IIP (E.T.S. d'Enginyeria Informàtica)
Curs 2021-2022

Pràctica 7. Arrays: una aplicació de gestió d'un grup de polígons en el pla

Professors d'IIP
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València

Índex

1 Objectius i treball previ a la sessió de pràctiques

Aquesta pràctica resumeix tots els conceptes vistos en l'assignatura i, en particular, els corresponents al *"Tema 7. Arrays: definició i aplicacions"*.

La seua realització pressuposa que, amb anterioritat al treball del laboratori, s'ha llegit detingudament la descripció del problema i l'organització de classes que es proposa per a la seua resolució.

La pràctica dura tres sessions en les quals es desenvoluparan les activitats proposades en aquest butlletí. Durant les dues primeres s'hauria de poder completar el codi de les classes `Polygon` i `PolygonGroup` plantejades, deixant la tercera sessió per a ultimar el codi que haguera quedat pendent de les sessions anteriors, completar el programa `Test7` i realitzar les últimes proves i correccions.

2 Descripció del problema

En aquesta pràctica s'abordarà el problema de gestionar un grup de polígons, que se suposen disposats sobre un pla.

Cada polígon ve donat per la seqüència dels seus vèrtexs (punts en el pla), i és d'un determinat color de farciment (color de la superfície del polígon) de manera que, a partir d'aquestes dades, es pot dibuixar en un espai de dibuix com els de la llibreria gràfica `Graph2D` utilitzada en les pràctiques anteriors. Un polígon es podrà traslladar en el pla i canviar-li el color de farciment.

Un grup de polígons és, com el seu propi nom indica, un agrupament de polígons al que es podran afegir nous elements, esborrar elements existents, o tractar de forma solidària a tots els elements del grup. També es pot seleccionar un polígon del grup per tractar-lo individualment, per exemple, per a desplaçar-lo o canviar-li el color.

Un comportament habitual en les aplicacions gràfiques en les quals es maneja un grup de figures, polígons en el cas que ens ocupa, és que es van superposant per l'ordre en què s'afigen al grup. Així doncs, d'haver-hi solapament entre les superfícies dels polígons, sota de tot es troba el més antic i a dalt del tot el més recent. Per exemple, el dibuix de la figura 1 representa gràficament un grup de polígons en el qual s'han afegit, per ordre, un rectangle verd, un triangle blau, un rectangle roig i un quadrilàter groc.

Aquest ordre es pot canviar a conveniència, seleccionant un polígon per a enviar-lo al fons, portar-lo davant del tot, etc.

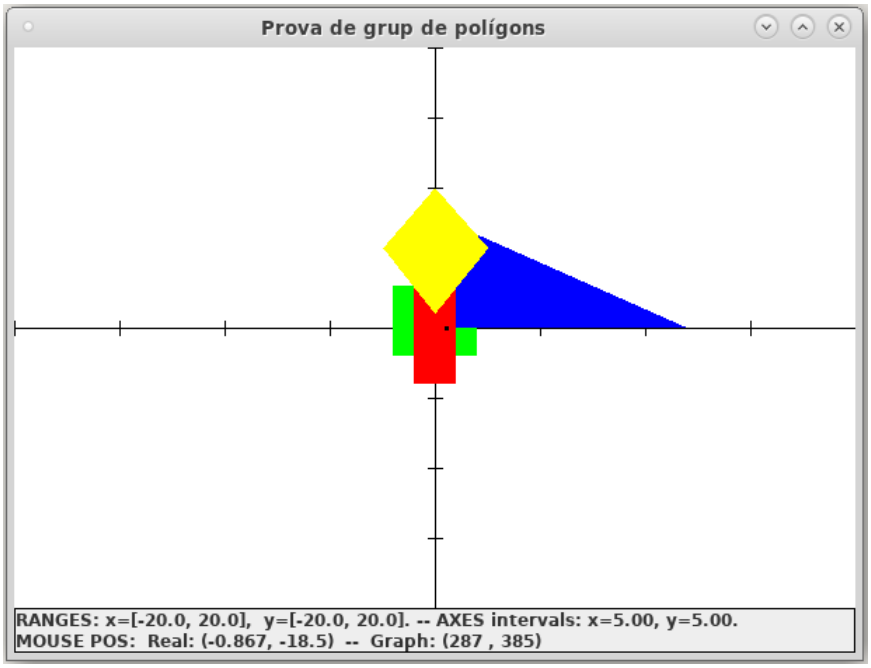

Figura 1: Grup de polígons.

