

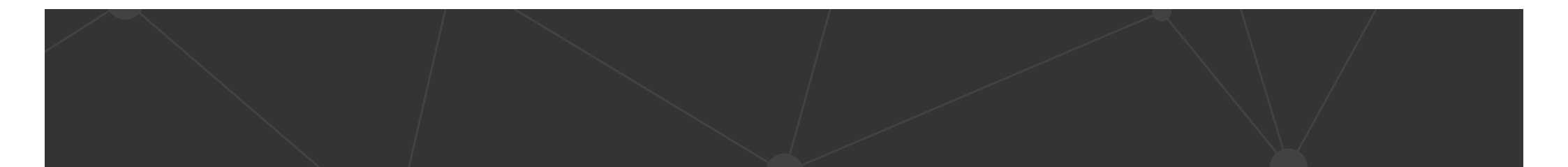


Essential of Object Oriented Programming

Alexandre Bergel

<http://bergel.eu>

23-08-2021



Today

The goal of today is to put *in practice* what we have seen last week

We will see a *small* exercise, but it is *essential*

I recommend that to redo the exercise on your own, without looking at the solution

Point example

A point is defined as having a *X and Y components*

First, we need *a way to create point*. You should therefore think in at least *one constructor*

Second, we could be able *to do some operations* between points. Summing and resting are two simple operations.

Write

A class that model points

An example that performs some operations between points

```
package cc3002.point;
```

```
/**
```

```
 * Model a 2D point and offer simple operations to manipulate points
```

```
 */
```

```
public class Point {
```

```
    private double x;
```

```
    private double y;
```

```
/**
```

```
 * Create a point from two coordinate
```

```
 * @param anX
```

```
 * @param anY
```

```
 */
```

```
public Point(double anX, double anY) { this.x = anX; this.y = anY; }
```

```
/**
```

```
 * Create a point at (0, 0)
```

```
 */
```

```
public Point() { this.x = 0; this.y = 0; }
```

```
...
```

```

/**
 * Create a point from another point.
 * This has the effect to copy a point
 * @param anotherPoint
 */
public Point(Point anotherPoint) { this.x = anotherPoint.x; this.y = anotherPoint.y; }

/**
 * Return the X coordinate
 * @return this point X
 */
public double getX() { return x; }

/**
 * Return the Y coordinate
 * @return this point Y
 */
public double getY() { return y; }

/**
 * Return a string representation of a point
 * @return the string that correspond to this point
 */
public String toString() {
    return "(" + this.getX() + ", " + this.getY() + ")";
}

/**
 * Sum two points, this and the provided argument point
 * @param anotherPoint
 * @return this summed with anotherPoint
 */
public Point add(Point anotherPoint) {
    return new Point(this.getX() + anotherPoint.getX(), this.getY() + anotherPoint.getY());
}

/**
 * Substracte two points, this and the provided argument point
 * @param anotherPoint
 * @return this substracted from anotherPoint
 */
public Point sub(Point anotherPoint) {
    return new Point(this.getX() - anotherPoint.getX(), this.getY() - anotherPoint.getY());
}
}

```

```
package cc3002.point;
```

```
/**
```

```
 * Example on how to use points
```

```
 */
```

```
public class PointExample {
```

```
    public static void main(String[] args) {
```

```
        Point p1 = new Point(3, 4);
```

```
        Point p2 = new Point(-4, 7);
```

```
        Point p3 = new Point();
```

```
        System.out.println("Value of p1 = " + p1.toString());
```

```
        System.out.println("Value of p2 = " + p2.toString());
```

```
        System.out.println("Value of p3 = " + p3.toString());
```

```
        System.out.println("Value of point copy = " + new Point(p2).toString());
```

```
        System.out.println("p1 + p2 = " + p1.add(p2).toString());
```

```
        System.out.println("p1 - p2 = " + p1.sub(p2).toString());
```

```
        System.out.println("(p1 - p2) + (p3 - p2) = " + ( p1.sub(p2).add(p3.sub(p2))).toString());
```

```
    }
```

```
}
```

License



Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

You are free to:

- Share: copy and redistribute the material in any medium or format
- Adapt: remix, transform, and build upon the material for any purpose, even commercially

The licensor cannot revoke these freedoms as long as you follow the license terms



Attribution: you must give appropriate credit



ShareAlike: if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original

Complete license: <https://creativecommons.org/licenses/by-sa/4.0/>



dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

f @ in / DCCUCHILE