

Code branche <b>PROGR</b>	Ministère de l'Éducation nationale, de l'Enfance et de la Jeunesse <b>EXAMEN DE FIN D'ÉTUDES SECONDAIRES TECHNIQUES</b> Régime technique – Session 2015	
Épreuve écrite	Branche	Division / Section
Durée de l'épreuve 4 heures	<b>Programmation</b>	<b>GI</b>
Date de l'épreuve <i>20 mai 2015</i>		<b>Division technique générale</b> <b>Section informatique</b>

**Renommez d'abord le dossier LTAM\_GI\_xx en remplaçant xx par votre numéro d'examen.**

### Question 1 [60 points]

Dans la suite, vous allez développer un petit jeu, qui consiste à faire 'exploder' un rectangle dans plusieurs rectangles plus petits, puis à le recoller à l'aide de la souris.

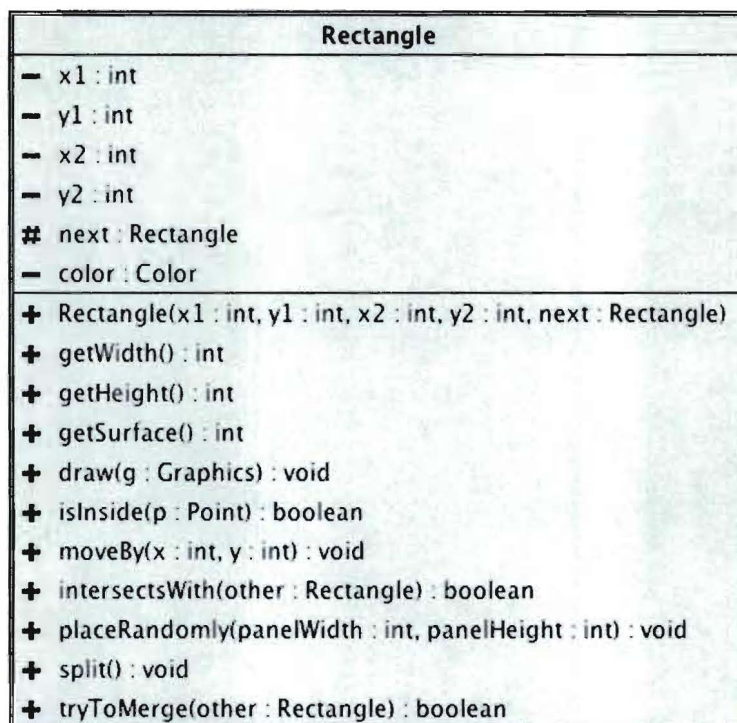
Essayez le programme de démonstration **Recploder.jar** qui vous est fourni dans votre dossier !

Une petite vidéo de démonstration **Recploder-demo.mov** vous est fournie dans le même dossier.

Développez une copie exacte de ce programme en suivant les précisions données dans la suite.

**Classe Rectangle [1 + 1 + 1 + 1 + 0,5 + 3 + 2,5 + 3,5 + 5 = 18,5 P]**

La classe **Rectangle** correspond au diagramme UML suivant :



1. Un rectangle est défini par les coordonnées du point supérieur gauche et du point inférieur droit (on a donc toujours :  $x1 \leq x2$  et  $y1 \leq y2$ ).

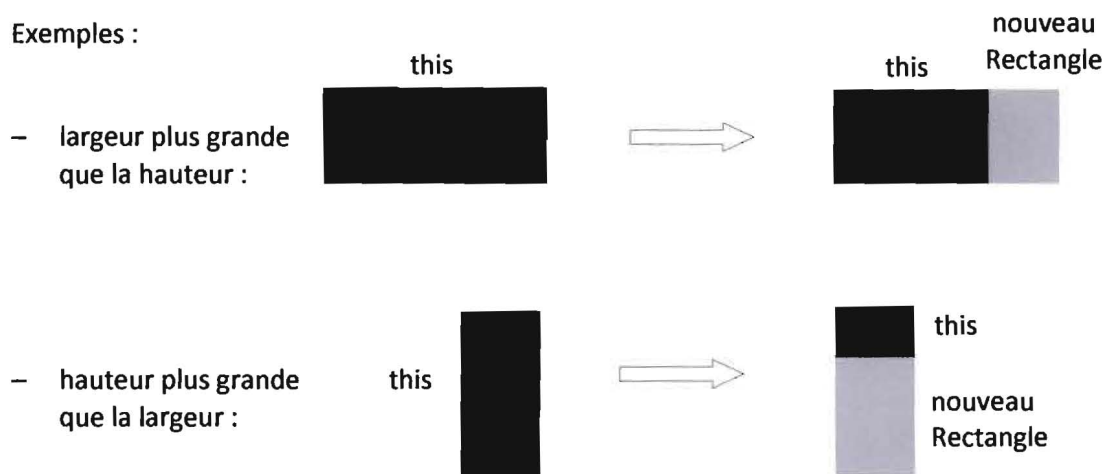
Les rectangles sont organisés dans une liste chaînée. L'attribut **next** pointe vers le prochain rectangle de la liste.

