IUT2 Université Grenoble Alpes – Département INFO

R1.01 – TP4
Agrégation vs. Composition

L'objectif de ce TP est de travailler les principes d'agrégation et de composition des attributs de type classe

Avant de commencer...

- Lisez entièrement ce sujet
- Ouvrez un terminal et placez-vous dans votre répertoire `R1.01`
- Exécutez la commande : `cp -r /users/info/pub/1a/R1.01/TP4_Files .`
- Lancez **IJ** avec la commande `idea`, puis créez un nouveau projet `TP4`

1. Classe Point

1.1. Création de la classe `Point` (étudiée dans le cours 4, partie 2)

- ✓ Ouvrez avec un éditeur de texte, le fichier `Point.txt` qui a été copié dans votre répertoire `R1.01/TP4_Files` et copiez son contenu (CTRL + A *puis* CTRL + C)
- ✓ Dans le projet TP4, créez une classe java `Point`
- ✓ Placez le curseur à droite de la 1ère accolade, passez à la ligne et collez le contenu du presse-papiers (CTRL + V)

2. Classes d'objets représentant des triangles

Dans ce TP, un triangle sera défini par ses trois sommets

PROPRIETES DE TOUT TRIANGLE

- LONGUEUR D'UN COTE D'UN TRIANGLE CONNAISSANT LES COORDONNEES DE SES EXTREMITES

  La longueur du côté d'extrémités E_1 (x_1, y_1) et E_2 (x_2, y_2) s'obtient par la formule :

  $$longueur = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- SURFACE D'UN TRIANGLE, CONNAISSANT LA LONGUEUR DE SES TROIS COTES

  La surface S d'un triangle est calculable à partir des longueurs a, b et c de ses côtés, en appliquant la ***formule de Héron*** *:*

  $$S = \sqrt{p(p-a)(p-b)(p-c)} \text{ avec } p = (a+b+c)/2$$

A : Triangles dont les points sont définis par composition

2.1. Dans le projet TP4, ajoutez une classe `TriangleCompose`

2.2. Dans la classe `TriangleCompose` :

- ✓ Déclarez les **attributs** (les sommets du triangle) : A, B et C (type `Point`)
- ✓ Codez le **constructeur** d'un triangle dont les trois points sont hébergés **par composition**
- ✓ Codez les **méthodes** suivantes :
  - Consultation des trois attributs : `getA`, `getB` et `getC`
  - Longueur des côtés :

```
public double coteAB() {
    // résultat = longueur du côté d'extrémités A et B}
```

```
public double coteAC() {
    // résultat = longueur du côté d'extrémités A et C}
```

```
public double coteBC() {
    // résultat = longueur du côté d'extrémités B et C}
```

- Périmètre :

```
public double perimetre() {
    //{} => {résultat = périmètre du triangle}
```

- Surface :

```
public double surface() {
    //{} => {résultat = surface du triangle}
```

- Déplacement dans le plan :

```
public void deplacer(int dx, int dy) {
    // {} => {A, B et C ont été déplacés horizontalement de dx
    //         et verticalement de dy}
```

B : Triangles dont les points sont définis par agrégation

2.3. Dans le projet TP4, ajoutez une classe `TriangleAgrege`

2.4. Dans la classe `TriangleAgrege` :

- ✓ Déclarez les **attributs** (les sommets du triangle): A, B et C (type `Point`)
- ✓ Codez le **constructeur** d'un triangle dont les trois points sont hébergés **par agrégation**
- ✓ Copiez les **méthodes** que vous avez définies dans la classe `TriangleCompose` et collez-les

3. Classe Triangle_Utilitaire

3.1. Dans le projet TP4, ajoutez une classe `Triangle_Utilitaire`

3.2. Dans la classe `Triangle_Utilitaire` :

- ✓ Inspirez-vous de la fonction `arrondi2` proposée (et intégrée) dans le projet TP3_A pour coder la fonction :

```
public static double arrondi2(double unDouble) {
// {} => {résultat = valeur de unDouble après arrondi à 2 décimales}
```

