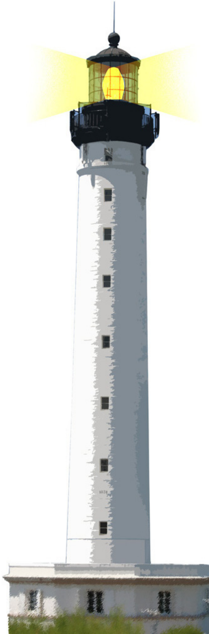
Learning Object-Oriented Programming and Design with TDD

Points as (real) Objects

Stéphane Ducasse

http://stephane.ducasse.free.fr



http://www.pharo.org

Core

Objectives

- Looking at two concrete implementations of Point
- Understanding the impact of strong API

Points in Java

Without getters and setters:

- boolean equals(Object obj) Determines whether or not two points are equal.
- void move(int x, int y) Moves this point to the specified location in the (x,y) coordinate plane.
- String toString() Returns a string representation of this point and its location in the (x,y) coordinate space.
- void translate(int dx, int dy) Translates this point, at location (x,y), by dx along the x axis and dy along the y axis so that it now represents the point (x+dx,y+dy).

Inherited from Point2D

- distance and clone

Points in Java

Getters and setters:

- Point getLocation() Returns the location of this point. (well this is to be polymorphic with Component - A location is just a point)
- void setLocation(double x, double y) Sets the location of this point to the specified double coordinates.
- void setLocation(int x, int y) Changes the point to have the specified location.
- void setLocation(Point p) Sets the location of the point to the specified location.
- double getX() Returns the X coordinate of this Point2D in double precision.
- double getY() Returns the Y coordinate of this Point2D in double precision.

Example

How to make our robot walks from distance in its current direction (in degree).

```
public class Bot {
 int direction = 0;
 Point position = new Point(0,0);
```

```
 public void go(int distance){
  position = (new Point((Math.round(Math.cos(Math.toRadians(direction))) * distance +
     position.x()),
      (Math.round(Math.sin(Math.toRadians(direction)) * distance + position.y())))) ;
  }
}
```

Analysis

- A poor data structure, not an object
- Arithmetic of Points is defined outside of them!
  - Points cannot sum themselves
  - Points cannot shape themselves (rounded, normal, reciprocal,...)
- When an object exposes a shallow API, it favors logic duplication in clients!

Go in Pharo

```
public void go(int distance){
  position = (new Point((Math.round(Math.cos(Math.toRadians(direction))) * distance +
      position.x()),
       (Math.round(Math.sin(Math.toRadians(direction)) * distance + position.y ())))) ;
  }
}
```

to

```
Bot >> go: aDistance
  "Return the point that is at a distance aDistance in the direction pointed by the
      receiver"
  position := position + (direction degreeCos @ direction degreeSin * aDistance)
      rounded
```

Points in Pharo

Point selectors

- r setR:degrees:, normalized, onLineFrom:to:, angleWith:, angle, onLineFrom:to:within:, rotateBy:about:, normal, degrees, rotateBy:centerAt:, theta, bearingToPoint:, distanceTo:
- >= > <= min:max: min: < closeTo: closeTo:precision: hash max: =
- negated, translateBy:, adhereTo:, scaleBy:, scaleTo:, scaleFrom:to:
- triangleArea:with: to:intersects:to: to:sideOf: isInsideCircle:with:with: sideOf:
- \ - * reciprocal / + min // abs max
- rectangle:, extent:, corner:
- roundUpTo: ceiling truncated truncateTo: roundTo: floor roundDownTo: rounded
- quadrantOf: leftRotated fourNeighbors grid: eightNeighbors nearestPointAlongLineFrom:to: sortsBefore: flipBy:centerAt: crossProduct: nearestPointOnLineFrom:to: dotProduct: squaredDistanceTo: insideTriangle:with:with: fourDirections directionToLineFrom:to: transposed reflectedAbout: sign octantOf: rightRotated

Simple example

```
Point >> abs
  "Answer a Point whose x and y are the absolute values of the receiver's x and y."

  ^ x abs @ y abs
```

Simple example

```
< aPoint
  "Answer whether the receiver is above and to the left of aPoint."
  "((100@200) < (330@400)) >>> true"
  "((100@200) < (330@100)) >>> false"

  ^ x < aPoint x and: [y < aPoint y]
```

Example

```
Point >> crossProduct: aPoint
  "Answer a number that is the cross product of the receiver and the
  argument, aPoint."
    ^ (x * aPoint y) – (y * aPoint x)
```

Example

```
Point >> degrees
  "Answer the angle the receiver makes with origin in degrees. right is 0; down is 90."
  | tan theta |
  ^ x = 0
    ifTrue:
      [ y >= 0
        ifTrue: [ 90.0 ]
        ifFalse: [ 270.0 ] ]
    ifFalse:
      [ tan := y asFloat / x asFloat.
      theta := tan arcTan.
      x >= 0
        ifTrue:
          [ y >= 0
            ifTrue: [ theta radiansToDegrees ]
            ifFalse: [ 360.0 + theta radiansToDegrees ] ]
        ifFalse: [ 180.0 + theta radiansToDegrees ] ]
```

Polymorphic

```
Point >> asPoint
  "Answer the receiver itself."

  ^ self
```

```
Object >> asPoint
  "Answer a Point with the receiver as both coordinates; often used to
  supply the same value in two dimensions, as with symmetrical gridding
  or scaling."

  ^ self @ self
```

- This way we can manage list of objects and easily convert them to point

```
{ 1 . 2 . 3 . 33@33 . 4} collect: [:a | a asPoint]
>> {1@1 . 2@2 . 3@3 . 33@33 . 4@4}
```

Point Arimethic

- Points know how to *, +, divide, substract themselves
- We can mix points, rectangles and number.

```
drawString: aString at: aPoint font: aFontOrNil color: aColor
  self
    drawString: aString
    in: (origin + aPoint extent: self clipRect extent)
    font: aFontOrNil
    color: aColor
```

- In Pharo Points are more than a data structure
- They embed behavior and hide the logic
- Functionality is pushed from clients to Point
- Point **offers** behavior: reuse here!

What you should know

- Objects are not data structures
- An object should encapsulate logic and lets its client reuse such logic!

A course by Stéphane Ducasse
http://stephane.ducasse.free.fr

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
http://mooc.pharo.org