TP1

Sujet à réaliser en binôme : Cercle

Préambule

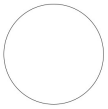
Ce sujet est à réaliser en binôme, mais chaque binôme a un sujet spécifique. En terme d'outillage, vous allez continuer d'utiliser VS Code avec les extensions Java tel que déjà vu dans l'UE PO du semestre précédent.

Nota Bene 1 *Afin de vous placer dans une situation proche de celle d'un développeur professionnel, les sujets de TP de GEN sont volontairement sous-spécifiés. C'est à vous de combler les trous de la manière la plus pertinente possible. Au début vous n'allez pas aimer, mais c'est une compétence à apprendre.*

Nota Bene 2 *Il est* **impératif** *de respecter le nommage de tous les éléments d'interface : cela sera testé automatiquement et toute erreur entrainera un 0.*

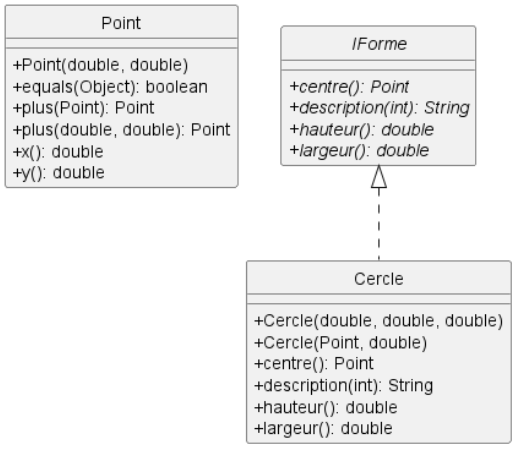
Objectif

L'objectif général des TP d'aujourd'hui est de réaliser dans VS Code un programme Java qui manipule des formes géométriques. Votre tâche particulière est de vous concentrer sur le concept de **Cercle**.



Un Cercle possède un centre et un rayon.

Question 1 *Commencez par créer un nouveau projet Java dans VS Code (option* No build tools*) que vous appelerez* `L2Gen`*. Y créer un dossier* `src` *et dans celui-ci un nouveau package* `fr.univrennes.istic.l2gen.geometrie`*. Dans ce package, créer une classe Point et une classe Cercle qui implémente l'interface IForme comme décrit ci-contre.*



La méthode `String description(int indentation)` prend en paramètre une indentation et doit retourner une description textuelle de Cercle préfixée de cette indentation (un "cran" d'indentation est constituée de 2 caractères blancs).

L'exécution du programme suivant :

```
IForme f = new Cercle(256, 256, 128);
System.out.println(f.description(1));
```

doit donc afficher exactement le texte :

```
 Cercle centre=256,256 r=128.0
```

Question 2 *Nous allons maintenant ajouter le concept de Groupe. Un Groupe contient des instances de IFormes, y compris d'autres Groupes, ce qui conduit à une structure d'arbre.*

Créer une classe Groupe qui implémente l'interface IForme telle que décrit ci-contre. Le constructeur de Groupe devra prendre comme argument `IForme ... formes`, ce qui permet de l'appeler avec un nombre variable d'arguments de type `IForme`, traités comme un tableau `IForme[]` (qui peut être de taille 0).

Comme indiqué sur le diagramme, ajouter dans IForme, Cercle et Groupe les méthodes `deplacer(double dx, double dy)` (déplacement relatif à la position courante), `dupliquer()` et `redimensionner(double px, double py)`.

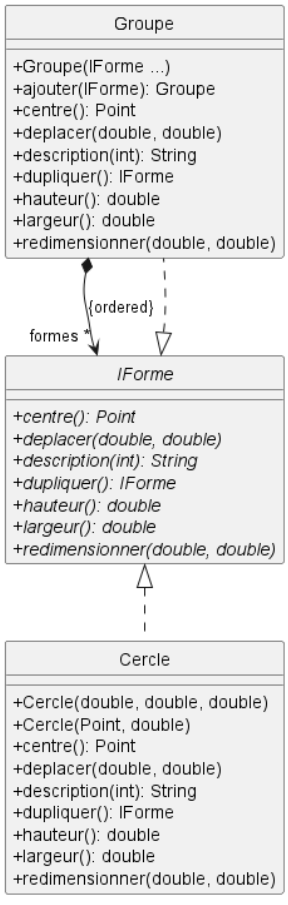
Construire un arbre dont les noeuds sont des instances de la classe Groupe et les feuilles des instances de la classe Cercle à l'aide du programme suivant :

```
Groupe arbre(IForme figure) {
        Groupe groupe = new Groupe(figure);
        IForme mini = figure.dupliquer();
        mini.redimensionner(0.5, 0.5);
        groupe.ajouter(mini);
        IForme minigroupe = groupe.dupliquer();
        minigroupe.redimensionner(0.25, 0.25);
        groupe.ajouter(minigroupe);
        return groupe;
}

Groupe arbre = arbre(new Cercle(256, 256, 128));
System.out.println(arbre.description(0));
```

L'exécution de ce programme doit afficher le texte suivant, reflétant la structure d'arbre grâce à l'indentation :

```
Groupe
  Cercle centre=256,256 r=128.0
  Cercle centre=256,256 r=64.0
  Groupe
    Cercle centre=256,256 r=32.0
    Cercle centre=256,256 r=16.0
```



