PRG (E.T.S. d'Enginyeria Informàtica)
Curs 2019-2020

Pràctica 5. Seqüències enllaçades: una aplicació de gestió d'un grup de polígons en el pla

(2 sessions)

Departament de Sistemes Informátics i Computació
Universitat Politècnica de València

Índex

1 Objectius i treball previ a la sessió de pràctiques

Aquesta pràctica és un exercici de desenvolupament d'una estructura de dades, en la qual s'usaran les seqüències enllaçades presentades en l'últim tema de l'assignatura. La implementació dels mètodes de la classe desenvolupada donarà motiu a exercitar els algorismes bàsics sobre tals seqüències: inserció i eliminació de dades, recorreguts i cerques, i altres canvis senzills en l'ordre de les dades.

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 dues sessions en les quals es desenvoluparan les activitats proposades en aquest butlletí. Durant la primera sessió s'haurien de poder completar les activitats 1 a 5 plantejades, i les activitats 6 i 7 durant la segona sessió.

2 Descripció del problema

En la pràctica 7 d'IIP s'abordava el problema de gestionar un grup de polígons, que se suposaven disposats sobre un pla, i que es descrivia de la següent forma.

Cada polígon es defineix per la seqüència dels seus vèrtexs (punts en el pla), i és d'un determinat color (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`[1] del paquet `graph2D`. Un polígon es podrà traslladar en el pla i canviar-li el color.

Un grup de polígons és, com el seu propi nom indica, un agrupament de polígons al qual 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 a tractar-lo individualment, com 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 al capdavant (davant del tot), etc.

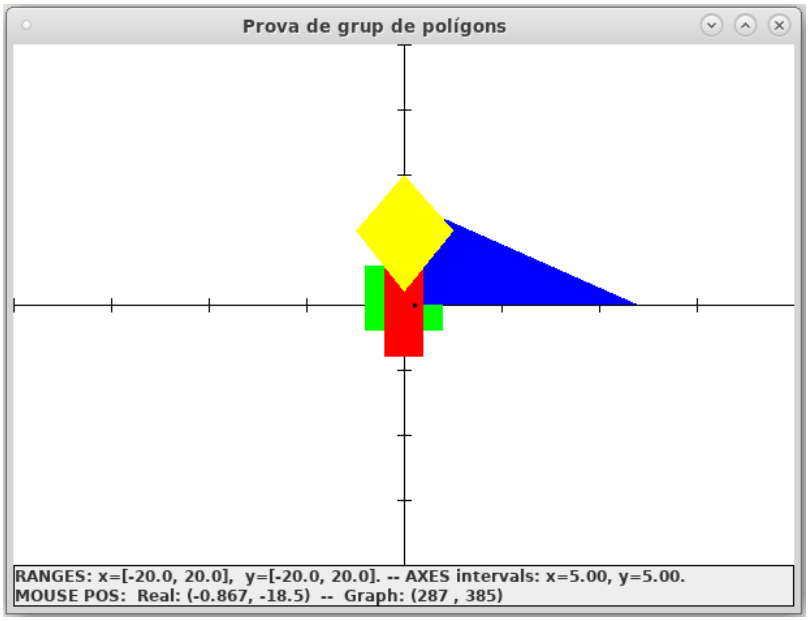


Figura 1: Grup de polígons.

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, en el grup de polígons es revisaran els seus elements per ordre, de més amunt a més a baix (des del front cap al fons), 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, el de coordenades `(0.6,0.8)` en concret.

[1] Instal·lada i usada en la pràctica 1 de PRG.

Per a donar compte del comportament descrit, en aquella pràctica es va desenvolupar una xicoteta aplicació formada per les següents classes:

- 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`.

Les classes anteriors usaven la classe `Point`, punts `(x, y)` en el pla cartesià, desenvolupada en una pràctica anterior.

En aquesta pràctica es tornaran a utilitzar aquestes classes. En concret:

- Les classes `Point` i `Polygon` se suposen ja implementades i disponibles per al seu ús. En el material per a la pràctica es proporcionen `Point.class` i `Polygon.class`. En la secció 3 es descriuen els mètodes de la classe `Polygon`.
- La classe `PolygonGroup` tindrà la mateixa interfície de mètodes, però ara es canviarà l'estructura dels seus objectes: si en la pràctica d'IIP s'usava un array per a emmagatzemar els polígons que formen part del grup, ara s'usarà una seqüència enllaçada. Els mètodes de la classe s'hauran d'implementar en conseqüència al llarg d'aquesta pràctica. Per a desenvolupar les seqüències enllaçades, s'usarà una classe `NodePol`, node la dada del qual és de tipus `Polygon`.
- Atès que la classe `Test7` usa únicament els mètodes públics de `PolygonGroup`, i aquests mantindran el mateix perfil i significat, en aquesta pràctica es podrà reutilitzar per a realitzar proves dels mètodes, visualitzant gràficament el seu resultat, fent ús de la llibreria gràfica `Graph2D` (usada ja en la pràctica 1). Així, la classe `Test7` s'anomena `Test5` i l'única diferència amb `Test7` és que, per tal de fer la lectura validada de dades des de teclat, s'utilitzen els mètodes de la classe `CorrectReading` (del paquet `utilPRG`) implementada en la pràctica 4.