Quan s'ha representat gràficament un grup de polígons, la manera en què es pot seleccionar un polígon del grup és clicant sobre un punt de la seua superfície que estiga a la vista, és a dir, que no quede ocult per altres polígons superposats. En la part gràfica d'aquesta pràctica, es detectarà el punt del clic de manera que, en el grup de figures, conegut aquest punt, es podrà accedir al polígon assenyalat. Per a això, el grup de polígons revisarà els seus elements per ordre invers, d'amunt cap a avall, cercant el primer que continga a aquest punt: aquest és el polígon a seleccionar, atès que se superposa a la resta de polígons que contingueren el mateix punt. En l'exemple de la figura 1 es veu com, per a assenyalar el rectangle roig, s'ha clicat un dels punts visibles de la seua superfície, en concret el de coordenades `(0.6,0.8)`.

Per donar compte del comportament descrit, en aquesta pràctica es desenvoluparà una xicoteta aplicació formada per les classes que es mostren en la figura 2, i que són:

- Classe `Polygon`. Representació i tractament de polígons en el pla.
- Classe `PolygonGroup`. Representació i tractament d'un grup de polígons.
- Classe `Test7`. Programa de prova que crea un grup de figures i permet provar diferents accions sobre el grup. Per a ajudar a comprovar l'efecte d'aquestes accions, aquest programa visualitza gràficament els resultats, usant la llibreria `Graph2D` del paquet `graph2D`.

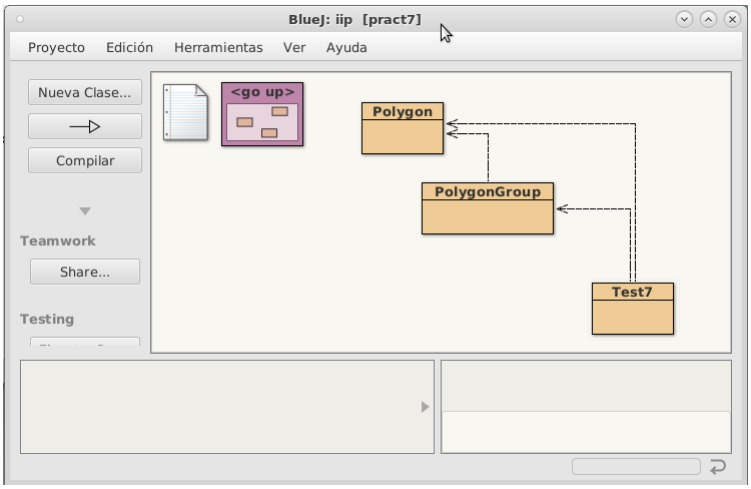

Figura 2: Classes del paquet `iip[pract7]`.

Activitat 1: preparació del paquet BlueJ `pract7`

1. Descarregar els fitxers `Polygon.java`, `GroupPolygon.java`, `Test7.java` disponibles en la carpeta de material per a la pràctica 7 de *PoliformaT*.

2. Obrir el projecte *BlueJ* de treball de l'assignatura (`iip`) i crear un nou paquet `pract7`.

3. Agregar al paquet `pract7` les classes descarregades amb l'opció (*Edició - Agregar Classe des d'Arxiu*). Comprovar que les seues primeres línies inclouen la directiva `package pract7;`, que indica que són classes del paquet.

3 La classe `Polygon`

Un `Polygon` ve donat per una seqüència de n vèrtexs, $v_0, v_1, \ldots, v_{n-1}$, de manera que els seus costats són els segments $\overline{v_0v_1}, \overline{v_1v_2}, \ldots, \overline{v_{n-2}v_{n-1}}, \overline{v_{n-1}v_0}$. A més, té un color de farciment.

La classe es defineix mitjançant els següents atributs d'instància privats:

- `v`: un array d'objectes de tipus `Point` (la classe desenvolupada en la pràctica 5), en el qual s'emmagatzemaran en posicions successives els n vèrtexs de tipus `Point` del polígon. Per a cada polígon de la classe, la longitud de `v` haurà de ser del nombre n de vèrtexs o costats del polígon.

- `color`: un color de farciment, de tipus `Color` del paquet `java.awt`.

En la figura 3 es mostra un exemple de `Polygon` i la representació gràfica del polígon corresponent.

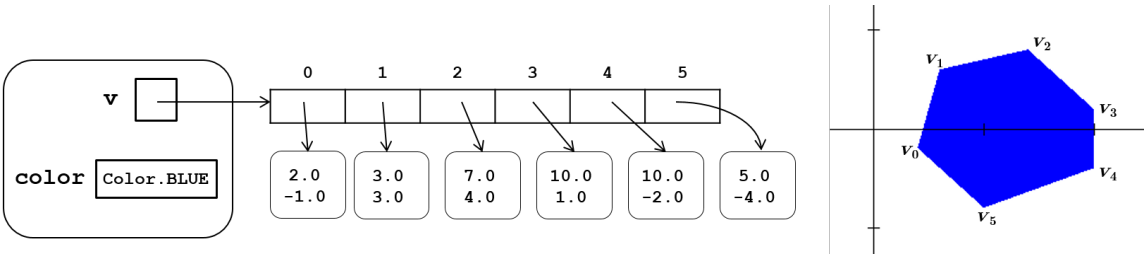

Figura 3: Un `Polygon` i la seua representació gràfica.