- ✓ Ajoutez et codez les procédures suivantes :

```
public static void afficherTriangleCompose(TriangleCompose T) {
    //{} => {les coordonnées des trois sommets de T ont été affichées
    //       ainsi que les longueurs de ses côtés, son périmètre et
    //       sa surface après arrondi à 2 décimales}
```

```
public static void afficherTriangleAgrege(TriangleAgrege T) {
    //{} => {les coordonnées des trois sommets de T ont été affichées
    //       ainsi que les longueurs de ses côtés, son périmètre et
    //       sa surface après arrondi à 2 décimales}
```

4. Classe Triangle_Main

4.1. Dans le projet TP4, ajoutez une classe `Triangle_Main` et insérez-y une procédure `main`, où vous devrez :

- ✓ déclarer trois objets de type `Point` : A, B et C de coordonnées respectives (1, 1), (3, 5) et (5, 1)
- ✓ afficher les coordonnées de ces points

4.2. Exécutez et testez

4.3. Ajoutez dans la procédure `main` :

- ✓ la déclaration d'un objet `tComp` de type `TriangleCompose` ayant pour sommets les points A, B et C
- ✓ l'affichage des caractéristiques de `tComp` à l'aide de la procédure `afficherTriangleCompose` de la classe `Triangle_Utilitaire`
- ✓ la déclaration d'un objet `tAg` de type `TriangleAgrege` ayant pour sommets les points A, B et C
- ✓ l'affichage des caractéristiques de `tAg` à l'aide de la procédure `afficherTriangleAgrege` de la classe `Triangle_Utilitaire`

4.4. Exécutez et testez (cf. exemple de trace d'exécution en page suivante)

EXEMPLE DE TRACE D'EXECUTION (*la forme peut varier, mais pas les valeurs numériques*)

```
Points sur lesquels un triangle composé et un triangle agrégé sont construits :
A(1, 1) / B(3, 5) / C(5, 1)
* Triangle composé de sommets A, B, C :
  - Sommets : A (1, 1)  /  B (3, 5)  /  C (5, 1)
  - longueur de AB : 4.47
  - longueur de AC : 4.0
  - longueur de BC : 4.47
  - périmètre : 12.94
  - surface : 8.0
* Triangle agrégé de sommets A, B, C
  - Sommets : A (1, 1)  /  B (3, 5)  /  C (5, 1)
  - longueur de AB : 4.47
  - longueur de AC : 4.0
  - longueur de BC : 4.47
  - périmètre : 12.94
  - surface : 8.0
```

4.5. Dans la procédure `main` de la classe `Triangle_Main` :

- ✓ ajoutez la déclaration de deux entiers `dx` et `dy` et les instructions permettant de les initialiser par saisie
- ✓ Ajoutez les instructions permettant de déplacer les triangles `tComp` et `tAg`
  (*décalage horizontal* : `dx`, *décalage vertical* `dy`)

4.6. Exécutez et testez

ATTENTION…

* UN DEPLACEMENT NE DOIT PAS IMPACTER LA LONGUEUR DES COTES, DONC LE PERIMETRE ET LA SURFACE D'UN TRIANGLE !
* LE DEPLACEMENT DE `tComp` NE DOIT AVOIR AUCUN EFFET SUR LES COORDONNEES DES POINTS A PARTIR DESQUELS IL A ETE CONSTRUIT.
* LE DEPLACEMENT DE `tAg` IMPLIQUE LE DEPLACEMENT DES POINTS A PARTIR DESQUELS IL A ETE CONSTRUIT.