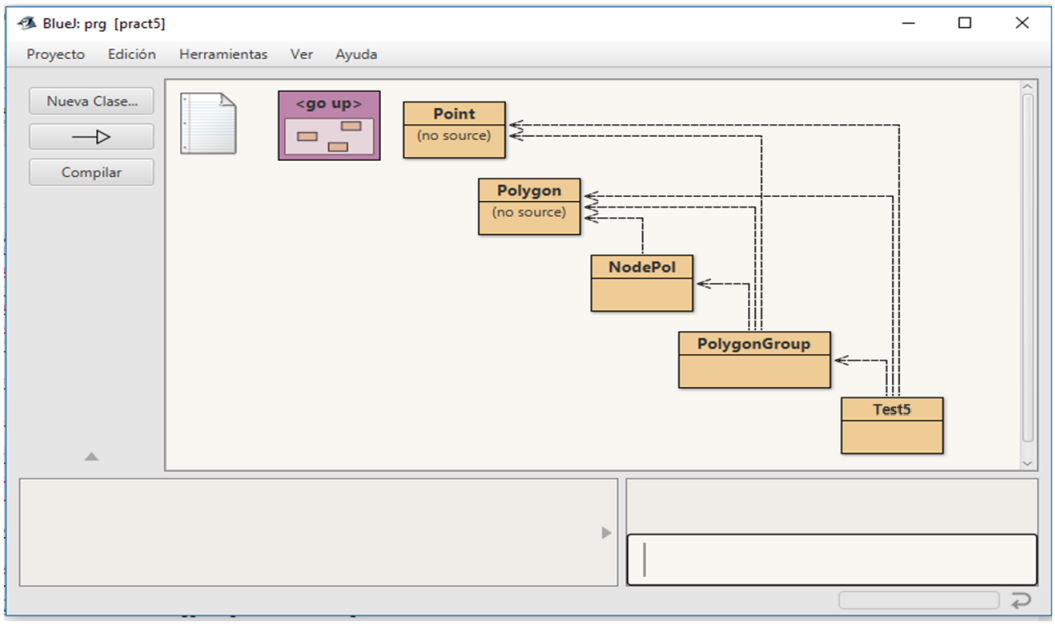


Figura 2: Classes del paquet `prg[pract5]`.

2.1 Activitat 1: preparació del paquet `pract5`

a) Descarregar els fitxers `Point.class`, `Polygon.class`, `NodePol.java`, `PolygonGroup.java` i `Test5.java` disponibles en la carpeta de material per a la pràctica 5 de *PoliformaT*.

b) Obrir en *BlueJ* el projecte de treball de l'assignatura `prg` (des d'un equip propi o mitjançant connexió remota a un equip del laboratori) i crear un nou paquet `pract5`.

c) Agregar al paquet `pract5` les classes `NodePol.java`, `PolygonGroup.java` i `Test5.java` descarregades. Es podran agregar les classes `Point.java` i `Polygon.java` que es van desenvolupar en les pràctiques 5 i 7 d'IIP, respectivament; si no es tinguera una versió validada d'aquestes classes o si es preferira, en el seu lloc es pot copiar en la carpeta del paquet el codi `.class` de totes dues classes, que s'haurà pogut descarregar de *PoliformaT*.

d) La llibreria gràfica (classe `Graph2D` del paquet `graph2D`) es facilita com una llibreria en el fitxer `graphLib.jar` i la seua documentació es proporciona en el fitxer `docGraph2D.zip` (en la carpeta *PRG:recursos/Laboratorio/Librería gráfica* de *PoliformaT*).

