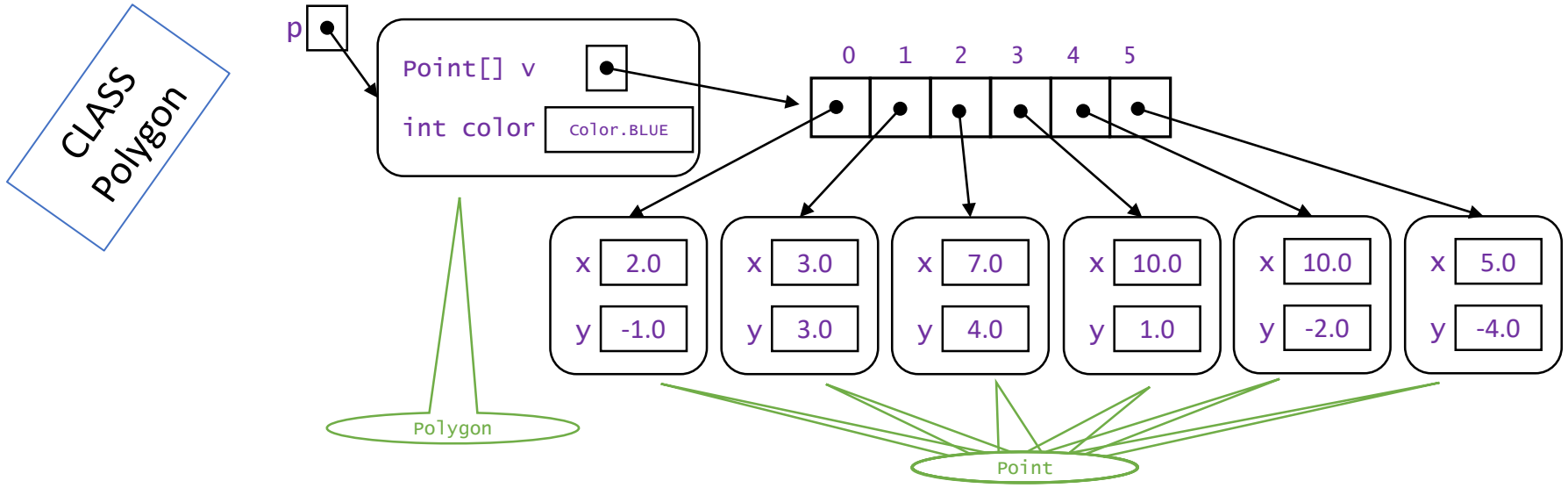
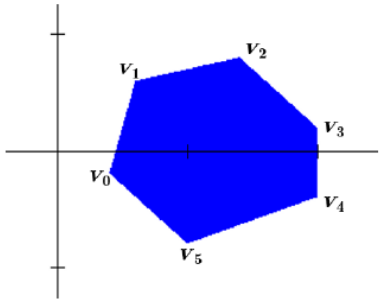
Lab activity 7.
(3 sessions)