Els mètodes de la classe són els que es descriuen a continuació:

- Constructor `public Polygon(double[] x, double[] y)`: Construeix un `Polygon` a partir d'un array `x`, amb les abscisses x_0, x_1, x_2, ..., x_{n-1} dels seus vèrtexs, i un array `y`, amb les ordenades y_0, y_1, y_2, ..., y_{n-1} dels seus vèrtexs, $n > 0$. Els vèrtexs defineixen un polígon els costats del qual s'estenen d'un vèrtex al següent, i tancant-se en (x_0, y_0). Per exemple, si `x` és l'array `{2.0, 3.0, 7.0, 10.0, 10.0, 5.0}` i `y` l'array `{-1.0, 3.0, 4.0, 1.0, -2.0, -4.0}`, ha de construir l'objecte de la figura 3.

  Per defecte, el polígon és de color blau (`Color.BLUE`).

- Mètodes `public Color getColor()` i `public void setColor(Color nC)`, consultor i modificador del color, respectivament.

- Mètodes `public double[] verticesX()` i `public double[] verticesY()` que, respectivament, retornen un array amb les successives abscisses dels vèrtexs i un array amb les successives ordenades dels vèrtexs.

- Mètode `public void translate(double incX, double incY)`, que trasllada els vèrtexs del polígon: `incX` en l'eix X, `incY` en l'eix Y.

- Mètode `public double perimeter()`, que retorna el perímetre del polígon.

- Mètode `public boolean inside(Point p)`, que, aplicant l'*algorisme del raig* que es discuteix en l'apèndix A, comprova si el `Point p` és interior al polígon. Si el punt és interior al polígon retorna `true`, i si el punt és exterior al polígon retorna `false`.

Activitat 2: implementació i prova de la classe `Polygon`

Implementar els atributs i mètodes de la classe `Polygon`. Noteu que, per a poder usar les classes `Point` i `Color`, s'han inclòs les següents clàusules d'importació:

```
import java.awt.Color;
import pract5.Point;
```

Els mètodes s'aniran provant a mesura que es vaja complentant el seu codi. En primer lloc, es comprovarà el mètode constructor, creant un triangle de vèrtexs $(0,0)$, $(0,3)$ i $(4,0)$, com en la figura 4, i inspeccionant l'objecte obtingut, com es mostra en la figura 5.

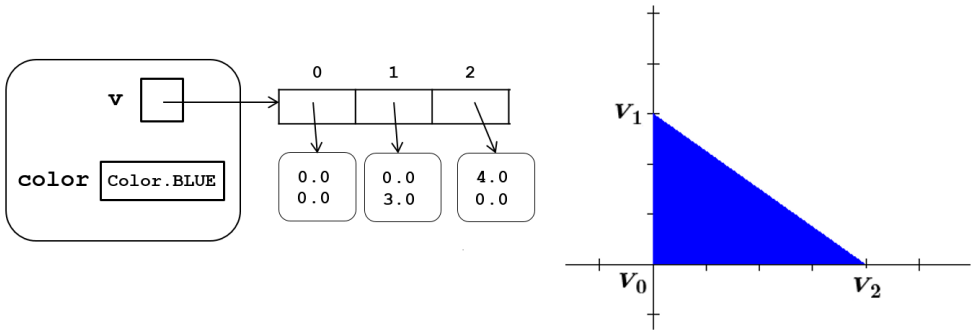

Figura 4: Un `Polygon` amb 3 vèrtexs (triangle) i la seua representació gràfica.

A continuació, s'aniran provant els mètodes d'instància, sobretot els que gestionen els vèrtexs del polígon, com `verticesX`, `verticesY`, `translate` i `perimeter`, tal com es mostra per a aquest últim en la figura 6.

En el cas de `translate`, si es prova de traslladar el triangle anterior `1.0` en abscisses i `1.0` en ordenades, com en la figura 7, s'ha de comprovar que `verticesX` i `verticesY` retornen, respectivament, els arrays `{1.0, 1.0, 5.0}` i `{1.0, 4.0, 1.0}`.