Aquesta llibreria s'ha de carregar de la manera adequada (indicant on es troba el fitxer `jar`, que s'haurà descarregat prèviament). Si la pràctica es realitza en el laboratori mitjançant connexió remota, ja la tindràs carregada i, per tant, pots obviar els passos d.1) i d.2) descrits a continuació. Si la pràctica es realitza en un equip propi:

d.1) Situa el fitxer `graphLib.jar` en el projecte de pràctiques `prg` i, des del menú *Preferències/Llibreries* de *BlueJ*, opció *Add File*, afegeix el fitxer `graphLib.jar`. Hauràs d'engegar novament *BlueJ* perquè la llibreria es carregue.

d.2) Descomprimeix el fitxer `docGraph2D.zip` en el projecte `prg` per tal de consultar la documentació de la classe `Graph2D` (fitxer `Graph2D.html`) quan siga necessari.

3 Interfície de la classe `Polygon`

Com ja es va veure en la pràctica 7 d'IIP, 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.

Els mètodes de la classe són els que es descriuen a continuació, i ja estan implementats en la classe `Polygon.class` que es proporciona com a material.

- Constructor `public Polygon(double[] x, double[] y)`: construeix un `Polygon` a partir d'un array `x` amb les abscisses $\mathtt{x}_0$, $\mathtt{x}_1$, $\mathtt{x}_2$, ..., $\mathtt{x}_{n-1}$ dels seus vèrtexs, i un array `y` amb les ordenades $\mathtt{y}_0$, $\mathtt{y}_1$, $\mathtt{y}_2$, ..., $\mathtt{y}_{n-1}$ dels seus vèrtexs, sent $n > 0$. Els vèrtexs defineixen un polígon els costats del qual s'estenen d'un vèrtex al següent, i tancant-se en $(\mathtt{x}_0, \mathtt{y}_0)$. 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)`, 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`.

Notar que els mètodes `verticesX()`, `verticesY()` i `getColor()` permeten obtenir les dades d'un polígon que necessita el mètode `fillPolygon` de la llibreria `Graph2D`, el qual dibuixa un polígon en una finestra a partir de les abscisses i ordenades dels seus vèrtexs. Per exemple, en la figura 3 es mostra un fragment de codi, en el *CodePad* de *BlueJ*, que crea un triangle blau i el dibuixa en una finestra gràfica.

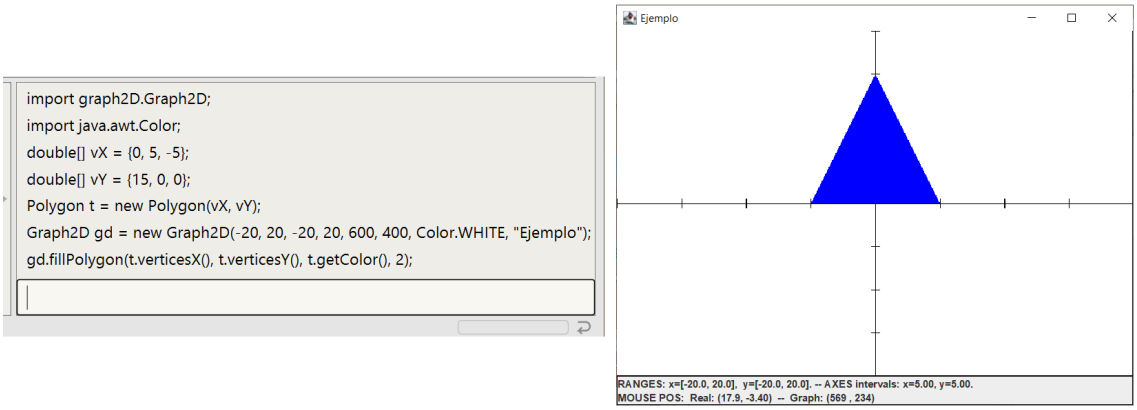


Figura 3: Exemple de creació d'un polígon i posterior dibuix en una finestra gràfica.

4 La classe `NodePol`

La classe `NodePol` és anàloga a la classe `NodeInt` vista en el tema 5, excepte que les dades han de ser de tipus `Polygon`. S'usa per a implementar seqüències enllaçades de polígons.

4.1 Activitat 2: declaració d'atributs de la classe `NodePol`

S'ha de completar la classe `NodePol` amb la definició dels seus atributs. Els mètodes constructors, anàlegs als de `NodeInt`, s'han deixat ja resolts.

5 La classe `PolygonGroup` implementada mitjançant seqüències enllaçades

Un `PolygonGroup` vindrà donat per la seqüència de polígons que formen part del grup. S'implementarà emmagatzemant els polígons en una seqüència enllaçada.

Atès que una de les operacions primordials en la gestió del grup, la de seleccionar un polígon mitjançant un clic, requereix cercar el polígon per ordre de més amunt a més a baix en el grup, el més indicat és que la seqüència vinga en aquest ordre, de manera que el primer en la seqüència siga el polígon al capdavant del grup, i l'últim el del fons. Per aquest motiu, com cada nou polígon que s'afija al grup s'ha de superposar als altres, s'inserirà al capdavant.