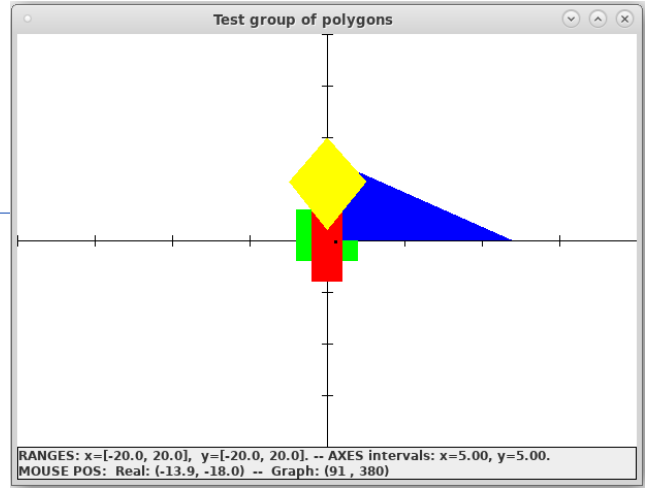
Group of Polygons

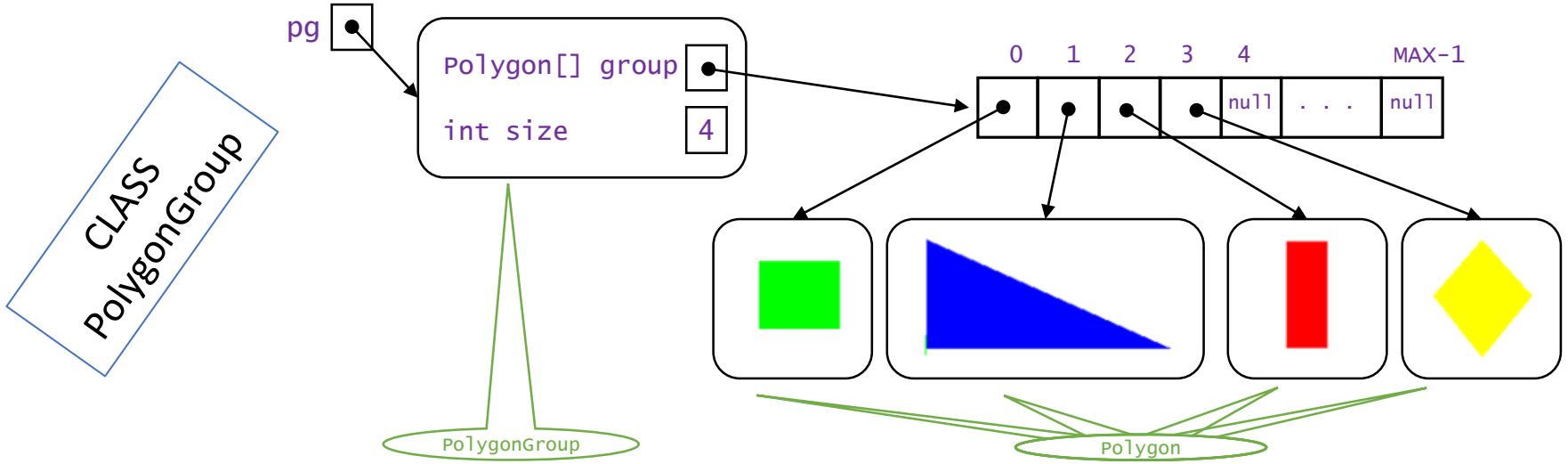
CC BY NC SA

Lab Activity 7









Activity 1: Preparation an download.

- Launch BlueJ, and create a new package `pract7` in project `iip` , then close BlueJ.
- Download all the files from `PoliformaT/resources/Laboratorio/Práctica_7/English/code` to `pract7`.
- Launch BlueJ again and open Package `pract7`

Activity 2: Implementation and test of Polygon class

- Complete Polygon.java file and compile it.
  - Complete the attribute declaration:
    - An array of `Point` (v)
    - and a `Color` (color)
  - Complete all methods: (Explanation in following slides)
    - **`Polygon(double[] x, double[] y)`** (create and initialize a new array of `Point` and set attribute `color` to `Color.Blue`)
    - **`double[] verticesX()`** (returns a new array of `double` containing the abscisses values of `v` with a length as `v.length`)
    - **`double[] verticesY()`** (returns a new array of `double` and size length as `v.length` containing the ordinates values of each `Point` of `v`)
    - **`Color getColor()`** (returns the `color` of the polygon, just return the attribute)
    - **`void setColor(Color nC)`** (updates the `color` of the polygon, just change the attribute)
    - **`double perimeter()`** (obtain the perimeter of the `Polygon` with an Array-Traversal algorithm)
    - **`void translate(double incX, double incY)`** (Moves all vertex of the `Polygon` with an Array-Traversal algorithm)
    - **`boolean inside(Point p)`** (checks if `Point p` is inside the `Polygon` by using method `cross` from class `Point`)

(See Appendix A)

Activity 2: Implementation and test of Polygon class

- **`Polygon(double[] x, double[] y)`** (create and initialize a new array of `Point` and set attribute `color` to `Color.Blue`)

  REMEMBER:
  - Declare = say of what type is something
    (attributes are declared outside the methods)
  - Initialise = create (with `new` in case of objects) if an array is created only length is defined, the array is filled with zeros or `null`
    (instance attributes are normally initialised inside the constructor method)
  - Fill = When initialized an array is filled with zeros or `null` (it depends on the declared type), but you can fill with any value of the declared type.
    (instance attributes can be filled inside the constructor method (<u>but not only</u>))

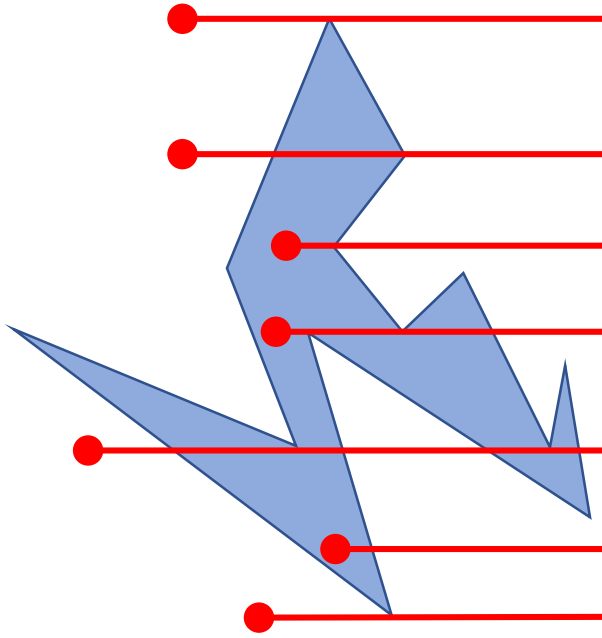
- **`double perimeter()`** (Obtain the perimeter of the `Polygon` with an Array-Traversal algorithm)
  - Use a loop in order to visit all the vertex of the Polygon and use the appropriate method of `Point` to measure each segment.

