Assignment 1

Pro & Cons of Designs

Factors to consider:

- Simplicity of code
- Efficiency when creating instances
- Efficiency when doing computations that require both coordinate systems
- Amount of memory used

Design 2

Pros

- Storing only the polar coordinates increases the simplicity of the code because there is no longer any use for the if statements (see below example)

Cons

- Since only one type of coordinate is stored, unnecessary conversions are being performed. For example, if Cartesian coordinates are passed in to the constructor, then, `getX` is called, the x coordinate will be used to convert to polar coordinates, then

```
// Design 1
public double getX()
  {
    if(typeCoord == 'C')
      return xOrRho;
    else
      return (Math.cos(
        Math.toRadians(yOrTheta)) * xOrRho
      );
  }
```

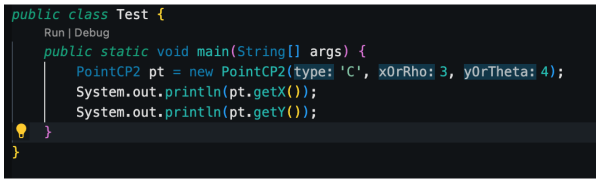
```
// Design 2: a significant simplification
public double getX()
  {
    return (Math.cos(
      Math.toRadians(yOrTheta)) * xOrRho
    );
  }
```

- Instantiation is efficient; rho and theta are stored as instance variables; O(1) time
- The computations are efficient; trig functions are relatively trivial

the polar coordinates will be converted right back to Cartesian coordinates when `getX` is called.

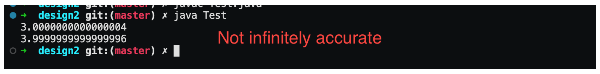
- The accuracy of the coordinates degrades over time; since the square root and trig functions is Java are not completely precise, the conversions will slowly decrease the accuracy of the coordinates. See the example shown below.

Test code:



```
public class Test {
    Run | Debug
    public static void main(String[] args) {
        PointCP2 pt = new PointCP2(type: 'C', xOrRho: 3, yOrTheta: 4);
        System.out.println(pt.getX());
        System.out.println(pt.getY());
    }
}
```

Output:



```
● ➜  design2 git:(master) ✗ java Test
  3.0000000000000004
  3.9999999999999996
○ ➜  design2 git:(master) ✗ 
```

Not infinitely accurate

Design 3

- Many of the same design flaws in design 2 are shared in design 3

Pros

- Storing only the Cartesian coordinates simplifies the code

Cons

- Since only one type of coordinate is stored, unnecessary conversions are because Cartesian coordinates are the most popular way of denoting points, so users/maintainers will most likely familiar with the Cartesian coordinates system

- Instantiation is efficient; x and y are stored as instance variables; O(1) time

being performed. For example, if polar coordinates are passed in to the constructor, then, `getRho` is called, rho will be used to convert to Cartesian coordinates, then the Cartesian coordinates will be converted right back to Cartesian coordinates when `getRho` is called.

Design 5

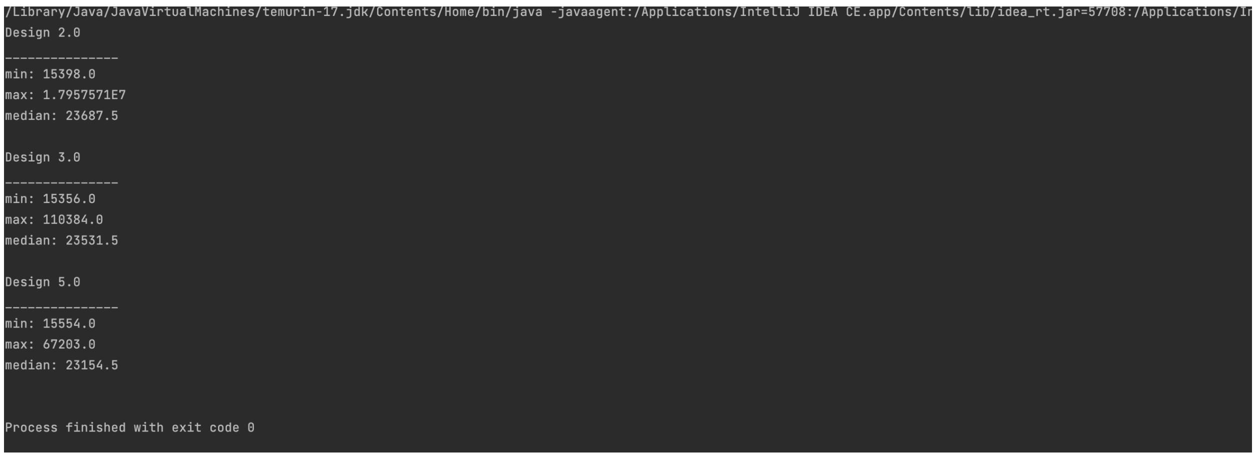
Pros

- Easily readable since the functionality has been abstracted into different classes

Cons

- Making design 2 & 3 sub classes does not increase the efficiency of the design
- Add complexity for the user since there are now two classes with very similar functionality

Runtime Comparisons



```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=57708:/Applications/In
Design 2.0
---------------
min: 15398.0
max: 1.7957571E7
median: 23687.5

Design 3.0
---------------
min: 15356.0
max: 110384.0
median: 23531.5

Design 5.0
---------------
min: 15554.0
max: 67203.0
median: 23154.5


Process finished with exit code 0
```

Run Time Results

Run Time Comparison Table

	Design 2	Design 3	Design 5
Minimum Run Time (ns)	15 398	15 356	15 554
Maximum Run Time (ns)	17 957 571	110 384	67 203
Median Run Time (ns)	23 687.5	23 531.5	23 154.5