Per a facilitar altres operacions, com la de portar al fons, és convenient mantenir en el grup una altra referència al final de la seqüència. Així doncs, la classe es defineix mitjançant els següents atributs (activitat 3):

- `front`: atribut privat d'instància de tipus `NodePol`, node al capdavant de la seqüència de polígons.
- `back`: atribut privat d'instància de tipus `NodePol`, node final de la seqüència de polígons.
- `size`: atribut privat d'instància de tipus `int`, la talla o nombre de polígons en el grup.

En la figura 4 es mostra un objecte `PolygonGroup` que correspon al grup de la figura 1.

Els mètodes de la classe s'han d'implementar en les activitats 4, 5 i 6.

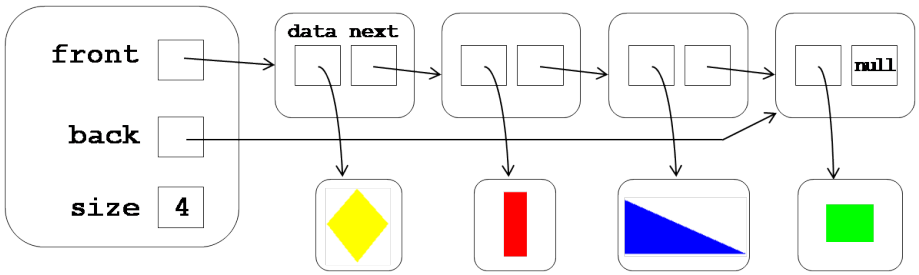


Figura 4: Un `PolygonGroup` de talla 4.

5.1 Activitat 3: declaració dels atributs de la classe `PolygonGroup`

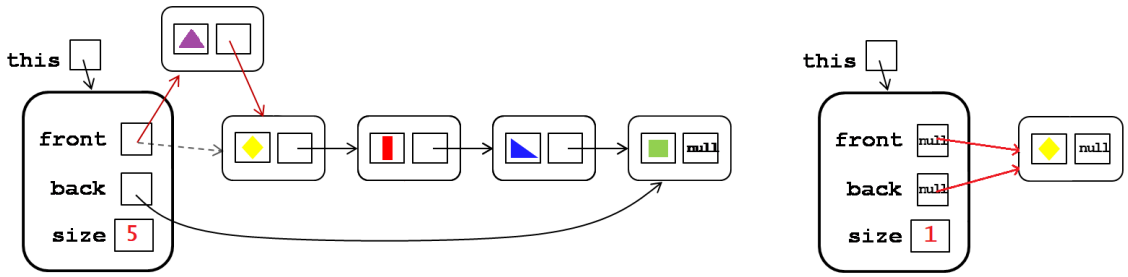
Editar la classe `PolygonGroup.java` per a completar la declaració dels seus atributs segons la descripció anterior.

5.2 Activitat 4: implementació i prova dels mètodes constructor `PolygonGroup`, `add`, `getSize` i `toArray`

Per a completar la classe, en aquesta activitat es començarà implementant els mètodes més senzills.

- Constructor `public PolygonGroup()`: construeix un `PolygonGroup` buit. Les referències `front` i `back` es deixaran a `null`, i la talla `size` a 0.
- Mètode `public int getSize()`, que retorna la talla o nombre de polígons en el grup.
- Mètode `public void add(Polygon pol)`, que afig al capdavant del grup, a dalt del tot, el polígon `pol`.

  Noteu que `pol` s'ha d'inserir en un nou node al capdavant de la seqüència de polígons (veure figura 5(a)). Si la inserció es fa en el grup buit, el nou node serà al mateix temps el primer i l'últim, per la qual cosa a més de `front` i `size`, caldrà actualitzar l'atribut `back` (veure figura 5(b)).



(a) Si el grup no està buit (b) Si el grup està buit

Figura 5: Canvis que realitza el mètode `add` sobre un grup de polígons quan s'afegeix un nou polígon a la seqüència.

- 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 a baix al de més amunt, és a dir, des del fons fins al front. El grup no es modifica. Per exemple, per a un grup com el de la figura 4 ha de retornar un array com el figura 6.

  Aquest mètode s'ha deixat resolt i resulta útil per a realitzar les proves del programa `Test5`. Notar que, com l'ordre dels polígons en l'array ha de ser l'invers al de la seqüència, en copiar els successius polígons de la seqüència es recorre el array en sentit descendent.

