# Projet Programmation Orientée Objet 2011-2012

## Description du sujet

L'objectif de ce projet est de programmer une application de gestion de découpe de panneaux de bois. Vous sous-traitez pour une entreprise (fictive) qui effectue des découpes de rectangles de bois spécifiées par des clients sur des planches qu'elle achète chez des fournisseurs.
Le problème de cette entreprise est qu'elle ne dispose d'aucune méthode automatique pour effectuer ces découpes, et perd donc du bois. Vous devez leur livrer une application qui va permettre d'automatiser le calcul des découpes et permettre ainsi une économie substantielle sur les achats de planches.

## Spécification des commandes clients

Les commandes clients sont spécifiées dans un fichier XML qui revêt la forme suivante :

```
<commandes>
        <commande id="1" largeur="12" longueur="14" quantite="5" />
        <commande id="2" largeur="6" longueur="48"  quantite="10"/>
        <commande id="3" largeur="9" longueur="31"  quantite="15"/>
</commandes>
```

Un élément *commandes* englobe des commandes d'un unique client. Cet ensemble de commandes devra être traitée en même temps. Chaque commande est spécifiée de la façon suivante à l'aide d'un élément *commande* qui contient les attributs suivants :

- *id* : identifiant numérique et unique de la commande.
- *largeur* : largeur de la planche en cm
- *longueur* : longueur de la planche en cm
- *quantite* : nombre de planche de largeur et longueurs données à fournir

La spécification des commandes est telle que largeur est inférieure ou égale à longueur. En outre il ne peut y avoir deux commandes avec un identifiant différent mais une largeur et longueur égales.

## Spécification des fournisseurs

Les planches de bois peuvent provenir de plusieurs fournisseur et ont donc une longueur et largeur qui peuvent varier. Elles possèdent en outre un prix unitaire. Les différents types de planches sont définies dans un fichier XML qui revêt la forme suivante :

```
<fournisseurs>
```

```
        <fournisseur id="1" largeur="140" longueur="320" prix="3.25" />
        <fournisseur id="2" largeur="320" longueur="320" prix="8" />
</fournisseurs>
```

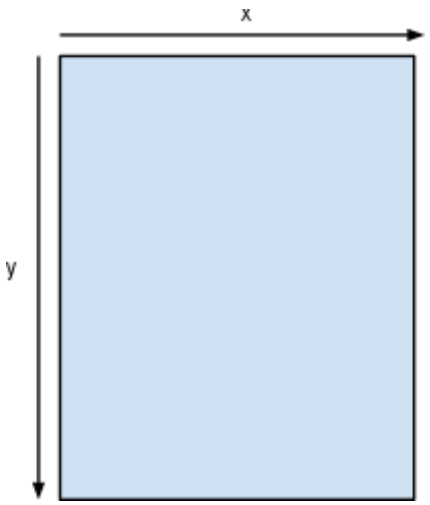
Chaque spécification est est donnée selon la façon suivante :

- *id* : identifiant numérique d'un fournisseur.
- *largeur* : largeur de la planche à découper en cm
- *longueur* : longueur de la planche à découper en cm
- *prix* : prix unitaire de la planche en euros

La spécification des fournisseurs est telle que largeur est inférieure ou égale à longueur.