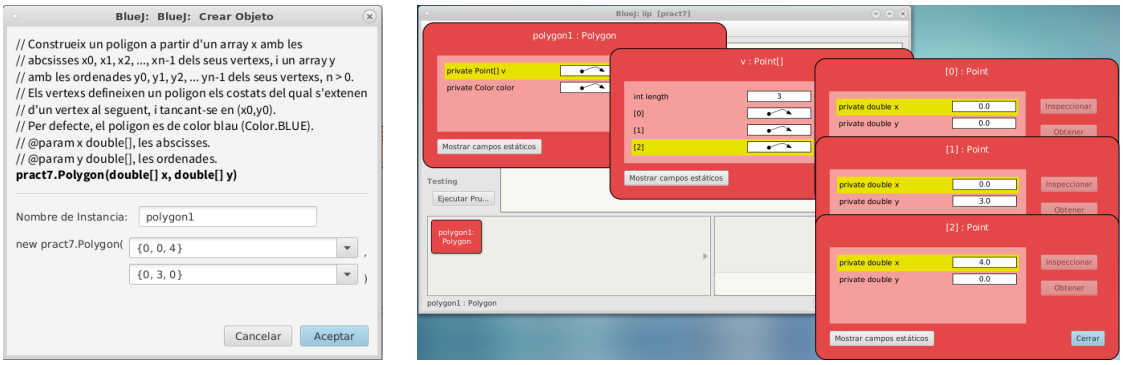

Figura 5: Construcció i examen d'un `Polygon` en *BlueJ*.

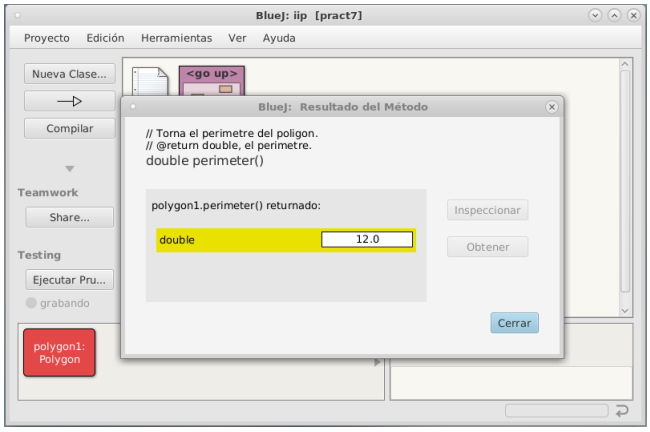

Figura 6: Prova del mètode `perimeter` sobre el triangle de la figura 5.

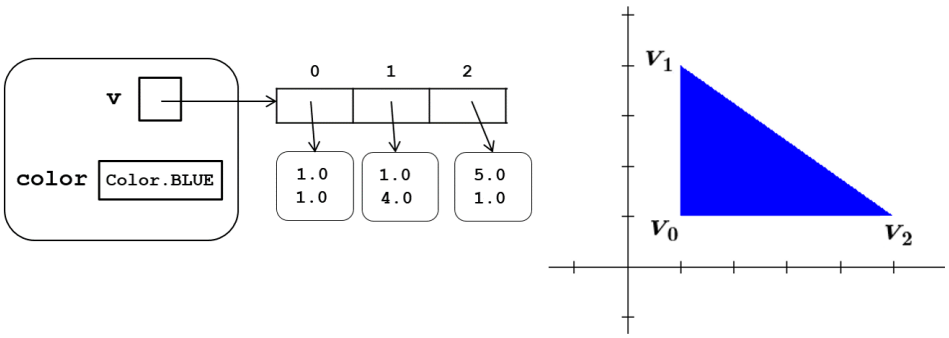

Figura 7: Resultat de traslladar el triangle de la figura 4.

Per a poder provar el mètode `inside`, en la zona de codi s'importarà la classe `Point` del paquet `pract5`, i es crearà un `Point` de coordenades $(1, 1)$, que es podrà arrossegar al banc d'objectes per a usar-lo com a paràmetre del mètode (veure figura 8). Com el punt és interior al triangle, el resultat ha de ser `true`. Repetir les proves amb els punts $(4, 5)$ (punt exterior) i $(-1, 3)$ (punt exterior per al qual el raig passa pel vèrtex $(0, 3)$ del triangle).

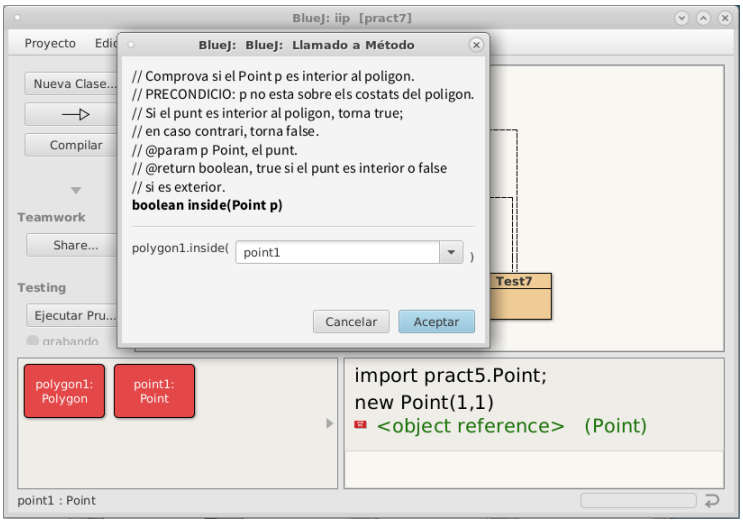

Figura 8: Prova del mètode `inside` sobre el triangle de la figura 5 i el `Point (1.0,1.0)`.