Lors de la construction, chaque rectangle est initialisé à une couleur aléatoire (qui ne changera jamais après la création du rectangle).

2. Les méthodes **getWidth**, **getHeight** et **getSurface** calculent et retournent la largeur, la hauteur et la surface du rectangle.
3. La méthode **draw** dessine le rectangle sur le canevas donné comme paramètre.
4. La méthode **isInside** retourne **true** si et seulement si le point **p** donné comme paramètre se trouve à l'intérieur du rectangle.
5. La méthode **moveBy** déplace le rectangle de **x** points en horizontale et de **y** points en verticale.
6. La méthode **intersectsWith** retourne **true** si et seulement si le rectangle actuel et le rectangle **other** donné comme paramètre se recoupent. Autrement dit, la méthode retourne **true** si et seulement si l'intersection des deux rectangles n'est pas vide. La méthode retourne **false** s'il s'agit deux fois du même rectangle (c.-à-d. si l'adresse des deux objets est la même).  
Méthode : calculez les coordonnées du rectangle formant l'intersection des deux rectangles.
7. La méthode **placeRandomly** place le rectangle à une position aléatoire du panneau de dessin. Le rectangle doit être placé de façon à ce qu'il se trouve entièrement à l'intérieur de la surface de dessin. Les dimensions du panneau de dessin sont fournies comme paramètres.
8. La méthode **split** coupe le rectangle en deux et ajoute le deuxième rectangle à la liste. La surface totale des deux rectangles doit être égale à la surface du rectangle original. La coupure est effectuée de façon à ce que la plus grande des deux dimensions soit diminuée. La position de la coupure est aléatoire, mais elle est effectuée à au moins 5 pixels du bord (c.-à-d. on ne crée pas de rectangles d'une hauteur ou largeur de moins de 5 pixels.).

Si p.ex. le rectangle est plus large qu'il est haut, il est coupé le long de la hauteur. Dans ce cas, la hauteur des deux rectangles reste identique tandis que la largeur change.

Exemples :



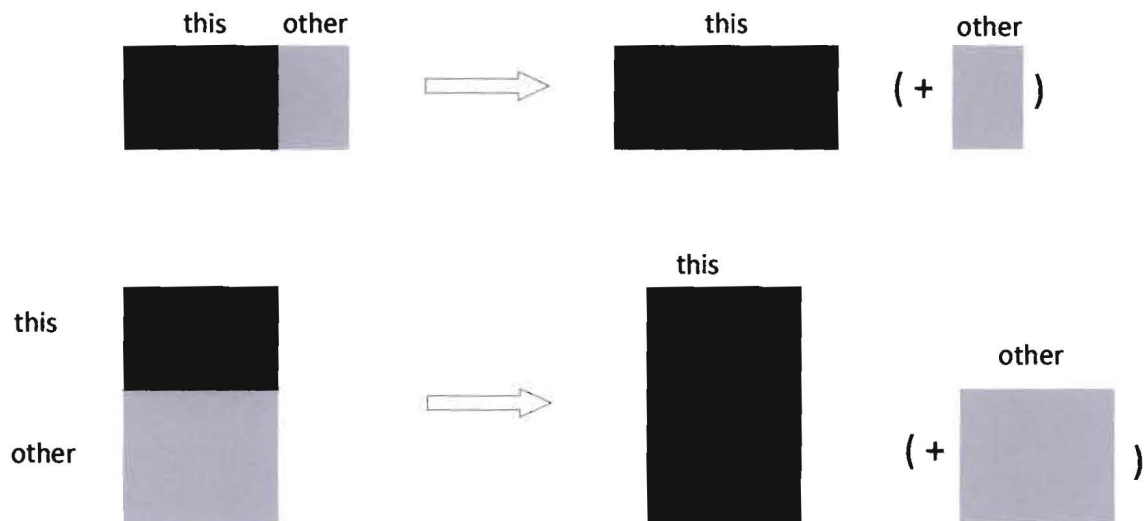
9. La méthode **tryToMerge** fusionne le rectangle actuel avec rectangle **other** donné comme paramètre. C.-à-d. après la fusion, la surface du rectangle actuel correspond à la somme de la surface des deux rectangles originaux. La fusion est effectuée uniquement si les deux rectangles se touchent de l'extérieur avec l'un de leurs bords et que les bords qui se touchent concordent exactement (même longueur, même coordonnées). Il est uniquement permis de fusionner des rectangles par les bords opposés, c.-à-d. bord gauche avec bord droit ou bord inférieur avec bord supérieur. (→ *Faites des essais avec le programme de démonstration !*)

Uniquement la surface du rectangle actuel change. La dimension (hauteur ou largeur) par laquelle les deux rectangles se touchent reste inchangée.