Question 3 *Nous allons maitenant visualiser nos figures à l'aide de la technologie SVG, pour Scalable Vector Graphics (ou encore Graphismes vectoriels redimensionnables), qui est un langage s'appuyant sur le XML du W3C qui permet de définir des éléments graphiques avec des balises.*

Ce langage est directement supporté par des navigateurs Web comme Firefox. On en trouvera un petit tutoriel en français ici: https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial

Ajouter dans l'interface IForme la méthode `String enSVG()` qui a pour objectif de retourner une description SVG de la forme. Implémenter cette méthode pour vos classes Cercle et Groupe. L'exécution du programme :

```
IForme f = new Cercle(256, 256, 128);
System.out.println(f.enSVG());
```

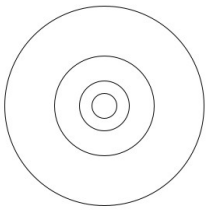
doit par exemple afficher le texte :

```
<circle cx="256.0" cy="256.0" r="128.0"
    fill="white" stroke="black"/>
```

Appeler `arbre.enSVG()` sur l'arbre défini à la question précédente pour obtenir :

```
<g>
<circle cx="256.0" cy="256.0" r="128.0"
    fill="white" stroke="black"/>
<circle cx="256.0" cy="256.0" r="64.0"
    fill="white" stroke="black"/>
<g>
<circle cx="256.0" cy="256.0" r="32.0"
    fill="white" stroke="black"/>
<circle cx="256.0" cy="256.0" r="16.0"
    fill="white" stroke="black"/>
</g>
</g>
```

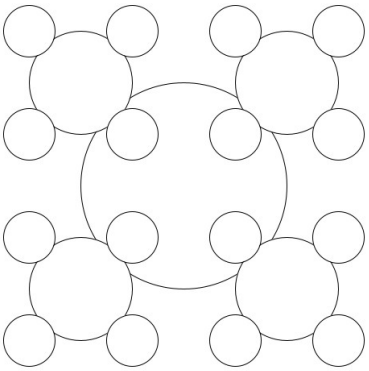
Sauvegarder le résultat dans un fichier Cercle.svg en le préfixant par `<svg xmlns="http://www.w3.org/2000/svg">` et en le terminant par `</svg>` pour obtenir un fichier visualisable dans votre navigateur préféré qui devrait ressembler au dessin ci-contre.

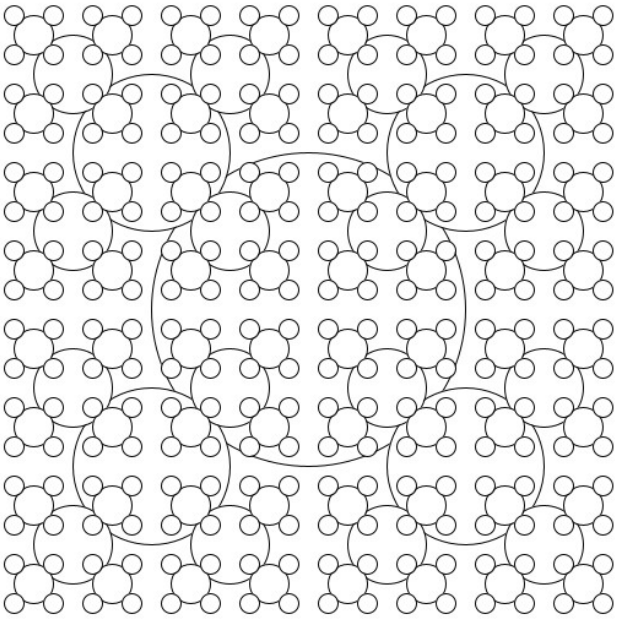


Question 4 Obligatoire pour le parcours défi, optionnelle pour les autres

Nous allons terminer ce premier TP en dessinant un objet de type fractal.

Ecrire une méthode récursive `IForme fractale(IForme base, int profondeur)` qui produit une répétition de la forme de base dupliquée 4 fois à une échelle 1/2 jusqu'à un certain niveau de profondeur. Par exemple pour votre Cercle de la question 1 on dessinera respectivement aux profondeur 2 et 4 :





TP1

Sujet à réaliser en binôme : Ligne

Préambule

Ce sujet est à réaliser en binôme, mais chaque binôme a un sujet spécifique. En terme d'outillage, vous allez continuer d'utiliser VS Code avec les extensions Java tel que déjà vu dans l'UE PO du semestre précédent.

Nota Bene 1 *Afin de vous placer dans une situation proche de celle d'un développeur professionnel, les sujets de TP de GEN sont volontairement sous-spécifiés. C'est à vous de combler les trous de la manière la plus pertinente possible. Au début vous n'allez pas aimer, mais c'est une compétence à apprendre.*

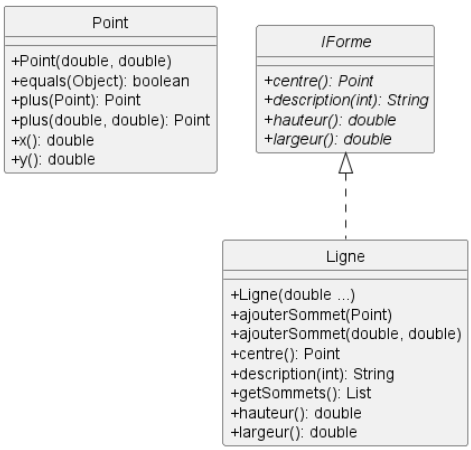
Nota Bene 2 *Il est* **impératif** *de respecter le nommage de tous les éléments d'interface : cela sera testé automatiquement et toute erreur entrainera un 0.*

Objectif

L'objectif général des TP d'aujourd'hui est de réaliser dans VS Code un programme Java qui manipule des formes géométriques. Votre tâche particulière est de vous concentrer sur le concept de **Ligne**.