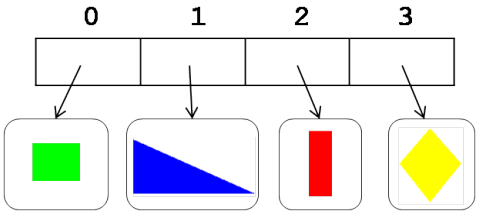


Figura 6: Array resultant de `toArray` sobre el grup de la figura 4.

Per a fer unes proves preliminars es pot usar el programa `Test5`. S'ha d'examinar el mètode `main` i comprovar que realitza les següents accions:

1. Crea un polígon verd, un blau, un roig i un groc, i els va afegint a un grup inicialment buit.
2. Escriu en l'eixida estàndard la talla del grup.
3. Usant el mètode auxiliar `drawGroup` definit en la pròpia classe `Test5`, obté amb el mètode `toArray` un array amb els polígons del grup en l'ordre en què es van afegir, i els dibuixa en aquest ordre en una finestra `Graph2D`.
4. Desplega un menú d'opcions, entre les quals es troba acabar el programa.

Així doncs, si els mètodes constructor, `add` i `getSize` són correctes, en executar el `main` de `Test5`, s'escriu en l'eixida estàndard que el grup té talla 4, i la seua representació gràfica és com la de la figura 1. Triar l'opció 0 de menú per a acabar la prova.

5.3 Activitat 5: implementació i prova dels mètodes `search` i `translate`

- Mètode auxiliar `private NodePol[] search(Point p)`, que cerca en el grup descendentment, de més amunt a més a baix, el primer polígon que conté a `p`. Retorna un array de tipus `NodePol`, de manera que la component 1 té una referència al node en el qual s'ha trobat el polígon, i la component 0 una referència al node anterior. Si el polígon trobat és el primer de la seqüència, la component 0 del resultat val `null`. Si el polígon no es troba, la component 1 del resultat val `null`.

  Aquest mètode s'usarà en els mètodes `remove`, `toFront`, `toBack` següents, per a trobar el node de la seqüència que continga el polígon assenyalat per un punt `p`.

  Per a implementar-ho, en el cos del mètode es podrà fer en primer lloc una cerca del primer node la dada del qual siga un polígon que continga a `p` (comprovant-ho amb el mètode `inside` de `Polygon`):

  ```
  NodePol aux = front, prevAux = null;
  while (aux != null && !aux.data.inside(p)) {
      prevAux = aux;
      aux = aux.next;
  }
  ```

  El mètode acabarà retornant un array les components del qual siguen `prevAux` i `aux`:

  ```
  NodePol[] s = new NodePol[2];
  s[0] = prevAux; s[1] = aux;
  return s;
  ```

- 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`. Aquest mètode no canvia la superposició relativa dels polígons en el grup.

  Aquest mètode s'implementarà cercant el primer node `mark` el polígon del qual continga a `p`:

  ```
  NodePol[] s = this.search(p);
  NodePol mark = s[1];
  ```

  Si existeix tal polígon (`mark != null`), llavors només cal aplicar al polígon `mark.data` el mètode `translate` de `Polygon`, que modifica les seues coordenades segons el desitjat.

Per a provar aquests mètode es podrà usar el programa `Test5`. El menú que desplega el mètode `main` permet seleccionar un polígon del grup per a aplicar-li una operació, redibuixant el grup en l'estat resultant; això possibilita fer les següents proves:

- Seleccionar mitjançant un clic el polígon del fons, demanar que es trasllade, i comprovar que s'ha trobat en el grup aquest polígon i que canvia correctament la seua posició en el pla.
- Repetir l'anterior per al polígon del front.
- Repetir l'anterior per a un polígon intermedi.
- Seleccionar mitjançant un clic un punt exterior a qualsevol polígon del grup, i comprovar que si es demana fer un trasllat del polígon seleccionat, llavors no es produeix cap canvi.

5.4 Activitat 6: implementació i prova dels mètodes `remove`, `toBack` i `toFront`

Una vegada que el mètode `search` és correcte, en aquesta activitat s'han de completar la resta de mètodes de la classe.

Tots aquests mètodes realitzen una acció que suposen una alteració de la seqüència enllaçada de polígons del grup. Reben com a paràmetre un punt `p` i, mitjançant el mètode `search`, han de cercar el primer node en la seqüència la dada del qual siga un polígon que continga a `p`, per a, en cas que existisca, eliminar-ho, portar-ho al capdavant, o portar-ho al fons del grup. En el que segueix, suposarem que tots ells executen en primer lloc el codi que ve a continuació.

```
NodePol[] s = this.search(p);
NodePol prevMark = s[0], mark = s[1];
```

De tal manera que en `mark` es tindria la referència al node, i en `prevMark` al node anterior.

