Remédiation 1er semestre 2023/2024

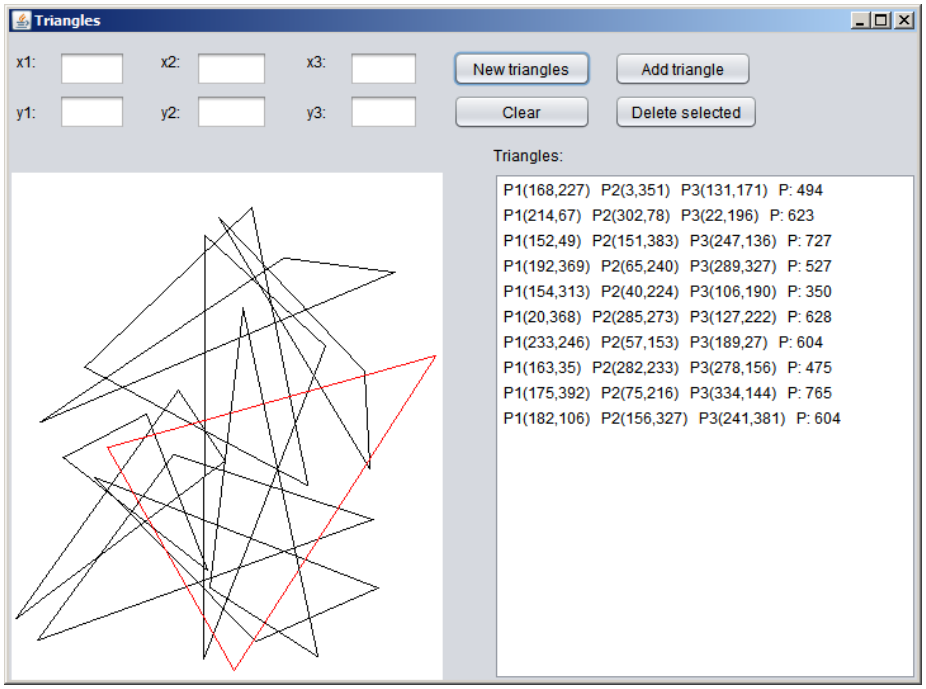
Les solutions sont à télécharger sous forme comprimée ZIP sur eduMoodle.

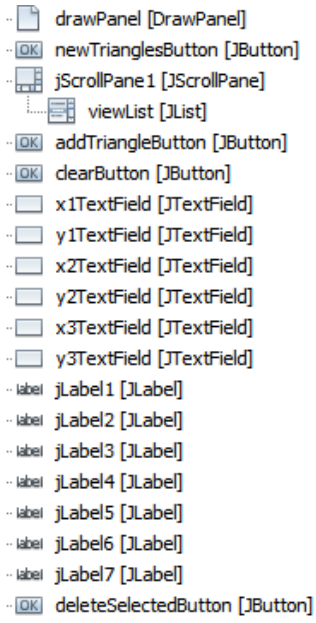
Révisez d'abord consciencieusement le cours et les exercices du cours !

Question 1 Traceur de triangles

Dans **NetBeans** créez un nouveau projet nommé **Remediation_I**, dans lequel vous réalisez l'exercice décrit ci-dessous.

Il s'agit de réaliser un petit programme pour dessiner des triangles sur une surface de dessin.





1. Réalisez la classe java **Triangle** suivant le schéma UML et les indications ci-dessous :
   - la classe sert à modéliser un triangle par les attributs **p1, p2** et **p3** de type *Point* qui désignent les 3 sommets du triangle.
   - la méthode privée **random** retourne un entier aléatoire de l'intervalle [pMin,pMax].
   - la méthode privée **getDistance** retourne la distance entre 2 points A et B. Faites un croquis sur une feuille avec 2 points $A(x_a, y_a)$ et $B(x_b, y_b)$. Pensez au théorème de Pythagore pour trouver la solution !
   - laissez **NetBeans** générer le **constructeur**.

- ajoutez un deuxième **constructeur** qui crée un triangle où les trois sommets sont distribués à des positions aléatoires sur la surface *pWidth* × *pHeight*. Profitez de la méthode privée **random** développée ci-dessus. Pensez à créer les 3 sommets avec l'opérateur **new**.
- la méthode **getPerimeter** retourne le périmètre du triangle. Profitez de la méthode **getDistance** développée ci-dessus.
- la méthode **draw** dessine le triangle colorié avec la couleur *pColor* sur le canevas *g*.
- la méthode **toString** retourne une description textuelle des sommets du triangle ainsi que son périmètre sous le format "P1(«x1», «y1») P2(«x2», «y2») P3(«x3», «y3») P: «périmètre»". La valeur du périmètre est arrondie à l'unité inférieure.
  Exemple : "`P1(182,106) P2(156,327) P3(241,381) P: 604`"

