4. Objects and Classes :: The *this* Instance Variable

Every object has an instance variable named *this* which is automatically declared by the Java compiler.

One use of *this* is when the name of an instance variable conflicts with the name of a method parameter:

```
public class Point {
  ...
  public void move(double x, double y) {
    this.x = x;
    this.y = y;
  }
}

Point pete = new Point(10, 20);
pete.move(30, 40);
```

4. Objects and Classes :: The *this* Instance Variable (continued)

I avoid having to use *this* by naming my data members with a leading 'm' character and my method parameters with a leading 'p' character.

```
//********************************************************************************
// CLASS: Point (declared in Point.java)
//********************************************************************************
public class Point {
  public double mX;
  public double mY;

  // Constructor.
  public Point(double pX, double pY) {
    setX(pX);
    setY(pY);
  }

  // Accessor method for mX.
  public double getX() {
    return mX;
  }

  // Accessor method for mY.
  public double getY() {
    return mY;
  }
```

4. Objects and Classes :: The *this* Instance Variable (continued)

```
    // Mutator method for mX.
    public void setX(double pX) {
      mX = pX;
    }
    // Mutator method for mY.
    public void setY(double pY) {
      mY = pY;
    }
  }
```

4. Objects and Classes :: The *this* Instance Variable (continued)

One constructor can call another constructor of the same class using *this*:

```
public class Point {
  // Default constructor.
  public Point() {
    this(0, 0);  // Calls Point(double, double).
  }
  // A second constructor.
  public Point(double pX, double pY) {
    setX(pX);
    setY(pY);
  }
}
```