4 La classe `PolygonGroup`

Un `PolygonGroup` vindrà donat per la seqüència de polígons que formen part del grup, que estarà ordenada per l'ordre d'inserció en el grup, de més antic a més recent. Tots els grups hauran de tindre una capacitat o nombre màxim de polígons.

La classe es defineix mitjançant els següents atributs:

- `MAX`: atribut públic de classe constant, de tipus `int`, que dóna la capacitat del grup, iniciada a 10.
- `group`: atribut privat d'instància, un array d'objectes de tipus `Polygon` i longitud `MAX`, en el qual s'emmagatzemaran, en posicions successives i per ordre d'inserció, els polígons del grup. La resta de posicions de l'array es mantenen a `null`.
- `size`: atribut privat d'instància, la talla o nombre de polígons en el grup.

En la figura 9 es mostra un objecte `PolygonGroup` que correspon al grup de la figura 1.

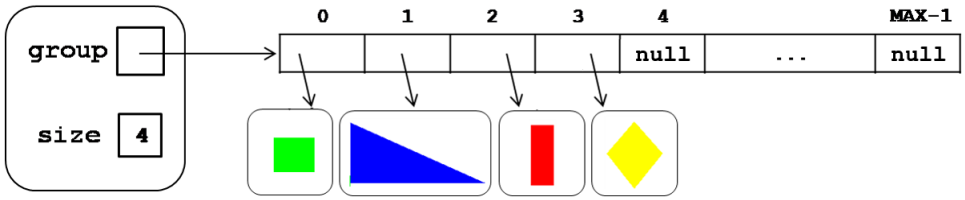

Figura 9: Un `PolygonGroup` de talla 4.

Els mètodes de la classe són els que es descriuen a continuació:

- Constructor `public PolygonGroup()`: Construeix un `PolygonGroup` buit, iniciant l'array `group` a un array de longitud `MAX` (amb totes les seues components a `null`, per defecte) i deixant la talla `size` a 0, com en la figura 10.
- Mètode `public int getSize()`, consultor de la talla del grup.
- Mètode `public boolean add(Polygon pol)`, que afig al grup, a dalt del tot, el polígon `pol`, retornant `true`. Si s'excedeix la capacitat del grup, llavors el polígon no es pot afegir i el mètode retorna `false`.

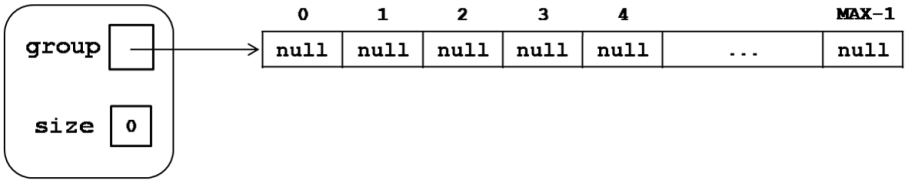

Figura 10: Un `PolygonGroup` de talla 0.

- Mètode `public Polygon[] toArray()`, que retorna un array de longitud igual a la talla del grup amb la seqüència de polígons del grup, per ordre des del de més avall al de més amunt.
- Mètode auxiliar `private int search(Point p)`, que cerca en el grup descendentment, de més amunt a més avall, el primer polígon que conté a `p`. Si el troba, retorna el seu índex en l'array `group`; si no existeix, retorna -1.

  Aquest mètode s'usarà en els mètodes `translate`, `remove`, `toFront` i `toBack` següents, per trobar la posició en l'array `group` del polígon assenyalat per un punt `p`.
- Mètode `public void translate(Point p, double incX, double incY)`, que trasllada en el pla el polígon seleccionat mitjançant el punt `p`. Les abscisses dels seus vèrtexs s'incrementen en `incX` i les ordenades en `incY`. No fa res si no hi ha cap polígon que continga a `p`.
- Mètode `public boolean remove(Point p)`, que elimina del grup el polígon seleccionat mitjançant el punt `p`, retornant `true`. Si el punt `p` no està contingut en cap polígon, el mètode retorna `false`.

  Noteu que en la implementació d'aquest mètode s'haurà de procurar que els polígons restants en el grup, després de l'eliminació, continuen apareguent en posicions consecutives de l'array, des de la 0 endavant, i per ordre d'inserció. La component de l'array següent a l'últim polígon restant en el grup s'haurà de posar a `null`.

  Els següents mètodes no fan res si no hi ha cap polígon en el grup que continga a `p`.
- Mètode `public void toFront(Point p)`, que situa al capdavant del grup, a dalt del tot, el polígon seleccionat mitjançant el punt `p`.
- Mètode `public void toBack(Point p)`, que situa al fons del grup, sota de tot, el polígon seleccionat mitjançant el punt `p`.