2. Réalisez la classe java **Triangles** suivant le schéma UML et les indications ci-dessous :
   - la classe sert à modéliser des triangles qui sont gérés dans une liste de type *ArrayList* nommée **alTriangles**.
   - le **constructeur** sert à initialiser la liste **alTriangles** avec *pN* triangles. Les coordonnées de chaque triangle sont limitées par la surface *pWidth* × *pHeight*.
   - la méthode **getIndexOfMaxPerimeter** retourne l'indice du triangle avec le plus grand périmètre. Si la liste est vide, la valeur -1 est retournée.
   - la méthode **draw** dessine tous les triangles de la liste **alTriangles** sur le canevas *g*. Le triangle avec le plus grand périmètre est dessiné en rouge, tous les autres en noir.
   - laissez **NetBeans** générer les méthodes **toArray**, **add**, **remove** et **clear**.

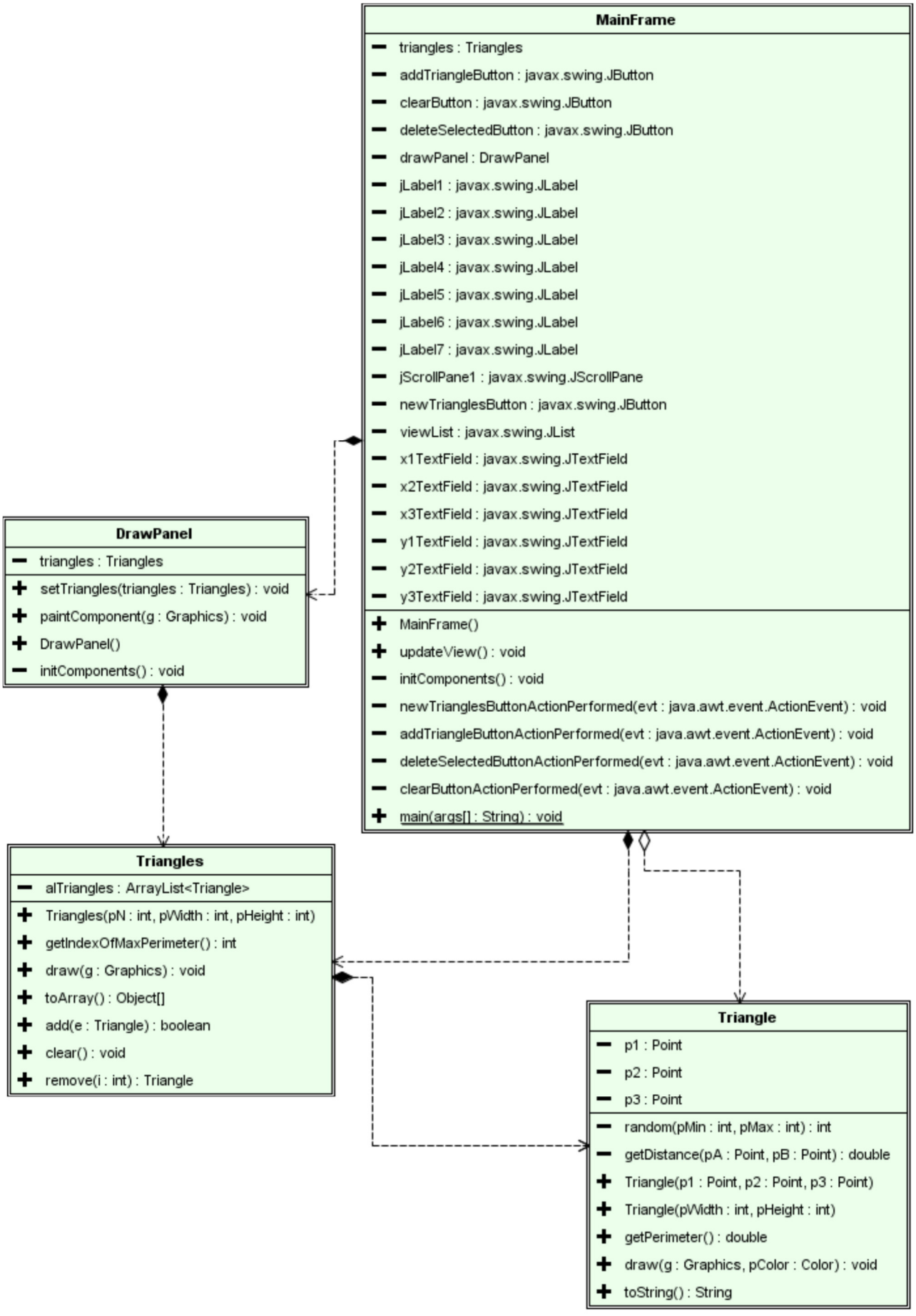
3. Réalisez la classe java **DrawPanel** suivant le schéma UML et les indications ci-dessous :
   - déclarez un attribut **triangles** du type **Triangles**.
   - réalisez le manipulateur **setTriangles**.
   - implémentez la méthode **paintComponent** qui dessine d'abord un rectangle blanc sur toute la surface du canevas et puis fait appel à **triangles.draw(…)** pour dessiner tous les triangles de la liste.