- Mètode `public boolean remove(Point p)`, que elimina del grup el polígon seleccionat mitjançant el punt `p`, tornat `true`. Si el punt `p` no està contingut en cap polígon, el mètode torna `false`.

  Per a implementar aquest mètode, si s'haguera trobat un node la dada del qual fóra un polígon que conté a `p`, s'eliminarà aquest node de la seqüència. Es distingeixen dos casos:

  - El node a eliminar és el primer de la seqüència (el del front), açò és, `mark` és igual a `front` (o, el que és el mateix, `prevMark` és `null`). En aquest cas, `front` passarà a ser el següent node de la seqüència (`null` si era l'únic), com es pot veure en l'exemple de la figura 7 a).
  - El node a eliminar té un node que li precedeix, açò és, `mark` no és `front` (o `prevMark` és distint de `null`). En aquest cas, caldrà actualitzar el següent de `prevMark`, com en l'exemple de la figura 7 b).

  A més, caldrà actualitzar l'atribut `size`. I si el node a eliminar és el del fons (`mark` és igual a `back`), caldrà actualitzar també l'atribut `back` (veure l'exemple de la figura 8).

  El mètode no fa res si no es troba cap polígon que continga al punt.

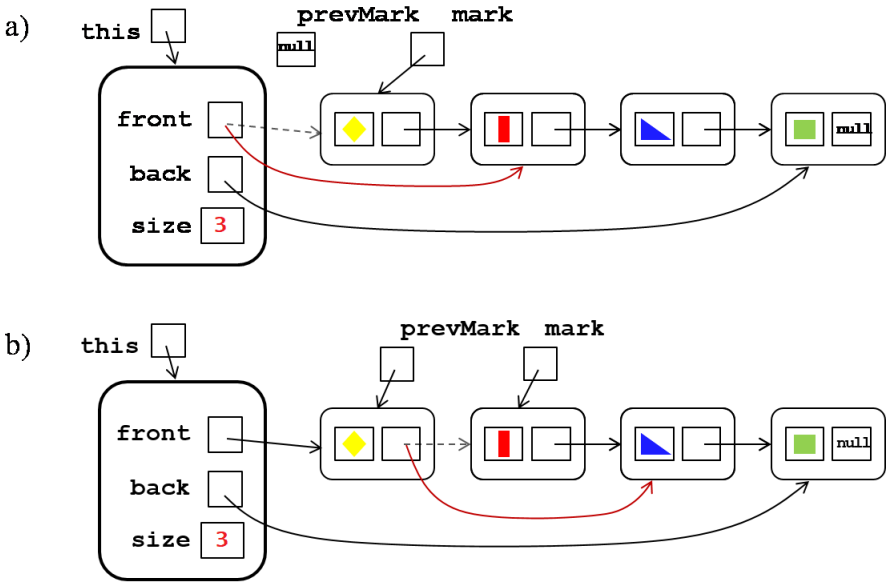


Figura 7: Canvis que realitza el mètode `remove` sobre el grup de la figura 4 quan s'ha clicat el polígon del front (polígon que es troba en el primer node de la seqüència) i quan s'ha clicat el rectangle roig (polígon en un node que té un anterior).

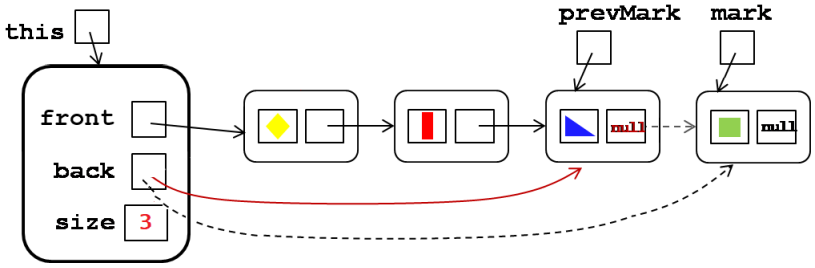


Figura 8: Canvis realitzats pel mètode `remove` sobre el grup de la figura 4 quan s'ha clicat el polígon del fons (polígon que es troba en el darrer nodo de la seqüència).

Una vegada implementat, es provarà amb el programa `Test5`, seleccionant i eliminant el polígon del front, un intermedi, el del fons i, finalment, l'únic polígon que quede en el grup (el programa escriu en l'eixida la talla del grup després de cada eliminació). També es provarà de clicar un punt exterior a qualsevol polígon, comprovant que el mètode no fa res.