Le rectangle **other** reste inchangé et il reste dans la liste (il sera supprimé dans la méthode **tryToMergeSelected** de la classe **Rectangles**).

La méthode **tryToMerge** retourne **true**, si et seulement si les deux rectangles ont été fusionnés avec succès.

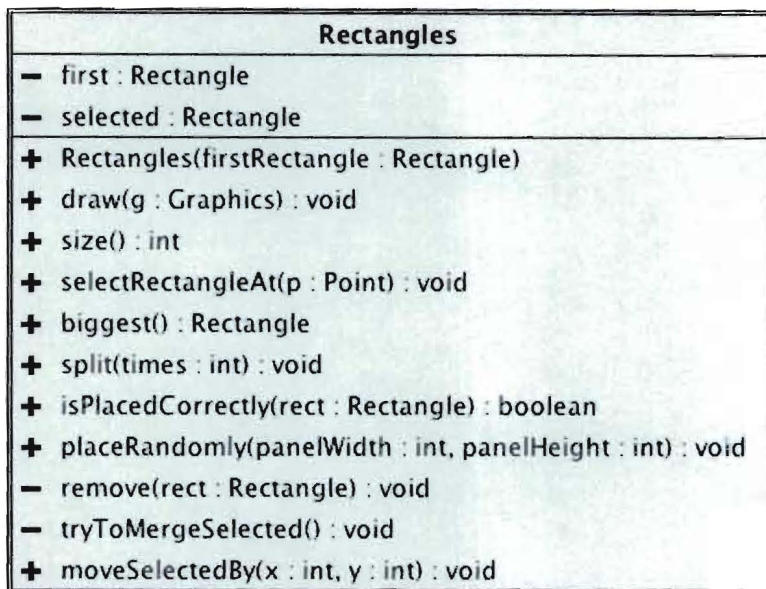
Exemples :





## Classe Rectangles [0,5 + 2,5 + 2,5 + 4 + 4 + 1 + 3 + 3 + 5 + 4 + 1 = 30,5 P]

La classe **Rectangles** correspond au diagramme UML suivant :



### Attention :

Lors de la réalisation de la classe **Rectangles**, il est nécessaire d'ajouter des méthodes supplémentaires pour réaliser les méthodes récursives. Ces méthodes supplémentaires ne figurent pas dans ce diagramme UML !

La liste chaînée est à développer par vos soins sans recourir aux types prédéfinis.

La classe **Rectangles** sert à effectuer des opérations sur des rectangles organisés dans une liste chaînée.

[Pour les questions où une méthode récursive est demandée, une solution itérative peut être évaluée au maximum à la moitié des points attribués à cette question.]