Une Ligne (brisée) est constituée d'une liste de Points reliés en séquence.

Question 1 *Commencez par créer un nouveau projet Java dans VS Code (option* No build tools*) que vous appelerez* `L2Gen`*. Y créer un dossier* `src` *et dans celui-ci un nouveau package* `fr.univrennes.istic.l2gen.geometrie`*. Dans ce package, créer une classe Point et une classe Ligne qui implémente l'interface IForme comme décrit ci-contre.*



La méthode `String description(int indentation)` prend en paramètre une indentation et doit retourner une description textuelle de Ligne préfixée de cette indentation (un "cran" d'indentation est constituée de 2 caractères blancs).

L'exécution du programme suivant :

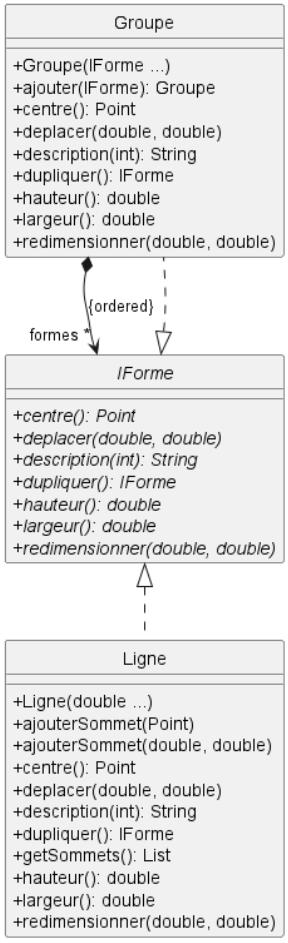
```
IForme f = new Ligne(128, 128, 128, 256, 256, 128, 256, 256);
System.out.println(f.description(1));
```

doit donc afficher exactement le texte :

```
 Ligne 128,128 128,256 256,128 256,256
```

Question 2 *Nous allons maintenant ajouter le concept de Groupe. Un Groupe contient des instances de IFormes, y compris d'autres Groupes, ce qui conduit à une structure d'arbre.*

Créer une classe Groupe qui implémente l'interface IForme telle que décrit ci-contre. Le constructeur de Groupe devra prendre comme argument `IForme ... formes`, ce qui permet de l'appeler avec un nombre variable d'arguments de type `IForme`, traités comme un tableau `IForme[]` (qui peut être de taille 0).



Comme indiqué sur le diagramme, ajouter dans IForme, Ligne et Groupe les méthodes `deplacer(double dx, double dy)` (déplacement relatif à la position courante), `dupliquer()` et `redimensionner(double px, double py)`.
Construire un arbre dont les noeuds sont des instances de la classe Groupe et les feuilles des instances de la classe Ligne à l'aide du programme suivant :

```
Groupe arbre(IForme figure) {
        Groupe groupe = new Groupe(figure);
        IForme mini = figure.dupliquer();
        mini.redimensionner(0.5, 0.5);
        groupe.ajouter(mini);
        IForme minigroupe = groupe.dupliquer();
        minigroupe.redimensionner(0.25, 0.25);
        groupe.ajouter(minigroupe);
        return groupe;
}

Groupe arbre = arbre(new Ligne(128, 128, 128, 256, 256, 128, 25
System.out.println(arbre.description(0));
```

L'exécution de ce programme doit afficher le texte suivant, reflétant la structure d'arbre grâce à l'indentation :

```
Groupe
  Ligne 128,128 128,256 256,128 256,256
  Ligne 160,160 160,224 224,160 224,224
  Groupe
    Ligne 176,176 176,208 208,176 208,208
    Ligne 184,184 184,200 200,184 200,200
```

Question 3 *Nous allons maitenant visualiser nos figures à l'aide de la technologie SVG, pour Scalable Vector Graphics (ou encore Graphismes vectoriels redimensionnables), qui est un langage s'appuyant sur le XML du W3C qui permet de définir des éléments graphiques avec des balises.*

Ce langage est directement supporté par des navigateurs Web comme Firefox. On en trouvera un petit tutoriel en français ici: https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial

Ajouter dans l'interface IForme la méthode `String enSVG()` qui a pour objectif de retourner une description SVG de la forme. Implémenter cette méthode pour vos classes Ligne et Groupe. L'exécution du programme :

```
IForme f = new Ligne(128, 128, 128, 256, 256, 128, 256, 256);
System.out.println(f.enSVG());
```

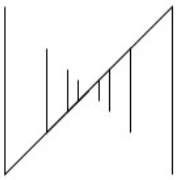
doit par exemple afficher le texte :

```
<polyline points="128␣128␣128␣256␣256␣128␣256␣256"
    fill="white" stroke="black"/>
```

Appeler `arbre.enSVG()` sur l'arbre défini à la question précédente pour obtenir :

```
<g>
<polyline points="128␣128␣128␣256␣256␣128␣256␣256"
    fill="white" stroke="black"/>
<polyline points="160␣160␣160␣224␣224␣160␣224␣224"
    fill="white" stroke="black"/>
<g>
<polyline points="176␣176␣176␣208␣208␣176␣208␣208"
    fill="white" stroke="black"/>
<polyline points="184␣184␣184␣200␣200␣184␣200␣200"
    fill="white" stroke="black"/>
</g>
</g>
```

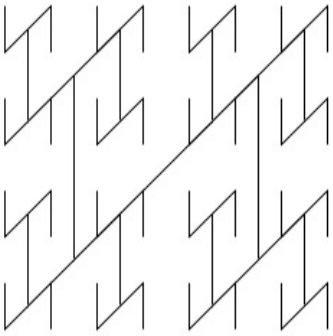
Sauvegarder le résultat dans un fichier Ligne.svg en le préfixant par `<svg xmlns="http://www.w3.org/2000/svg">` et en le terminant par `</svg>` pour obtenir un fichier visualisable dans votre navigateur préféré qui devrait ressembler au dessin ci-contre.