- Mètode `public void toFront(Point p)`, que situa al capdavant del grup, a dalt del tot, el polígon seleccionat mitjançant el punt `p`. Si no hi ha cap polígon que continga a `p`, el mètode no fa res. En el codi que implementa aquest mètode, caldrà comprovar en primer lloc que s'haja trobat tal polígon, i que en aquest cas no estiga ja al capdavant del grup; en cas contrari, no cal fer res.

  Feta aquesta comprovació, i tenint en compte que el node a traslladar tindrà un que li precedisca (no és el primer), cal traure'l de la seqüència i situar-lo al capdavant (veure l'exemple de la figura 9 a) i b)). A més, si el node traslladat fóra el del fons, cal actualitzar també l'atribut `back` (veure l'exemple de la figura 10).

  Es repetiran les següents proves sobre el grup del programa `Test5`: provar de portar al capdavant el polígon que ja està en el front, portar al capdavant un polígon intermedi, i portar al capdavant el polígon del fons; comprovar igualment que el mètode no fa res si es clica un punt exterior a qualsevol polígon.

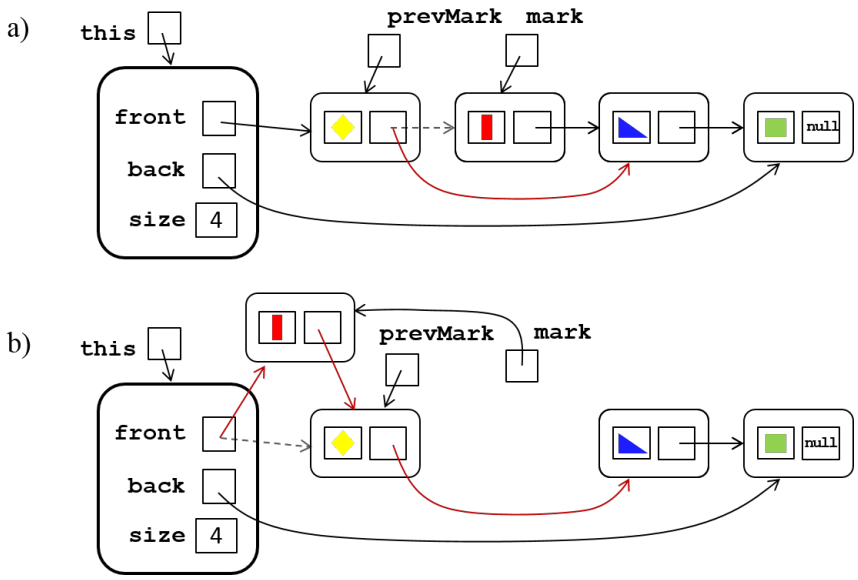


Figura 9: Canvis que realitza el mètode `toFront` sobre el grup de la figura 4 quan s'ha clicat el rectangle roig.

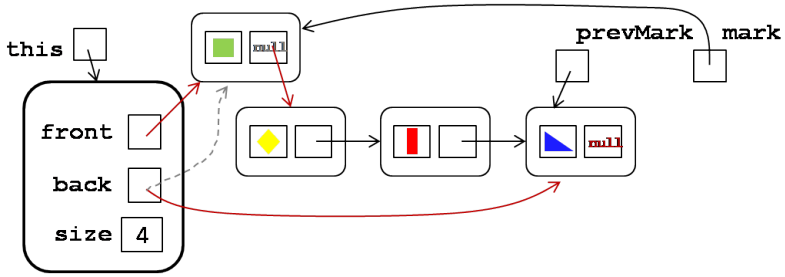


Figura 10: Canvis realitzats pel mètode `toFront` sobre el grup de la figura 4 quan s'ha clicat el polígon del fons (polígon que es troba en l'últim node de la seqüència).

- Mètode `public void toBack(Point p)`, que situa al fons del grup, sota de tot, el polígon seleccionat mitjançant el punt `p`. Si no hi ha cap polígon que continga a `p`, el mètode no fa res.

  En la implementació d'aquest mètode, si s'haguera trobat un node el polígon del qual continguera a `p`, i aquest node no estiguera ja en el fons, llavors cal resituar aquest node en la seqüència. A diferència del mètode anterior, en traure el node de la seqüència cal distingir si el node té o no un node que li precedeix (figures 11 a) i 12 a) respectivament). En qualsevol d'aquests casos, cal acabar posant el node al final de la seqüència (figures 11 b) i 12 b)).

  Amb el programa `Test5` es faran proves d'aquest mètode anàlogues a les realitzades per al mètode `toFront`.

