Lab6 Part 1 -- 5 violations in Lab5

**1. Missing Access Modifiers for Instance Variables**

The x and y instance variables are declared without explicit access modifiers. Java coding standards recommend making class fields private and providing access via getter and setter methods to ensure encapsulation.

Original:  
double x;

double y;

Changed:

private double x;

private double y;

**2. Magic Numbers in hashCode**

The numbers 3 and 5 used in the hashCode method are hardcoded (commonly referred to as "magic numbers"). Magic numbers reduce code readability and should be replaced with named constants.

Before:

@Override

public int hashCode() {

    return (int)(3\*x + 5\*y);

}

Changed:

private static final int HASH\_MULTIPLIER\_X = 3;

private static final int HASH\_MULTIPLIER\_Y = 5;

@Override

public int hashCode() {

return (int)(HASH\_MULTIPLIER\_X \* x + HASH\_MULTIPLIER\_Y \* y);

}

3. **Inconsistent Use of Braces**

The equals method does not consistently use braces for if statements with single-line blocks. Even for single-line blocks, using braces enhances readability and prevents errors during code modifications.

Before:

if (this == obj)

    return true;

if (obj == null)

    return false;

if (getClass() != obj.getClass())

    return false;

Changed:

if (this == obj) { return true; }

if (obj == null) { return false; }

if (getClass() != obj.getClass()) { return false; }

4. **Class-Level Javadoc Comments Missing**

The Point class does not have a descriptive Javadoc comment explaining its purpose. Every class should have a Javadoc comment to describe its functionality, making the code easier to understand for others.

Before:

public class Point {

...

}

Changed:

/\*\*

\* Represents a point in 2D space with x and y coordinates.

\* Provides methods for distance calculation and object comparison.

\*/

public class Point {

...

}

**5. Equals Method Should Use Double.compare**

Direct comparison of floating-point numbers using != in the equals method can lead to issues due to precision errors. Floating-point numbers should be compared using Double.compare for consistent results.

Before:

if (x != other.x)

    return false;

if (y != other.y)

    return false;

Changed:

if (Double.compare(x, other.x) != 0) {

return false;

}

if (Double.compare(y, other.y) != 0) {

return false;

}