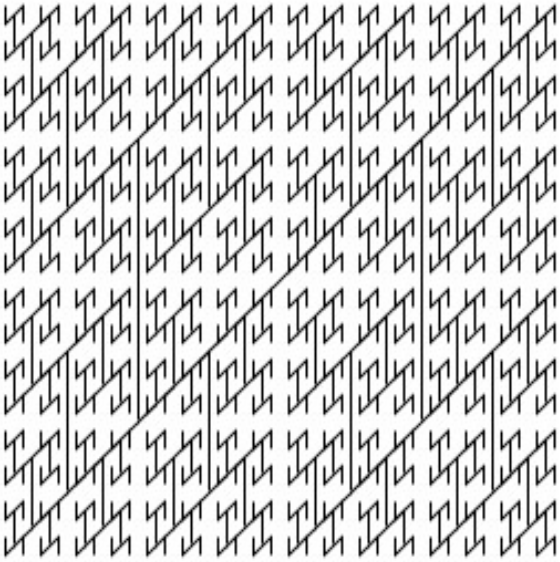


Question 4 Obligatoire pour le parcours défi, optionnelle pour les autres

Nous allons terminer ce premier TP en dessinant un objet de type fractal.

Ecrire une méthode récursive `IForme fractale(IForme base, int profondeur)` qui produit une répétition de la forme de base dupliquée 4 fois à une échelle 1/2 jusqu'à un certain niveau de profondeur. Par exemple pour votre Ligne de la question 1 on dessinera respectivement aux profondeur 2 et 4 :





TP1

Sujet à réaliser en binôme : Polygone

Préambule

Ce sujet est à réaliser en binôme, mais chaque binôme a un sujet spécifique. En terme d'outillage, vous allez continuer d'utiliser VS Code avec les extensions Java tel que déjà vu dans l'UE PO du semestre précédent.

Nota Bene 1 *Afin de vous placer dans une situation proche de celle d'un développeur professionnel, les sujets de TP de GEN sont volontairement sous-spécifiés. C'est à vous de combler les trous de la manière la plus pertinente possible. Au début vous n'allez pas aimer, mais c'est une compétence à apprendre.*

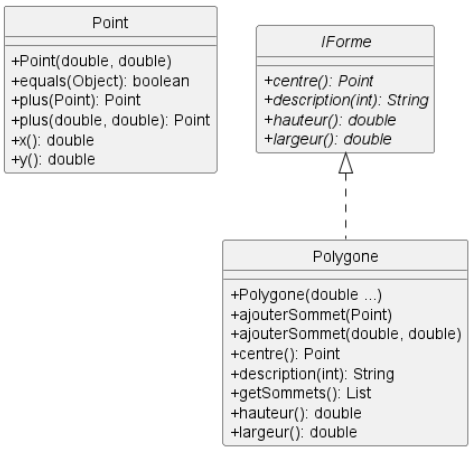
Nota Bene 2 *Il est* **impératif** *de respecter le nommage de tous les éléments d'interface : cela sera testé automatiquement et toute erreur entrainera un 0.*

Objectif

L'objectif général des TP d'aujourd'hui est de réaliser dans VS Code un programme Java qui manipule des formes géométriques. Votre tâche particulière est de vous concentrer sur le concept de **Polygone**.

Un Polygone est constitué d'une liste de Points représentant ses sommets.

Question 1 *Commencez par créer un nouveau projet Java dans VS Code (option* No build tools*) que vous appelerez* `L2Gen`*. Y créer un dossier* `src` *et dans celui-ci un nouveau package* `fr.univrennes.istic.l2gen.geometrie`*. Dans ce package, créer une classe Point et une classe Polygone qui implémente l'interface IForme comme décrit ci-contre.*



La méthode `String description(int indentation)` prend en paramètre une indentation et doit retourner une description textuelle de Polygone préfixée de cette indentation (un "cran" d'indentation est constituée de 2 caractères blancs).