EXEMPLE DE TRACE D'EXECUTION (*la forme peut varier, mais pas les valeurs numériques*)

```
Déplacement des triangles de 3 horizontalement et -4 verticalement
-------------------------------------------------------------------------------------
Déplacement du triangle composé
Triangle composé après déplacement :
* Triangle composé :
  - Sommets : A (4, -3)  /  B (6, 1)  /  C (8, -3)
  - longueur de AB : 4.47
  - longueur de AC : 4.0
  - longueur de BC : 4.47
  - périmètre : 12.94
  - surface : 8.0
Points A, B et C après déplacement du triangle composé
A(1, 1) / B(3, 5) / C(5, 1)
-------------------------------------------
Déplacement du triangle agrégé
Triangle agrégé après déplacement :
* Triangle agrégé :
  - Sommets : A (4, -3)  /  B (6, 1)  /  C (8, -3)
  - longueur de AB : 4.47
  - longueur de AC : 4.0
  - longueur de BC : 4.47
  - périmètre : 12.94
  - surface : 8.0
Points A, B et C après déplacement du triangle agrégé
A(4, -3) / B(6, 1) / C(8, -3)
```

5. Triangles construits à partir de sommets saisis par l'utilisateur

CONSTRUCTION D'UN TRIANGLE A PARTIR DE TROIS POINTS

*On peut toujours construire un triangle à partir de trois points, **à condition que** les trois points soient distincts (ils diffèrent par au moins une de leurs coordonnées) et qu'ils ne soient pas alignés (il n'existe pas de droite passant par ces trois points)*

ALIGNEMENT DE TROIS POINTS (COLINEARITE)

Trois points P_1 (x_1, y_1), P_2 (x_2, y_2) et P_3 (x_3, y_3) sont alignés si et seulement si, ils ne sont pas confondus deux à deux et leurs coordonnées vérifient l'un des trois cas suivants :

Cas 1 : $x_1 = x_2$ et $x_1 = x_3$ (les trois points sont situés sur la même droite verticale)

Cas 2 : $y_1 = y_2$ et $y_1 = y_3$ (les trois points sont situés sur la même droite horizontale)

Cas 3 : $x_1 \neq x_2$ et $x_1 \neq x_3$ et $x_2 \neq x_3$ et $\left(\frac{y_2 - y_1}{x_2 - x_1} = \frac{y_3 - y_1}{x_3 - x_1}\right)$ (les trois points sont situés sur la même droite oblique)

5.1. Dans la classe `Triangle_Utilitaire`, ajoutez les fonctions suivantes :

a) Saisie d'un point

```
public static Point saisirPoint() {
    // {} => {résultat = un Point dont les
    //         coordonnées sont saisies par l'utilisateur}
```

b) Saisie d'un point non confondu avec un autre point

```
public static Point saisirPointNonConfondu(Point unPoint) {
    // {} => {résultat = un Point non confondu avec unPoint,
              dont les coordonnées sont saisies par l'utilisateur}
```

c) Saisie d'un point non aligné avec deux autres points

```
public static Point saisirPointNonAligne(Point P1, Point P2) {
    // {} => {résultat = un Point non aligné avec P1 et P2,
    //           dont les coordonnées sont saisies par l'utilisateur}
```

5.2. Créez une nouvelle classe `Triangle_Main2`

- ✓ Créez dans cette classe une procédure main
- ✓ Copiez dans la procédure main les instructions de la procédure main de la classe `Triangle_Main`
- ✓ Modifiez l'initialisation des points A, B et C :
  - chacun d'entre eux doit être initialisé par saisie
  - il doit être possible de construire un triangle avec les points qu'ils représentent

5.3. **Exécutez et testez**

6. Extensions…

6.1. Ajoutez à la classe `Triangle_Utilitaire` deux fonctions :

- ✓ la première retournant le triangle composé symétrique d'un triangle composé donné, par rapport à l'axe vertical
- ✓ la seconde retournant un triangle composé symétrique d'un triangle composé donné, par rapport à l'axe horizontal

6.2. Complétez la procédure `main` de la classe `Triangle_Main2` avec les instructions permettant de tester ces fonctions.

6.3. **Exécutez et testez**

6.4. Complétez la procédure `main` de la classe `Triangle_Main2` avec les instructions permettant de tester l'impact d'un déplacement (dx, dy) d'un triangle composé, sur les symétriques de ce triangle…

6.5. **Exécutez et testez**