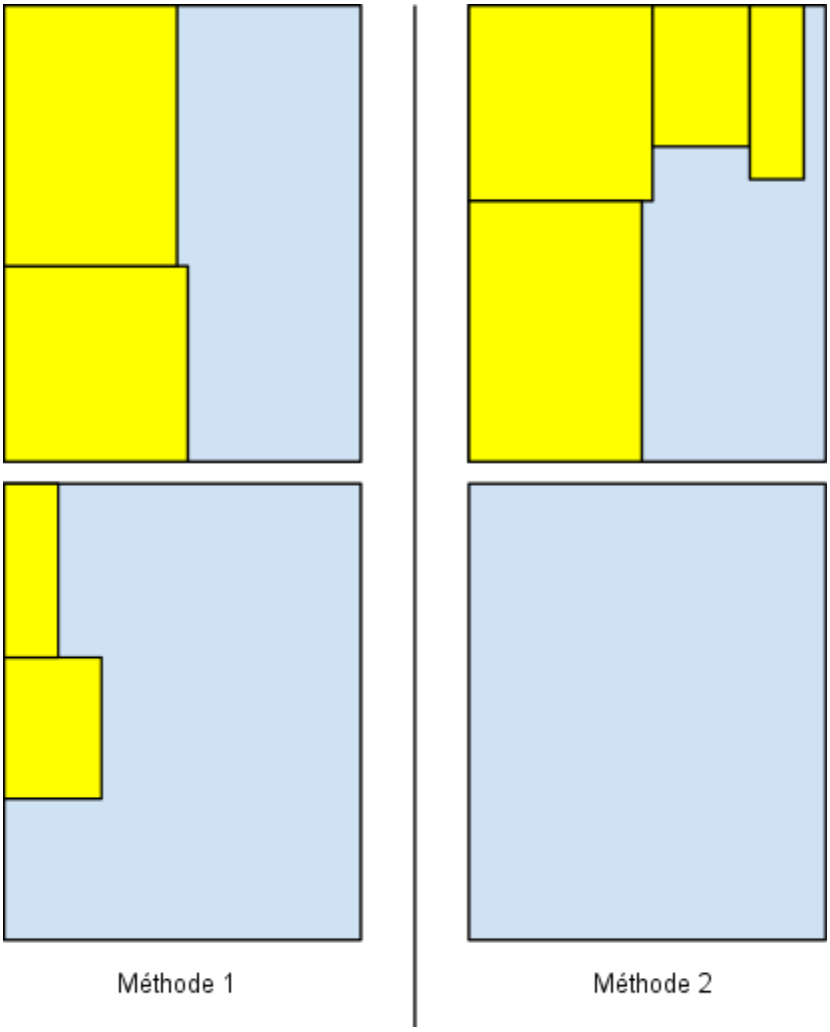
# Planification des découpes

Vous devez implémenter deux méthodes différentes de planification des découpes. Pensez à éviter au maximum toute duplication de code. En outre favorisez la réutilisabilité et l'interchangeabilité des algorithmes. (pour pouvoir utiliser un algorithme au lieu d'un autre, les changements à appliquer doivent être minimaux). Les pièces qui seront découpées sur la *planche* sont nommées par la suite *découpe.* La *planche* est l'élément amené par le fournisseur. La figure suivante montre un exemple de planche. La largeur de la *planche* correspond à la coordonnée *x* et la longueur à la coordonnée *y*. Pour un découpe d donnée, la largeur est notée l(d) et la longueur L(d). La longueur de la planche est notée L(P) et la largeur de la planche l(P).




## Méthode 1

La méthode 1 est simple mais pas très efficace. Tout d'abord les pièces sont ordonnées par ordre de longueur décroissante (en cas d'égalité par largeur décroissante). La première découpe d0 est placée, si possible, à la coordonnée (0,0). Ensuite d1 est placée à (0,L(d0)) et ainsi de suite. Si il n'est pas possible de placer une découpe sur la planche en cours car la longueur restante est trop petite, il faut alors passer sur une nouvelle planche.




### Méthode 2

La méthode 2 est théoriquement plus efficace. Le principe est de classer d'abord les *découpes* par largeur décroissante (en cas d'égalité par longueur décroissante). La première découpe d0 est placée, si possible, à la coordonnée (0,0). Ensuite les découpes suivantes sont parcourues par ordre de largeur décroissante et la première découpe di vérifiant L(di) < l(d0) et l(di) < l(P) - l(d0). Cette découpe est alors placée à la coordonnée (l(d0),0)). Ensuite les découpes suivantes sont parcourues par ordre de largeur décroissante pour trouver une découpe dj telle que L(dj) < l(d0) et l(dj) < l(P) - l(d0) - l(di). Elle est placée à la coordonnée (l(d0)+l(di),0). Cette opération est répétée jusqu'a ce que l'on ne trouve plus de planche de longueur inférieure à d0 et rentrant sur la largeur de planche restante. Quand c'est le cas, on passe à le découpe la plus large restante dk qui est placée à la coordonnée (L(d0),0) est le processus reprend. Si il n'est pas possible de placer cette découpe sur la planche en cours car la longueur restante est trop petite (si L(dk) > L(P) - L(d0)), il faut alors passer sur une nouvelle planche.

## Résultats

Votre programme permet de faire une simulation de découpe sur un fichier de commandes donné. Il produit en sorti pour chaque fournisseur:

1. Un fichier XML qui décrit les *découpes* pour la méthode 1
2. Un fichier XML qui décrit les *découpes* pour la méthode 2
3. Un fichier SVG qui montre de manière graphique les *découpes* obtenues par la méthode 1
4. Un fichier SVG qui montre de manière graphique les *découpes* obtenues par la méthode 2

Vous devez fournir une classe Main qui effectue cette opération et se lance de la manière suivante :

```
java Main commandes.xml
```

Il y a donc un total de 4*n fichiers, n étant le nombre de fournisseurs. Voici la convention de nommage de ces fichiers (dans l'ordre, pour le fournisseur 1) :

1. results.1.m1.xml
2. results.1.m2.xml
3. results.1.m1.svg
4. results.1.m2.svg

Exemple de fichiers results.1.mX.xml

```
<simulation fournisseur="1" prix="3.25"  rejets="2" algorithme="1">
        <planche id="1">
                <decoupe commande="1" x="0" y="0" />
                <decoupe commande="2" x="0" y="14" />
                <decoupe commande="3" x="0" y="62" />
        </planche id="1">
        <rejet commande="2"/>
        <rejet commande="4"/>
</simulation>
```

Vous pouvez créer les fichiers SVG en même temps que les fichiers de résultats, ou choisir de les produire à partir des fichiers de résultats.