L'exécution du programme suivant :

```
IForme f = new Polygone(128, 128, 128, 256, 256, 128, 256, 256);
System.out.println(f.description(1));
```

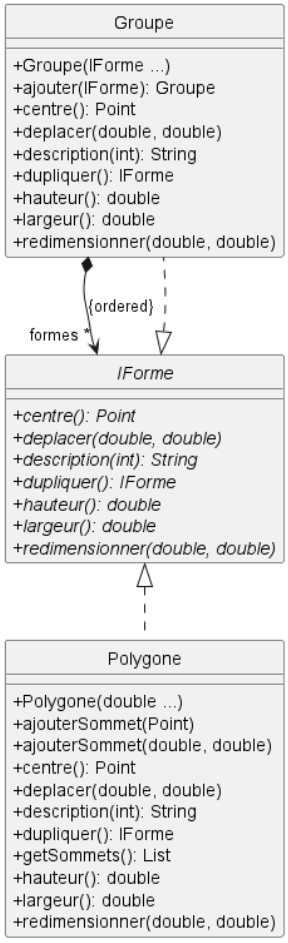
doit donc afficher exactement le texte :

```
 Polygone 128,128 128,256 256,128 256,256
```

Question 2 *Nous allons maintenant ajouter le concept de Groupe. Un Groupe contient des instances de IFormes, y compris d'autres Groupes, ce qui conduit à une structure d'arbre.*

Créer une classe Groupe qui implémente l'interface IForme telle que décrit ci-contre. Le constructeur de Groupe devra prendre comme argument `IForme ... formes`, ce qui permet de l'appeler avec un nombre variable d'arguments de type `IForme`, traités comme un tableau `IForme[]` (qui peut être de taille 0).

Comme indiqué sur le diagramme, ajouter dans IForme, Polygone et Groupe les méthodes `deplacer(double dx, double dy)` (déplacement relatif à la position courante), `dupliquer()` et `redimensionner(double px, double py)`.
Construire un arbre dont les noeuds sont des instances de la classe Groupe et les feuilles des instances de la classe Polygone à l'aide du programme suivant :




```
Groupe arbre(IForme figure) {
        Groupe groupe = new Groupe(figure);
        IForme mini = figure.dupliquer();
        mini.redimensionner(0.5, 0.5);
        groupe.ajouter(mini);
        IForme minigroupe = groupe.dupliquer();
        minigroupe.redimensionner(0.25, 0.25);
        groupe.ajouter(minigroupe);
        return groupe;
}

Groupe arbre = arbre(new Polygone(128, 128, 128, 256, 256, 128,
System.out.println(arbre.description(0));
```

L'exécution de ce programme doit afficher le texte suivant, reflétant la structure d'arbre grâce à l'indentation :

```
Groupe
  Polygone 128,128 128,256 256,128 256,256
  Polygone 160,160 160,224 224,160 224,224
  Groupe
    Polygone 176,176 176,208 208,176 208,208
    Polygone 184,184 184,200 200,184 200,200
```

Question 3 *Nous allons maitenant visualiser nos figures à l'aide de la technologie SVG, pour Scalable Vector Graphics (ou encore Graphismes vectoriels redimensionnables), qui est un langage s'appuyant sur le XML du W3C qui permet de définir des éléments graphiques avec des balises.*

Ce langage est directement supporté par des navigateurs Web comme Firefox. On en trouvera un petit tutoriel en français ici: https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial

Ajouter dans l'interface IForme la méthode `String enSVG()` qui a pour objectif de retourner une description SVG de la forme. Implémenter cette méthode pour vos classes Polygone et Groupe. L'exécution du programme :

```
IForme f = new Polygone(128, 128, 128, 256, 256, 128, 256, 256);
System.out.println(f.enSVG());
```

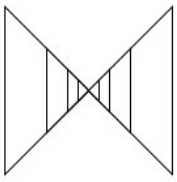
doit par exemple afficher le texte :

```
<polygon points="128␣128␣128␣256␣256␣128␣256␣256"
    fill="white" stroke="black"/>
```

Appeler `arbre.enSVG()` sur l'arbre défini à la question précédente pour obtenir :

```
<g>
<polygon points="128␣128␣128␣256␣256␣128␣256␣256"
    fill="white" stroke="black"/>
<polygon points="160␣160␣160␣224␣224␣160␣224␣224"
    fill="white" stroke="black"/>
<g>
<polygon points="176␣176␣176␣208␣208␣176␣208␣208"
    fill="white" stroke="black"/>
<polygon points="184␣184␣184␣200␣200␣184␣200␣200"
    fill="white" stroke="black"/>
</g>
</g>
```

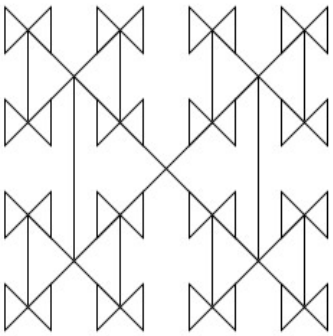
Sauvegarder le résultat dans un fichier Polygone.svg en le préfixant par `<svg xmlns="http://www.w3.org/2000/svg">` et en le terminant par `</svg>` pour obtenir un fichier visualisable dans votre navigateur préféré qui devrait ressembler au dessin ci-contre.

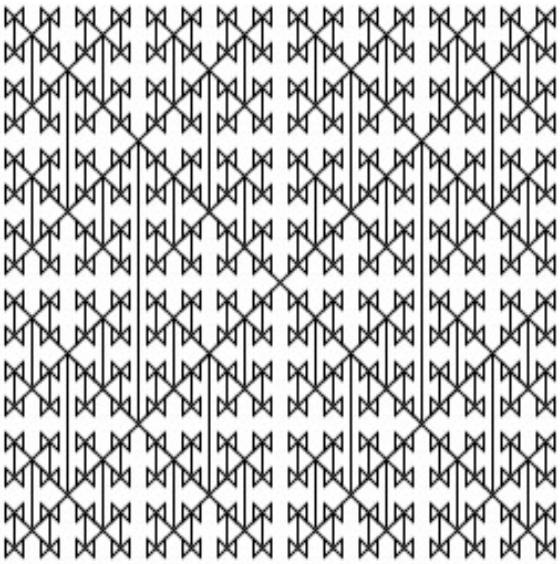


Question 4 Obligatoire pour le parcours défi, optionnelle pour les autres

Nous allons terminer ce premier TP en dessinant un objet de type fractal.