4. Réalisez la classe java **MainFrame** suivant le schéma UML et les indications ci-dessous :
   - réalisez l'interface graphique selon la figure de la première page.
   - déclarez un attribut **triangles** du type **Triangles**.
   - initialisez le titre de l'application à "Triangles" et désactivez les boutons **addTriangleButton** et **clearButton** dès le démarrage du programme.
   - réalisez la méthode **updateView** qui fait une mise à jour de la liste textuelle (**viewList**) et du dessin (**drawPanel**).
   - réalisez la fonctionnalité du bouton **newTrianglesButton** qui sert à initialiser l'attribut **triangles** par 10 nouveaux triangles. Les boutons **addTriangleButton** et **clearButton** sont activés.

- réalisez la méthode de réaction du bouton **addTriangleButton** qui sert à ajouter le triangle aux sommets spécifiés dans les boîtes texte de l'interface à la liste des triangles.
- réalisez la méthode de réaction du bouton **deleteSelectedButton** qui supprime le triangle sélectionné dans la liste **viewList**.
- réalisez la méthode de réaction du bouton **clearButton** qui vide la liste des triangles.
- N'oubliez pas de faire une mise à jour de la liste textuelle et du dessin chaque fois qu'il s'avère nécessaire !

Analyse et compréhension

Ajoutez la possibilité de cliquer avec le bouton gauche dans un triangle sur le canevas pour le sélectionner. Le triangle est alors dessiné en vert. Le bouton droit de la souris permet de désélectionner un triangle sélectionné. Quelles méthodes faut-il ajouter dans quelles classes ? Implémentez-les !


MainFrame
- triangles : Triangles
- addTriangleButton : javax.swing.JButton
- clearButton : javax.swing.JButton
- deleteSelectedButton : javax.swing.JButton
- drawPanel : DrawPanel
- jLabel1 : javax.swing.JLabel
- jLabel2 : javax.swing.JLabel
- jLabel3 : javax.swing.JLabel
- jLabel4 : javax.swing.JLabel
- jLabel5 : javax.swing.JLabel
- jLabel6 : javax.swing.JLabel
- jLabel7 : javax.swing.JLabel
- jScrollPane1 : javax.swing.JScrollPane
- newTrianglesButton : javax.swing.JButton
- viewList : javax.swing.JList
- x1TextField : javax.swing.JTextField
- x2TextField : javax.swing.JTextField
- x3TextField : javax.swing.JTextField
- y1TextField : javax.swing.JTextField
- y2TextField : javax.swing.JTextField
- y3TextField : javax.swing.JTextField
+ MainFrame()
+ updateView() : void
- initComponents() : void
- newTrianglesButtonActionPerformed(evt : java.awt.event.ActionEvent) : void
- addTriangleButtonActionPerformed(evt : java.awt.event.ActionEvent) : void
- deleteSelectedButtonActionPerformed(evt : java.awt.event.ActionEvent) : void
- clearButtonActionPerformed(evt : java.awt.event.ActionEvent) : void
+ main(args[] : String) : void
DrawPanel
- triangles : Triangles
+ setTriangles(triangles : Triangles) : void
+ paintComponent(g : Graphics) : void
+ DrawPanel()
- initComponents() : void
Triangles
- alTriangles : ArrayList<Triangle>
+ Triangles(pN : int, pWidth : int, pHeight : int)
+ getIndexOfMaxPerimeter() : int
+ draw(g : Graphics) : void
+ toArray() : Object[]
+ add(e : Triangle) : boolean
+ clear() : void
+ remove(i : int) : Triangle
Triangle
- p1 : Point
- p2 : Point
- p3 : Point
- random(pMin : int, pMax : int) : int
- getDistance(pA : Point, pB : Point) : double
+ Triangle(p1 : Point, p2 : Point, p3 : Point)
+ Triangle(pWidth : int, pHeight : int)
+ getPerimeter() : double
+ draw(g : Graphics, pColor : Color) : void
+ toString() : String