Activitat 3: implementació i prova de la classe `PolygonGroup`

Implementar els atributs i mètodes de la classe `PolygonGroup`. Noteu que, com en la classe `Polygon`, per poder usar la classe `Point`, s'ha inclòs la clàusula d'importació:

```
import pract5.Point;
```

En anar acabant els mètodes, s'aniran fent les proves corresponents. En primer lloc, es crearà i inspeccionarà el grup buit, tal com es mostra en la figura 11.

Per provar els mètodes `add`, `getSize`, `toArray`, `search` i `remove`, per simplicitat es crearan tres triangles idèntics de coordenades $(0,1),(2,2)$ i $(0,3)$ (arrays d'abscisses `{0.0, 2.0, 0.0}` i d'ordenades `{1.0, 2.0, 3.0}`) i s'afegiran al grup (tres triangles perfectament superposats), inspeccionant el grup a mesura que va creixent. En la figura 12 es mostra l'estat que s'ha d'observar després d'afegir els tres polígons.

Comprovar a continuació que `toArray` retorna un array de tres polígons (veure figura 13).

Seguidament, s'han de fer tres operacions d'eliminació del primer polígon que continga el punt $(1,2)$, inspeccionant el grup després de cada eliminació. Atès que els tres polígons són idèntics i contenen aquest punt, s'ha de comprovar que s'elimina en primer lloc el que ocupa la posició 2 de l'array `group` (el més recent), en segon lloc el que ocupa la posició 1 (el següent més recent)

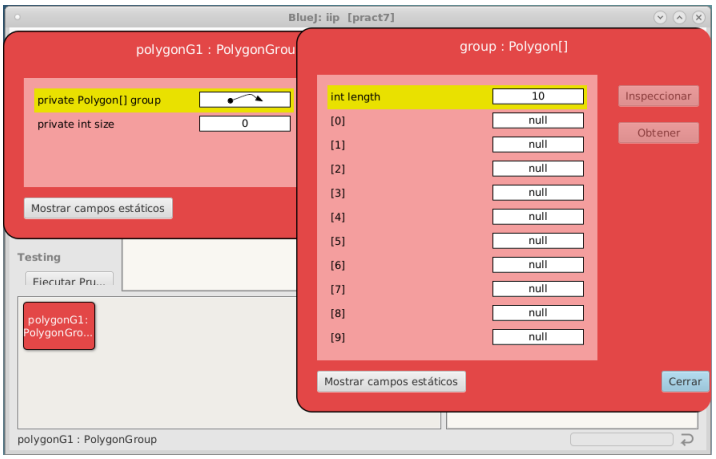

Figura 11: Prova del constructor de `PolygonGroup`.

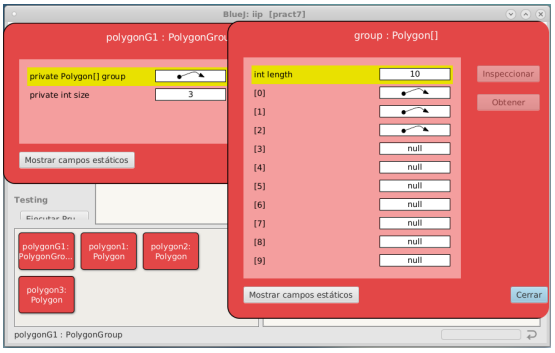

Figura 12: Prova del mètode `add` de `PolygonGroup` i inspecció del resultat.

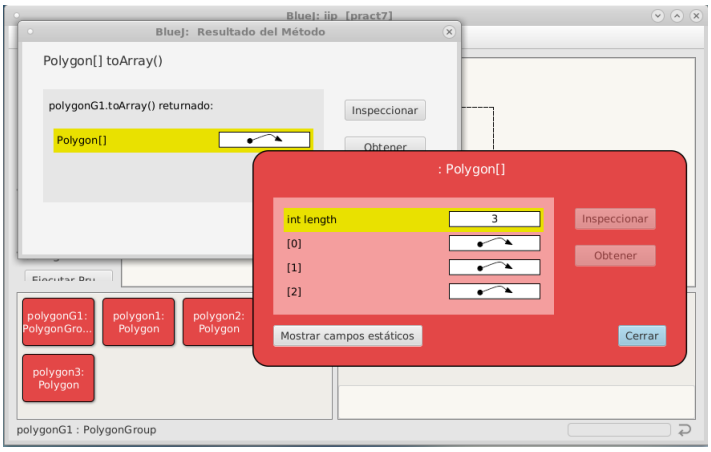

Figura 13: Prova del mètode `toArray` de `PolygonGroup`.

i, finalment, el que ocupa la posió 0 (el més antic), quedant el grup buit. Una última operació d'eliminació continuaria deixant el grup buit.