Ecrire une méthode récursive `IForme fractale(IForme base, int profondeur)` qui produit une répétition de la forme de base dupliquée 4 fois à une échelle 1/2 jusqu'à un certain niveau de profondeur. Par exemple pour votre Polygone de la question 1 on dessinera respectivement aux profondeur 2 et 4 :





TP1

Sujet à réaliser en binôme : Rectangle

Préambule

Ce sujet est à réaliser en binôme, mais chaque binôme a un sujet spécifique. En terme d'outillage, vous allez continuer d'utiliser VS Code avec les extensions Java tel que déjà vu dans l'UE PO du semestre précédent.

Nota Bene 1 *Afin de vous placer dans une situation proche de celle d'un développeur professionnel, les sujets de TP de GEN sont volontairement sous-spécifiés. C'est à vous de combler les trous de la manière la plus pertinente possible. Au début vous n'allez pas aimer, mais c'est une compétence à apprendre.*

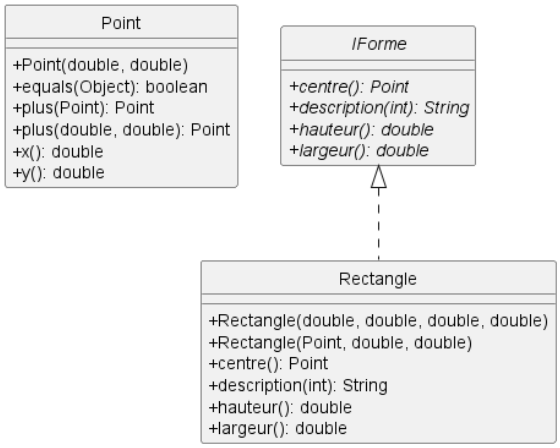
Nota Bene 2 *Il est* **impératif** *de respecter le nommage de tous les éléments d'interface : cela sera testé automatiquement et toute erreur entrainera un 0.*

Objectif

L'objectif général des TP d'aujourd'hui est de réaliser dans VS Code un programme Java qui manipule des formes géométriques. Votre tâche particulière est de vous concentrer sur le concept de **Rectangle**.

☐ Un Rectangle est caractérisé par un centre, une largeur et une hauteur.

Question 1 *Commencez par créer un nouveau projet Java dans VS Code (option* No build tools*) que vous appelerez* `L2Gen`*. Y créer un dossier* `src` *et dans celui-ci un nouveau package* `fr.univrennes.istic.l2gen.geometrie`*. Dans ce package, créer une classe Point et une classe Rectangle qui implémente l'interface IForme comme décrit ci-contre.*



La méthode `String description(int indentation)` prend en paramètre une indentation et doit retourner une description textuelle de Rectangle préfixée de cette indentation (un "cran" d'indentation est constituée de 2 caractères blancs).

L'exécution du programme suivant :

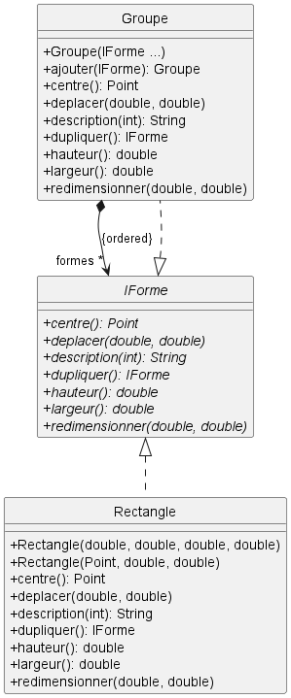
```
IForme f = new Rectangle(256, 256, 256, 128);
System.out.println(f.description(1));
```

doit donc afficher exactement le texte :

```
 Rectangle Centre=256,256 L=256.0 H=128.0
```

Question 2 *Nous allons maintenant ajouter le concept de Groupe. Un Groupe contient des instances de IFormes, y compris d'autres Groupes, ce qui conduit à une structure d'arbre.*

Créer une classe Groupe qui implémente l'interface IForme telle que décrit ci-contre. Le constructeur de Groupe devra prendre comme argument `IForme ... formes`, ce qui permet de l'appeler avec un nombre variable d'arguments de type `IForme`, traités comme un tableau `IForme[]` (qui peut être de taille 0).
Comme indiqué sur le diagramme, ajouter dans IForme, Rectangle et Groupe les méthodes `deplacer(double dx, double dy)` (déplacement relatif à la position courante), `dupliquer()` et `redimensionner(double px, double py)`.
Construire un arbre dont les noeuds sont des instances de la classe Groupe et les feuilles des instances de la classe Rectangle à l'aide du programme suivant :




```
Groupe arbre(IForme figure) {
        Groupe groupe = new Groupe(figure);
        IForme mini = figure.dupliquer();
        mini.redimensionner(0.5, 0.5);
        groupe.ajouter(mini);
        IForme minigroupe = groupe.dupliquer();
        minigroupe.redimensionner(0.25, 0.25);
        groupe.ajouter(minigroupe);
        return groupe;
}

Groupe arbre = arbre(new Rectangle(256, 256, 256, 128));
System.out.println(arbre.description(0));
```

