isOther with rhombus
00380:      */
00381:     @Test
00382:     public void testIsOtherWithRhombus() {
00383:         Point p1 = new Point(1, 2);
00384:         Point p2 = new Point(2, 5);
00385:         Point p3 = new Point(5, 6);
00386:         Point p4 = new Point(4, 3);
00387:         assertEquals(true, Quadrilateral.isKite(p1, p2, p3, p4));
00388:     }
00389:
00390:     /**
00391:      * Test of isRightAngle method, with three points that form a right angle.
00392:      */
00393:     @Test
00394:     public void testIsRightAngleWithValidRightAngle() {
00395:         Point p1 = new Point(1, 1);
00396:         Point p2 = new Point(1, 2);
```

```
00397:         Point p3 = new Point(2, 2);
00398:         assertEquals(true, Quadrilateral.isRightAngle(p1, p2, p3));
00399:     }
00400:
00401:     /**
00402:      * Test of isRightAngle method, with three points that don't form a right
00403:      * angle.
00404:      */
00405:     @Test
00406:     public void testIsRightAngleWithInVvlidRightAngle() {
00407:         Point p1 = new Point(0, 1);
00408:         Point p2 = new Point(1, 2);
00409:         Point p3 = new Point(2, 5);
00410:         assertEquals(false, Quadrilateral.isRightAngle(p1, p2, p3));
00411:     }
00412: }
```