En la figura 14 s'observa com queda el grup després de fer la primera crida de les anomenades a `remove`.

Després d'aquestes proves preliminars, es podrà fer una bateria de proves més completa per als mètodes de la classe amb el programa `Test7` de la següent secció.

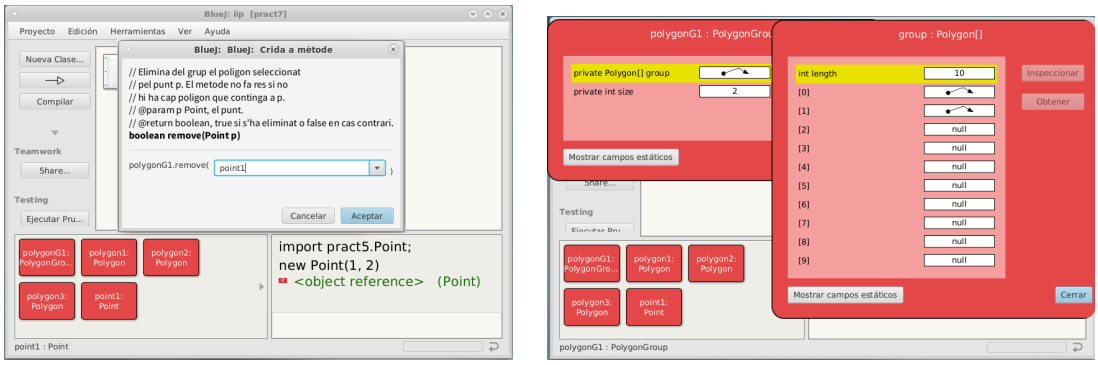

Figura 14: Prova de cerca i eliminació d'un polígon: crida al mètode `remove` de `PolygonGroup` i inspecció del resultat.

5 La classe `Test7`

La classe `Test7` és un programa que, a més de provar diferents accions sobre un grup de polígons, facilita la visualització del comportament d'aquestes accions mitjançant una representació gràfica. Per a això, té implementat un mètode `drawGroup` que mostra el grup en un espai de dibuix de la llibreria `Graph2D`. També fa ús del mètode `nextMousePressed` de `Graph2D` per a recuperar les coordenades del punt clicat en la finestra de dibuix i, d'aquesta manera, seleccionar de forma senzilla un polígon del grup.

En el programa es crea un grup de polígons com el de la figura 1, sobre el qual es podran fer les proves.

Activitat 4: finalització i prova del mètode `main` de `Test7`

La classe inclou els mètodes `menu(Scanner)` i `submenu(Scanner)` que permeten a l'usuari seleccionar diferents accions a provar sobre el grup de polígons. S'ha de llegir detingudament el mètode `main` i completar el seu codi perquè s'execute el mètode de `PolygonGroup` que corresponga, segons el cas seleccionat.

A continuació s'haurà d'executar el programa per poder fer tantes proves com siga necessari, a través de les diferents opcions del menú.

6 Validació de les classes

Quan el teu professor ho considere convenient, deixarà disponibles en *PoliformaT* unes classes de prova per a validar el teu codi. En aquest cas, hauràs de descarregar els arxius corresponents sobre el directori del paquet `pract7` i reobrir el teu projecte `iip` des de *BlueJ*.

En general, per a passar els tests correctament, cal assegurar-se que s'usen els mateixos identificadors d'atributs i mètodes proposats en aquest document i en la documentació dels arxius `.java` que se't proporcionen, seguint estrictament les característiques sobre modificadors i paràmetres proposats en la capçalera d'aquests.

Activitat 5: validació de la classe `Polygon`

1. Per usar el test d'aquesta activitat, has d'haver validat prèviament el mètode `cross` de la classe `Point` (usant el test de validació `PointUnitTest.class` del paquet `pract5`), ja que s'utilitza en el mètode `inside` de la classe. Qualsevol error en la classe `Point` probablement causaria errors en el test de la classe `Polygon`.

2. Tria l'opció *Test All* del menú contextual que apareix en fer clic amb el botó dret del ratolí sobre la icona de la classe *Unit Test*. Com sempre, s'executaran un conjunt de proves sobre

els mètodes implementats de la classe corresponent, comparant resultats esperats amb els realment obtinguts.

3. Igual que en pràctiques anteriors, si els mètodes estan bé, apareixeran marcats amb el símbol ✓ (green color) en la finestra *Test Results* de *BlueJ*. Per contra, si algun dels mètodes no funciona correctament, llavors apareixerà marcat mitjançant el símbol `X`. Si selecciones qualsevol de les línies marcades amb `X`, en la part inferior de la finestra es mostra un missatge més descriptiu sobre la possible causa d'error.

4. Si, després de corregir errors i recompilar la classe, la icona de la *Unit Test* està ratllada a quadres, llavors tanca i torna a obrir el projecte *BlueJ*.