L'exécution de ce programme doit afficher le texte suivant, reflétant la structure d'arbre grâce à l'indentation :

```
Groupe
  Rectangle Centre=256,256 L=256.0 H=128.0
  Rectangle Centre=256,256 L=128.0 H=64.0
  Groupe
    Rectangle Centre=256,256 L=64.0 H=32.0
    Rectangle Centre=256,256 L=32.0 H=16.0
```

Question 3 *Nous allons maitenant visualiser nos figures à l'aide de la technologie SVG, pour Scalable Vector Graphics (ou encore Graphismes vectoriels redimensionnables), qui est un langage s'appuyant sur le XML du W3C qui permet de définir des éléments graphiques avec des balises.*

Ce langage est directement supporté par des navigateurs Web comme Firefox. On en trouvera un petit tutoriel en français ici: https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial

Ajouter dans l'interface IForme la méthode `String enSVG()` qui a pour objectif de retourner une description SVG de la forme. Implémenter cette méthode pour vos classes Rectangle et Groupe. L'exécution du programme :

```
IForme f = new Rectangle(256, 256, 256, 128);
System.out.println(f.enSVG());
```

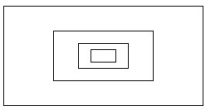
doit par exemple afficher le texte :

```
<rect x="128.0" y="192.0" width="256.0" height="128.0"
    fill="white" stroke="black"/>
```

Appeler `arbre.enSVG()` sur l'arbre défini à la question précédente pour obtenir :

```
<g>
<rect x="128.0" y="192.0" width="256.0" height="128.0"
    fill="white" stroke="black"/>
<rect x="192.0" y="224.0" width="128.0" height="64.0"
    fill="white" stroke="black"/>
<g>
<rect x="224.0" y="240.0" width="64.0" height="32.0"
    fill="white" stroke="black"/>
<rect x="240.0" y="248.0" width="32.0" height="16.0"
    fill="white" stroke="black"/>
</g>
</g>
```

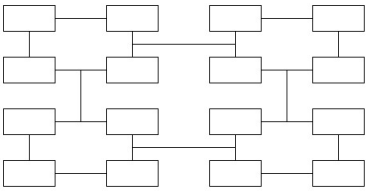
Sauvegarder le résultat dans un fichier Rectangle.svg en le préfixant par `<svg xmlns="http://www.w3.org/2000/svg">` et en le terminant par `</svg>` pour obtenir un fichier visualisable dans votre navigateur préféré qui devrait ressembler au dessin ci-contre.

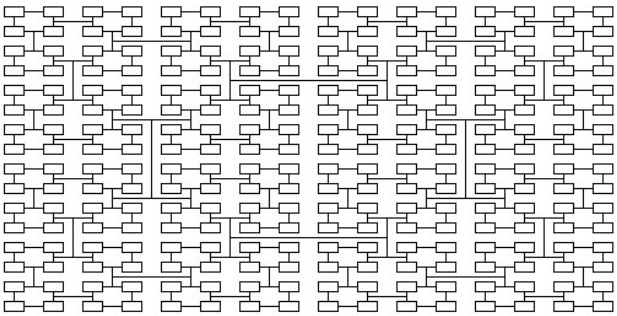


Question 4 Obligatoire pour le parcours défi, optionnelle pour les autres

Nous allons terminer ce premier TP en dessinant un objet de type fractal.

Ecrire une méthode récursive `IForme fractale(IForme base, int profondeur)` qui produit une répétition de la forme de base dupliquée 4 fois à une échelle 1/2 jusqu'à un certain niveau de profondeur. Par exemple pour votre Rectangle de la question 1 on dessinera respectivement aux profondeur 2 et 4 :





TP1

Sujet à réaliser en binôme : Triangle

Préambule

Ce sujet est à réaliser en binôme, mais chaque binôme a un sujet spécifique. En terme d'outillage, vous allez continuer d'utiliser VS Code avec les extensions Java tel que déjà vu dans l'UE PO du semestre précédent.

Nota Bene 1 *Afin de vous placer dans une situation proche de celle d'un développeur professionnel, les sujets de TP de GEN sont volontairement sous-spécifiés. C'est à vous de combler les trous de la manière la plus pertinente possible. Au début vous n'allez pas aimer, mais c'est une compétence à apprendre.*

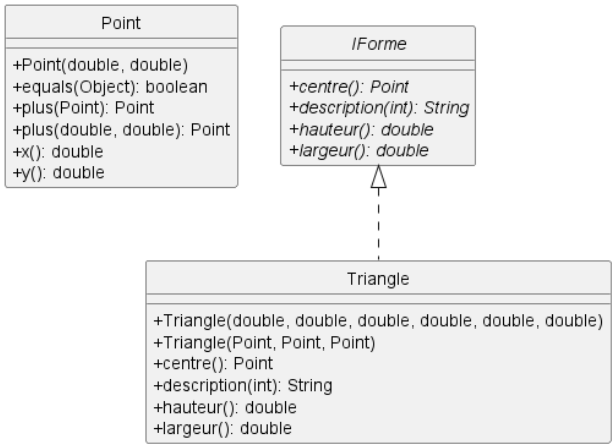
Nota Bene 2 *Il est* **impératif** *de respecter le nommage de tous les éléments d'interface : cela sera testé automatiquement et toute erreur entrainera un 0.*

Objectif

L'objectif général des TP d'aujourd'hui est de réaliser dans VS Code un programme Java qui manipule des formes géométriques. Votre tâche particulière est de vous concentrer sur le concept de **Triangle**.

Un Triangle (scalène) est caractérisé par ses 3 sommets.

