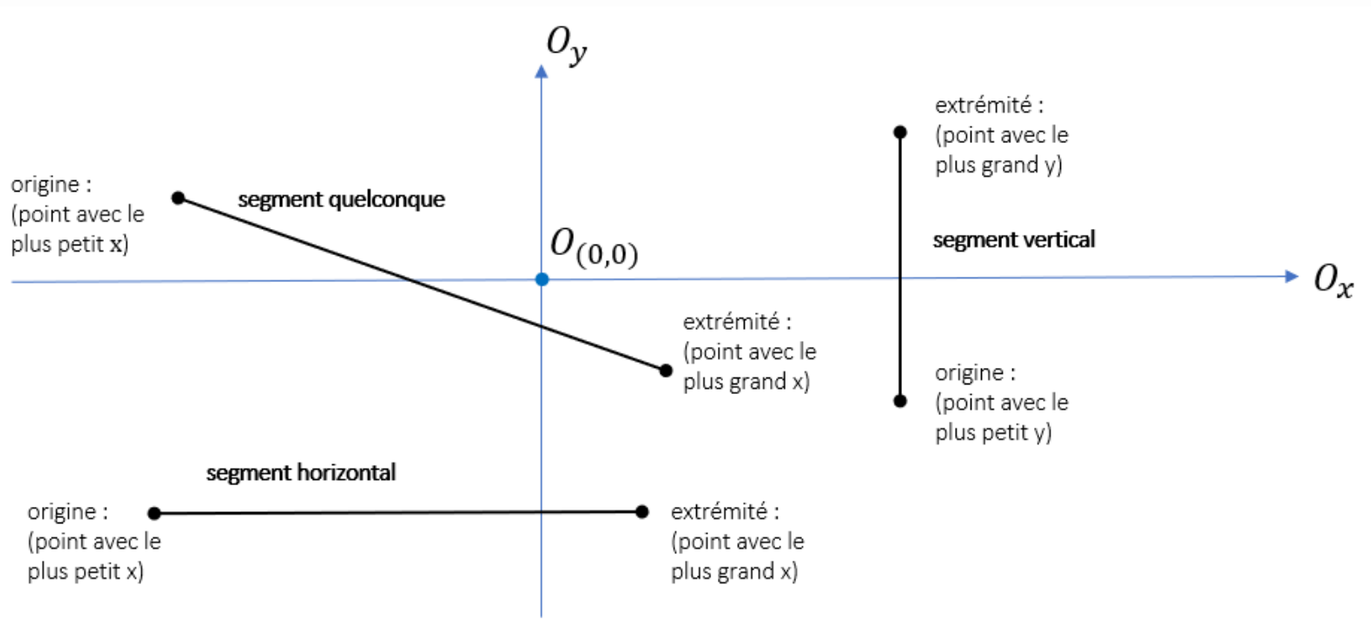
L3 MIAGE - APO : TD - Segments de Droites - Encapsulation

Dernière mise à jour : 20/10/2025, 12:32:19 par Philippe.Genoud.

Un projet Java pour proposer une librairie de formes géométriques simple est en cours de réalisation, l'étudiant stagiaire en ayant la charge etant parti on vous demande de reprendre son développement.

Pour le moment deux classes ont été spécifiées :

1. Une classe $Point$ qui représente un point dans le plan (voir la javadoc). Cette classe a été complétement implémentée et dont vous pouvez accéder au code source ICI.
2. Une classe $SegmentDeDroite$ qui représente un segment dans le plan et dont la spécification est la suivante :
    - Un segment de droite est défini par deux points (ses extrémités) qui ne peuvent être confondus.
    - On appelle $origine$ du segment, celui de ces deux points qui a l'abscisse x la plus petite et si le segment est vertical (abscisses identiques) celui qui a l'ordonnée y la plus petite.
    - On appelle $extremité$ du segment, celui de ces deux points qui a l'abscisse x la plus grande et si le segment est vertical (abscisses identiques) celui qui a l'ordonnée y la plus grande.



origine et extrémité d'un segment.

    - Les constructeurs proposés par la classe $SegmentDeDroite$ permettent de définir un segment en spécifiant ses extrémités
        - soit par leurs coordonnées $(x1, y1)$ et $(x2, y2)$,
        - soit par deux Points $p1$ et $p2$.
        - l'ordre dans lequels ces points sont passés au constructeur n'a pas d'importance. Les instructions $new\ SegmentDeDroite(x1, y1, x2, y2)$, $new\ SegmentDeDroite(x2, y2, x1, y1)$, $new\ SegmentDeDroite(p1, p2)$ ou $new\ SegmentDeDroite(p2, p1)$ devront toutes produire le même objet $SegmentDeDroite$.
        - ces constructeurs doivent garantir le fait que le segment créé n'est pas "dégénéré", c'est à dire que ses points extrémités ne sont pas confondus. Dans le cas contraire l'objet n'est pas créé et une exception de type **java.lang.IllegalArgumentException** doit être lancée.
    - Les méthodes de cette classe sont :
        - $getOrigine()$ qui retourne un point correspondant à ***l'origine*** du segment.
        - $getExtremite()$ qui retourne un point correspondant à ***l'extremité*** du segment.
        - $longueur()$ qui calcule et retourne la longueur du segment.
        - $translater(double\ dx,\ double\ dy)$ qui permet de translater le segment dans le plan selon un vecteur (dx, dy).
        - $estVertical()$ et $estHorizontal()$ qui permettent de tester si le segment est respectivment vertical ou horizontal.
        - $toString()$ qui renvoie une représentation textuelle (chaîne de caractères) du segment, plus précisément les coordonnées des deux points le définissant, les coordonnées de son ***origine*** en premier, celles de son ***extremité*** en second. Par exemple pour un segment défini par les points $P1(12, 24)$ et $P2(32, -7)$, la chaîne retournée par cette méthode sera
          `SegmentDeDroite[(12.0,24.0);(32.0,-7.0)]`
        - $egale(SegmentDeDroite\ s)$ qui permet de tester l'égalité du segment avec un autre segment.

Question 1 :

Quels sont les attributs que définit la classe $SegmentDeDroite$ pour modéliser un segment de droite ?

Question 2:

En respectant les spécifications données précédemment, écrivez le code Java de la classe $SegmentDeDroite$.

Procédez de manière incrémentale :

1. écrivez tout d'abord un constructeur et les méthodes $getOrigine()$ et $getExtremite()$ et écrivez un petit programme de test pour vérifier qu'elle sont correctes en jouant les tests (un certain nombre de cas de test devraient être réussis),
2. ecrivez ensuite le deuxième constructeur et vérifiez qu'il est conforme (des tests supplémentaires doivent être écrits),
3. Est-ce que la manière dont vous avez écrit les constructeurs et les méthodes $getOrigine$ et $getExtremité$ garantit que le segment n'est **jamais** dégénéré ? Quelles solutions peut-on envisager pour cela ?

Question 3:

Modifiez la classe $Point$ pour la rendre immuable et si nécessaire la classe $SegmentDeDroite$ pour prendre en compte ces modifications.

Question 4:

Implémentez et testez les autres méthodes de la classe $SegmentDeDroite$