Activitat 6: validació de la classe `PolygonGroup`

Per passar el test de la classe `PolygonGroup`, tria l'opció *Test All* del menú contextual que apareix en fer clic amb el botó dret del ratolí sobre la icona de la *Unit Test* de la classe corresponent. Igual que en l'activitat anterior, s'executaran un conjunt de proves sobre els mètodes implementats de la classe `PolygonGroup`, comparant resultats esperats amb obtinguts. Per passar aquest test, és requisit imprescindible que les classes `Point` i `Polygon` estiguen correctament implementades. Per tant, assegura't d'haver passat primer els tests d'aquestes dues classes anteriorment.

A El mètode `inside`

Com es va comentar en la pràctica 5, el mètode del raig és un algorisme basat en el teorema de Jordan, que permet de manera senzilla comprovar si un punt és interior o exterior a un polígon:

Es llança un raig des del punt i es compta el nombre de vegades que el raig travessa el perímetre del polígon; tenint en compte que els encreuaments *d'entrada* i *d'eixida* del raig en el polígon s'alternen en la direcció del raig, i que l'últim encreuament és necessàriament d'eixida, llavors:

- Si el nombre d'encreuaments és parell, el nombre de vegades que el raig entra en el polígon és igual al número que vegades que ix, per la qual cosa el punt està fora del polígon.

- Si és imparell, hi ha un encreuament més d'eixida que d'entrada, per la qual cosa el punt està dins.

En la figura 15 es veu un exemple de com un raig llançat des d'un punt interior al polígon, creua el seu perímetre un nombre imparell de vegades.

El mètode es pot implementar recorrent tots els costats del polígon i comptant el nombre de costats que són travessats pel raig. Només cal tindre la precaució de comptabilitzar correctament el cas en què el raig travessa un vèrtex, punt que pertany a dos costats del polígon. En aquest cas, es pot considerar el següent:

- Si, com succeeix amb el vèrtex v_8 del polígon de la figura 15, el vèrtex connecta dos costats, un per l'extrem més baix i un altre pel més alt, llavors el raig travessa el perímetre, i només s'ha de comptar un encreuament.

- Si, com succeeix amb els vèrtexs v_3 i v_6 del mateix polígon, el vèrtex connecta dos costats, tots dos per l'extrem més baix, o tots dos per l'extrem més alt, llavors no travessa el perímetre, i no s'ha de comptar cap encreuament o, equivalentment, comptar un nombre parell.

- Cal notar que si el raig transcorre d'una banda del polígon paral·lel a l'eix X, no s'ha de comptar cap encreuament d'aquest costat.

Per tal de tindre en compte aquesta observació respecte als vèrtexs, es pot adoptar el conveni de comptar un encreuament del perímetre quan es travessa un costat, per exemple, per l'extrem més baix i no comptar-lo quan es travessa un costat per l'extrem més alt (o viceversa).

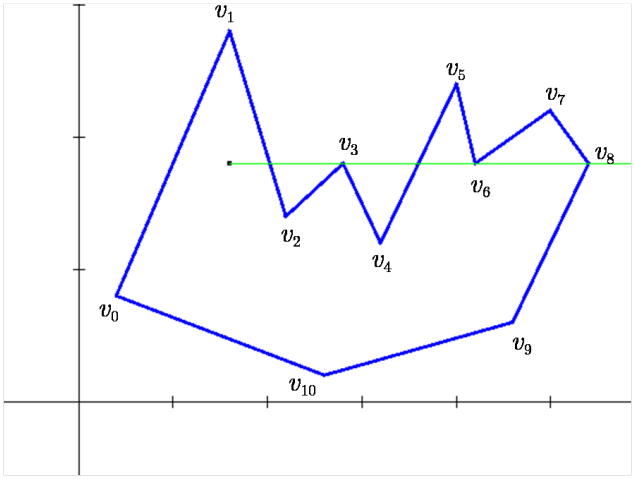

Figura 15: Mètode del raig.

La següent iteració realitza el comptatge d'encreuaments dels costats d'un polígon per un raig llançat des d'un punt `p`, usant el mètode `cross` de la classe `Point` i adoptant el conveni anterior per als vèrtexs:

```
int nCuts = 0, n = v.length;
int cross;
for (int i = 0; i < n - 1; i++) {
    cross = p.cross(v[i], v[i + 1]);
    if (cross == Point.CROSS || cross == Point.LOW_CROSS) { nCuts++; }
}
cross = p.cross(v[n - 1] , v[0]);
if (cross == Point.CROSS || cross == Point.LOW_CROSS) { nCuts++; }
```

Així doncs, la implementació del mètode `inside` de `Polygon` ha de realitzar aquest comptatge, i acabar retornant `true` si `nCuts` és senar, i `false` en cas contrari.