Question 1 *Commencez par créer un nouveau projet Java dans VS Code (option* No build tools*) que vous appelerez* `L2Gen`*. Y créer un dossier* `src` *et dans celui-ci un nouveau package* `fr.univrennes.istic.l2gen.geometrie`*. Dans ce package, créer une classe Point et une classe Triangle qui implémente l'interface IForme comme décrit ci-contre.*



La méthode `String description(int indentation)` prend en paramètre une indentation et doit retourner une description textuelle de Triangle préfixée de cette indentation (un "cran" d'indentation est constituée de 2 caractères blancs).

L'exécution du programme suivant :

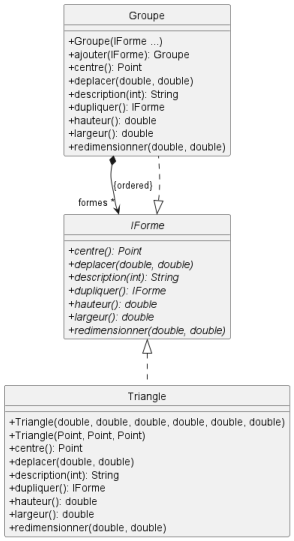
```
IForme f = new Triangle(192, 128, 256, 128, 256, 256);
System.out.println(f.description(1));
```

doit donc afficher exactement le texte :

```
 Triangle 192,128128,256256,256
```

Question 2 *Nous allons maintenant ajouter le concept de Groupe. Un Groupe contient des instances de IFormes, y compris d'autres Groupes, ce qui conduit à une structure d'arbre.*

Créer une classe Groupe qui implémente l'interface IForme telle que décrit ci-contre. Le constructeur de Groupe devra prendre comme argument `IForme ... formes`, ce qui permet de l'appeler avec un nombre variable d'arguments de type `IForme`, traités comme un tableau `IForme[]` (qui peut être de taille 0).
Comme indiqué sur le diagramme, ajouter dans IForme, Triangle et Groupe les méthodes `deplacer(double dx, double dy)` (déplacement relatif à la position courante), `dupliquer()` et `redimensionner(double px, double py)`.
Construire un arbre dont les noeuds sont des instances de la classe Groupe et les feuilles des instances de la classe Triangle à l'aide du programme suivant :




```
Groupe arbre(IForme figure) {
        Groupe groupe = new Groupe(figure);
        IForme mini = figure.dupliquer();
        mini.redimensionner(0.5, 0.5);
        groupe.ajouter(mini);
        IForme minigroupe = groupe.dupliquer();
        minigroupe.redimensionner(0.25, 0.25);
        groupe.ajouter(minigroupe);
        return groupe;
}

Groupe arbre = arbre(new Triangle(192, 128, 256, 128, 256, 256));
System.out.println(arbre.description(0));
```

L'exécution de ce programme doit afficher le texte suivant, reflétant la structure d'arbre grâce à l'indentation :

```
Groupe
  Triangle 192,128128,256256,256
  Triangle 192,171160,235224,235
  Groupe
    Triangle 192,192176,224208,224
    Triangle 192,203184,219200,219
```

Question 3 *Nous allons maitenant visualiser nos figures à l'aide de la technologie SVG, pour Scalable Vector Graphics (ou encore Graphismes vectoriels redimensionnables), qui est un langage s'appuyant sur le XML du W3C qui permet de définir des éléments graphiques avec des balises.*

Ce langage est directement supporté par des navigateurs Web comme Firefox. On en trouvera un petit tutoriel en français ici: https://developer.mozilla.org/fr/docs/Web/SVG/Tutorial

Ajouter dans l'interface IForme la méthode `String enSVG()` qui a pour objectif de retourner une description SVG de la forme. Implémenter cette méthode pour vos classes Triangle et Groupe. L'exécution du programme :

```
IForme f = new Triangle(192, 128, 256, 128, 256, 256);
System.out.println(f.enSVG());
```

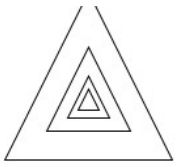
doit par exemple afficher le texte :

```
<polygon points="192␣128␣128␣256␣256␣256"
    fill="white" stroke="black"/>
```

Appeler `arbre.enSVG()` sur l'arbre défini à la question précédente pour obtenir :

```
<g>
<polygon points="192␣128␣128␣256␣256␣256"
    fill="white" stroke="black"/>
<polygon points="192␣171␣160␣235␣224␣235"
    fill="white" stroke="black"/>
<g>
<polygon points="192␣192␣176␣224␣208␣224"
    fill="white" stroke="black"/>
<polygon points="192␣203␣184␣219␣200␣219"
    fill="white" stroke="black"/>
</g>
</g>
```

Sauvegarder le résultat dans un fichier Triangle.svg en le préfixant par `<svg xmlns="http://www.w3.org/2000/svg">` et en le terminant par `</svg>` pour obtenir un fichier visualisable dans votre navigateur préféré qui devrait ressembler au dessin ci-contre.



Question 4 Obligatoire pour le parcours défi, optionnelle pour les autres

Nous allons terminer ce premier TP en dessinant un objet de type fractal.

Ecrire une méthode récursive `IForme fractale(IForme base, int profondeur)` qui produit une répétition de la forme de base dupliquée 4 fois à une échelle 1/2 jusqu'à un certain niveau de profondeur. Par exemple pour votre Triangle de la question 1 on dessinera respectivement aux profondeur 2 et 4 :