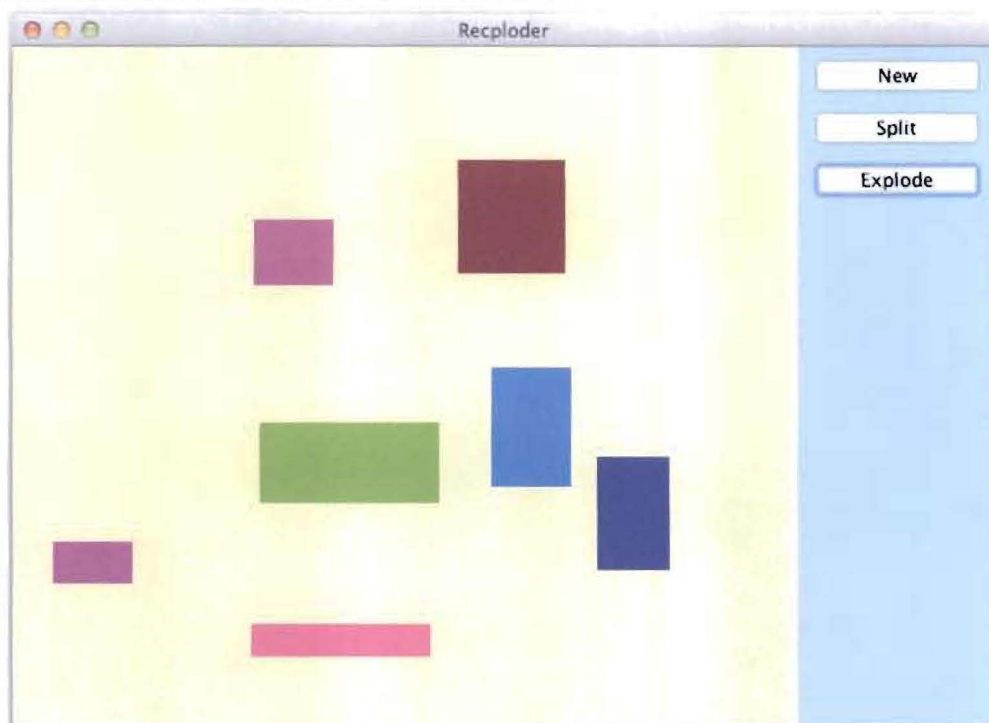
1. **first** pointe sur le premier rectangle de la liste. **first** est **null** si la liste est vide.  
**selected** pointe sur le rectangle actuellement sélectionné. (Au cours du jeu, le rectangle sélectionné est celui qui est en train d'être déplacé à l'aide de la souris). **selected** est **null** si aucun rectangle n'est sélectionné).  
Lors de la création un rectangle initial fourni comme paramètre est placé au début de la liste.
2. La méthode **draw** dessine tous les rectangles de la liste. La méthode **draw** doit être réalisée de façon récursive.
3. La méthode **size** calcule et retourne comme résultat le nombre de rectangles qui se trouvent actuellement dans la liste. La méthode **size** doit être réalisée de façon récursive.
4. La méthode **selectRectangleAt** sélectionne le dernier rectangle de la liste qui se trouve à la position du point **p** donné comme paramètre. Si aucun rectangle ne se trouve aux coordonnées données, **selected** sera **null**. La méthode **selectRectangleAt** doit être réalisée de façon récursive.
5. La méthode **biggest** retourne comme résultat le rectangle avec la plus grande surface. La méthode **biggest** doit être réalisée de façon récursive.
6. Autant de fois que l'indique le paramètre **times**, la méthode **split** recherche le plus grand rectangle de la liste et le coupe en deux.
7. La méthode **isPlacedCorrectly** retourne **true**, si et seulement si le rectangle fourni comme paramètre ne se recoupe avec aucun autre des rectangles dans la liste. La méthode **isPlacedCorrectly** doit être réalisée de façon récursive.
8. La méthode **placeRandomly** place tous les rectangles à une position aléatoire du panneau de dessin mais de façon à ce que tous les rectangles soient séparés les uns des autres (c.-à.-d. aucun rectangle ne se recoupe avec aucun autre des rectangles de la liste).
9. La méthode privée **remove** supprime de la liste le rectangle donné comme paramètre.

10. La méthode privée **tryToMergeSelected** fusionne, si possible le rectangle sélectionné avec un rectangle de la liste. Le rectangle avec lequel la fusion a été effectuée est supprimé de la liste. La méthode **tryToMergeSelected** doit être réalisée de façon récursive.
11. La méthode **moveSelectedBy** déplace le rectangle sélectionné de **x** points en horizontale et de **y** points en verticale. Si une fusion avec un autre rectangle est possible après le déplacement, elle est effectuée immédiatement.

### Classe DrawPanel [1 P]

La classe **DrawPanel** sert comme support de dessin (*View*) pour une instance de **Rectangles**. Une instance de **DrawPanel** sera placée sur l'interface graphique.

### Classe MainFrame [2 + 2,5 + 0,5 + 0,5 + 4,5 = 10 P]



1. Construisez l'interface graphique montré ci-dessus. Respectez les conventions de noms du cours. Les boutons marqués **Split** et **Explode** sont désactivés au démarrage et activés seulement après le premier clic sur **New**.
2. Le bouton **New** crée une nouvelle instance de **Rectangles** en commençant par un rectangle qui est placé au milieu du canevas et dont la largeur correspond à 1/3 de la largeur du canevas et la hauteur à 1/3 de la hauteur du canevas.
3. Le bouton **Split** effectue six coupures dans les rectangles de la liste. (Cette opération peut être répétée à n'importe quel moment).
4. Le bouton **Explode** place tous les rectangles à une position aléatoire du panneau de dessin mais de façon à ce que tous les rectangles soient séparés les uns des autres.
5. A l'aide de la souris, il est possible de déplacer l'un des rectangles (*drag & drop*). Si une fusion est possible lors du déplacement, elle est effectuée automatiquement. Si lors d'un déplacement on a réussi à fusionner les 2 derniers rectangles en un seul, le message 'Congratulations' est affiché dans un dialogue.