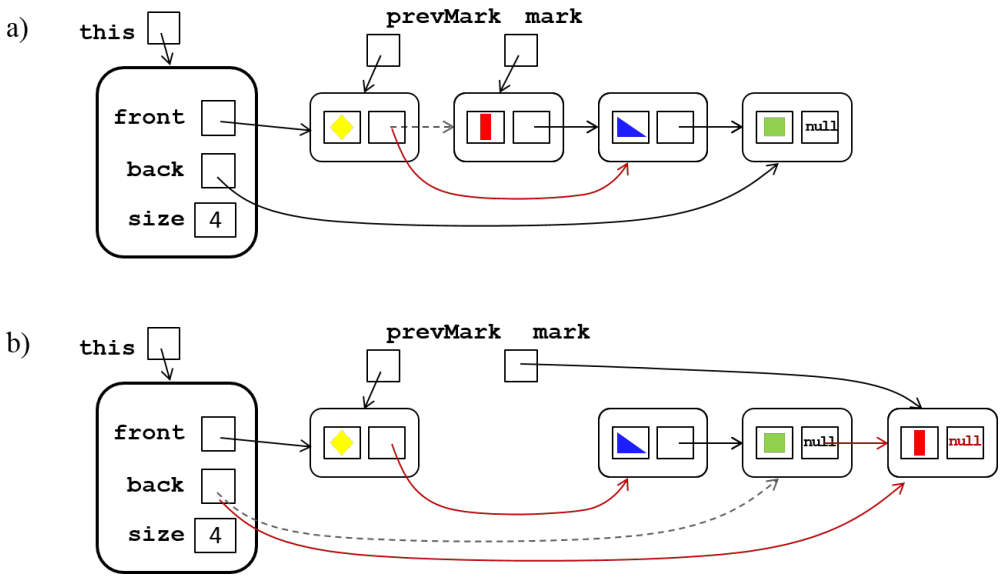


Figura 11: Canvis que realitza el mètode `toBack` sobre el grup de la figura 4 quan s'ha clicat el rectangle roig (polígon en un node que té un anterior).

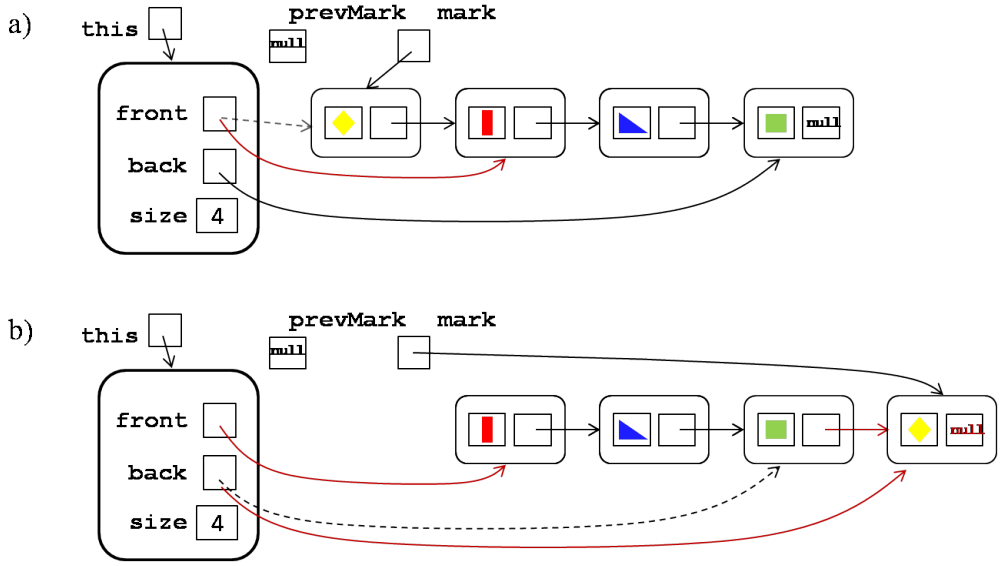


Figura 12: Canvis que realitza el mètode `toBack` sobre el grup de la figura 4 quan s'ha clicat el polígon del front (polígon que es troba en el primer node de la seqüència).

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 més enllà de les proves preliminars realitzades en les activitats anteriors.

En general, per a passar el test 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.

6.1 Activitat 7: validació de les classes `NodePol` i `PolygonGroup`

Descarregar els arxius corresponents a les *Unit Test* sobre el directori del paquet `pract5` i reobrir el teu projecte `prg` des de *BlueJ*.

Primer es farà la validació de la classe `NodePol` i, si estiguera correcta, es procedirà a validar `PolygonGroup`.

Triar 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. 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.

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*.