Report 1

1. Reflection

In this assessment, I learned to establish the Java class, create methods for the class, overridden for the existing methods and create unit test to test the correctness for the methods and increase the covering rates.

In terms of test files, it is quite clear to make comparation with expected output to the actual output in the separate test file, the syntax is simple and easy to write. The tests will help us detect errors in an early stage.

2. Testing Comparison

What's the difference between testing with JUnit tests and with a Driver? Why did I ask you to create a driver this time?

JUnit tests have a very clear and simple syntax and formatted structure, gives the tester a direct view if the output is identical to the expected result.

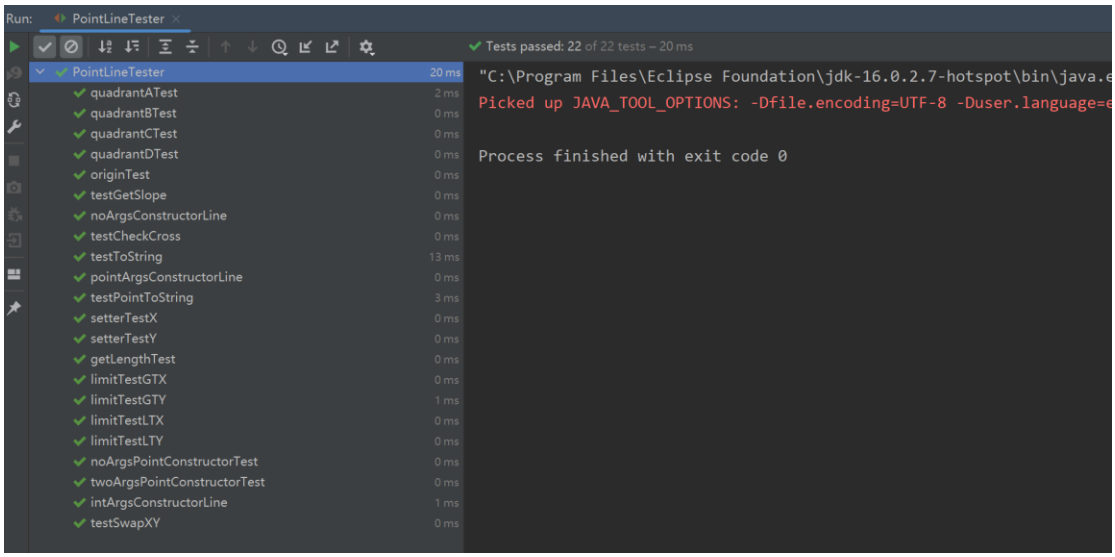
In the driver, a lot of instances and if-else, println, equals statements are written, the readability is not good and also difficult to maintain, it is even hard to know how many test are created.

The purpose of creating a driver is to make a direct comparation in unit test and driver-based tests.

3. Extensions

- **Increase the percent coverage of the provided testing class.**

  I have created 5 additional tests, including testSwapXY(), testPointToString(), testGetSlope(), testCheckCross(), testLineToString(). In the following pic, 22 tests all passed, including 5 student created tests.



- **Add more simple functionality to the points or lines like getting line slope.**

  I have added methods SwapXY for point class, getSlope for Line class

```
public void swapXY() {
    int temp = x;
    x = y;
    y = temp;
}
```

```
public double getSlope() {
    double deltaY = p2.getY() - p1.getY();
    double deltaX = p2.getX() - p1.getX();
    return deltaY / deltaX;
}
```

- **Add more complex functionality to the line class like checking to see if a line crosses another line.**

  I have added checkcross method for Line class

```
public boolean checkCross(Line l) {
    return !(this.getSlope() == l.getSlope());
}
```

- **Do a string override.**

  I have overridden the toString method for point class and line class

```
@Override
public String toString() {
    return "x:" + String.valueOf(x) + " y:" + String.valueOf(y);
}
```

```
@Override
public String toString() {
    return "Start : " + p1.toString()
            + " End : " + p2.toString();
}
```

- **Do an equals override.**

  I have overridden the equals as well as hashcode method for point class

```
@Override
public boolean equals(Object obj) {
    if (obj == this) {
        return true;
    }
    if (!(obj instanceof Point)) {
        return false;
    }
    Point p = (Point) obj;
    return p.x == x && p.y == y;
}
```

```
@Override
public int hashCode() {
    int hash = 17;
    hash = 31 * hash + x;
    hash = 31 * hash + y;
    return hash;
}
```

Grading Statement

Tests: all tests are passed
Point class: print method created and tested in the driver
Line class: print method created and tested in the driver
Driver: Point and Line class are tested in Driver
Misc: report is provided and code is written based on the requirement
Extensions: 5 extensions are implemented as follows

Increase the percent coverage of the provided testing class.

Add more simple functionality to the points or lines like getting line slope.

I have added methods SwapXY for point class, getSlope for Line class

Add more complex functionality to the line class like checking to see if a line crosses another line.

Do a string override.

Do an equals override.

For the above reasons, I think this work deserves a full mark.

	Possible	**Given**
Tests		
Provided Point Tests Pass (2 pts per test)	26	0
Provided Line Tests Pass (4 pts per test)	16	0
Point Class		
Print Method Created (Driver Test)	5	0
Line Class		
Line : Print method created (Driver test)	5	0
Driver		
Point tested as requested	10	0
Line tested as requested	10	0
Misc		
Report	5	0
Code Quality (correct indentation, comment blocks, variable naming, etc)	10	0
Not included in total possible:		
Does not compile	-100	0
Extensions (Not calculated without report)	15	0
Late penalty	-20	0
Creative or went above and beyond	10	0
Code contains warnings	-20	0
TOTAL POINTS POSSIBLE out of 100	87	0