- **`void translate(double incX, double incY)`** (Moves all vertex of the `Polygon` with an Array-Traversal algorithm)
  - Use a loop in order to visit all the vertex of the `Polygon` and use the appropriate method of `Point` to translate each one.*

- **`boolean inside(Point p)`** (Checks if `Point p` is inside the polygon (See Appendix A) by using method `cross` from class `Point`)
  - Use a loop in order to visit all the vertex of the `Polygon` and use the appropriate method of `Point` check if `Point` p crosses each segment. Then use the algorithm described at the pdf document in order to find the answer of the method. (see next slide)

Activity 2: Implementation and test of Polygon class

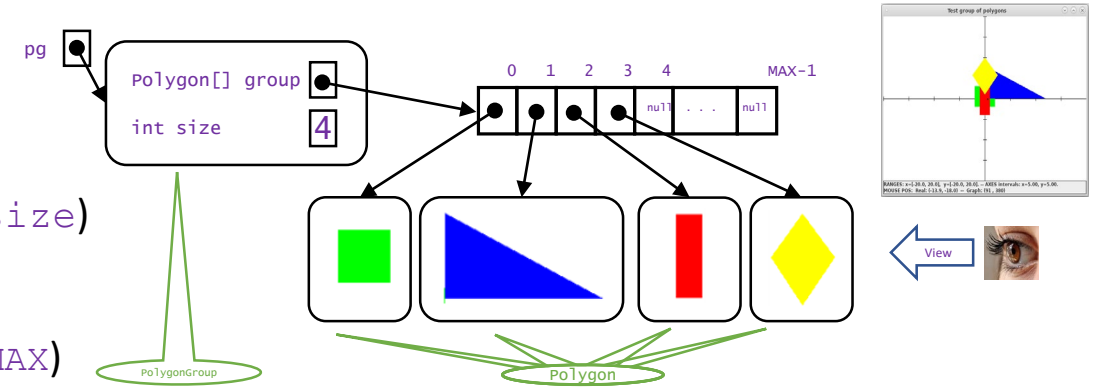
- **`boolean inside(Point p)`** (Checks if `Point p` is inside the polygon (See Appendix A) by using method `cross` from class `Point`)
  - Use a loop in order to visit all the vertex of the `Polygon` and use the appropriate method of **`Point`** to check if `Point` p crosses each segment. Then use the algorithm described at the pdf document in order to find the answer of the method.



	CROSS		LOW_CROSS	HIGH_CROSS	DONT_CROSS
(OUTSIDE) 0 =	0	+	0	0	13
(OUTSIDE) 0 =	0	+	0	2	11
(OUTSIDE) 2 =	1	+	1	1	10
(INSIDE) 1 =	0	+	1	1	11
(INSIDE) 3 =	1	+	2	2	8
(OUTSIDE) 8 =	4	+	4	0	5
(INSIDE) 1 =	1	+	0	0	12
(OUTSIDE) 2 =	0	+	2	0	11

Activity 3: Implementation and test of PolygonGroup class

- After testing `Poligon.class`, complete `PolygonGroup.java` file and compile it.
  - Complete the attribute declaration:
    - An constant integer value , (name: `MAX`)
    - An array of `Polygon` (name: `group`)
    - And an integer with the number of `Polygon`s stored in the array (name: `size`)



  - Complete all methods:
    - **`PolygonGroup()`** (create and initialize a new array of `Polygon` with length `MAX`)
    - **`boolean add(Polygon pol)`** (returns a `true` if adding the new `Polygon pol` to the array `group` was successful (it is not full))
    - **`int getSize()`** (returns the `size` of the polygon)
    - **`int search(Point p)`** (Obtain the index of the upper `Polygon` that contains `Point p` or `-1` if it doesn't exist)
    - **`boolean remove(Point p)`** (Removes the upper `Polygon` that contains `p`, displacing the rest, returns `true` or `false` if it exists or not)
    - **`void toFront(Point p)`** (Moves the upper `Polygon` that contains `p` to the position `size - 1` of the array, **displacing** the rest)
    - **`void toBack(Point p)`** (Moves the upper `Polygon` that contains `p` to the position `0` of the array, **displacing** the rest)
    - **`void translate(Point p, double incX, double incY)`** (Translates the upper `Polygon` that contains `p` using **its** method `translate`)
    - **`Polygon[] toArray()`** (returns a new Array of `Polygon` and length `size`, using the `Polygon`s from the array `group`)

Do this method BEFORE the next 4

Uses