Un élément *simulation* englobe le document. Les attributs de cet élément sont :

- *fournisseur* : l'identifiant du fournisseur pour qui la simulation a été faite.
- *prix* : le prix total du montant de la commande chez ce fournisseur. Il correspond au prix unitaire d'une *planche* multiplié par le nombre de planches utilisées.
- *rejet* : le nombre de commande non placées
- *algorithme* : le numéro de l'algorithme utilisé

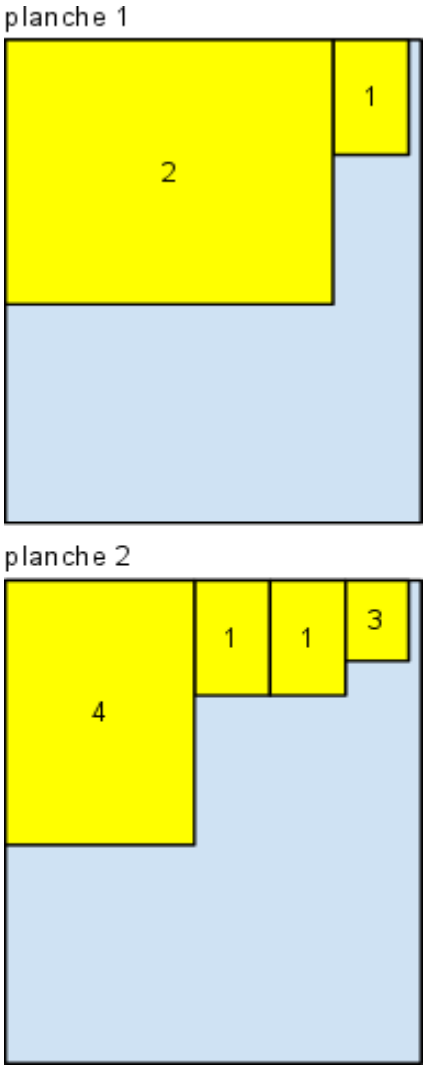
L'élément *simulation* contient ensuite plusieurs éléments *planche* et *rejet*. Les éléments *planche*

ont un attribut *id* numérique et unique correspondant aux différentes *planches* utilisées pour placer les commandes. Cet élément contient autant d'éléments *découpe* qu'il y a de rectangles à découper sur la *planche*. Les attributs de cet élément découpes sont les suivants :

- *commande* : identifiant de la commande à laquelle appartient cette découpe
- *x* : la coordonnée x de la découpe sur la planche (voir figure suivante pour le repère)
- *y* : la coordonnée y de la découpe sur la planche (voir figure suivante pour le repère)

L'élément *rejet* signifie que la commande dont l'identifiant est spécifiée à l'aide de l'attribut *commande* n'a pas été traitée car les dimensions souhaitées ne peuvent être satisfaites par les produits du fournisseur.

Le fichier SVG général doit ressembler à ceci (et ce pour chacune des planches) :




## API

Pour parser les fichiers XML vous devez utiliser l'API *XMLStreamReader*.

Pour générer le SVG vous devez utiliser l'API *XMLStreamWriter*.
Une API (*Application Programming Interface*) est une interface fourni pour un programme extérieur, généralement de librairies. Pour chacune de ces API, les services (méthodes) qu'il est possible d'utiliser depuis l'extérieur sont exposés (via la javadoc notamment). Encore une fois, un exemple d'encapsulation des données.
Ces API sont répandues et vous devrez essayer de trouver vous-même comment utiliser cette API. Cela vous fera prendre en main une API externe.
Vous pourrez trouver des informations sur le format SVG ici. Ce format peux être lu par la majorité des navigateurs web.
Pensez en général à éviter toute redondance de code, réfléchissez à la conception avant de coder. Pensez aux évolutions de votre système. Est-ce qu'intégrer un nouvel algorithme peut se faire "facilement" au sein de votre programme ?

## Contraintes sur le code

Vous devez absolument RESPECTER ces contraintes. Une pénalité de 2 points sera appliquée sur chaque projet qui ne les respectent pas.

- Votre projet doit contenir un ou plusieurs packages
- Les noms de classes et interfaces doivent commencer par une Majuscule
- Les noms de méthodes, d'attributs et de variables doivent commencer par une minuscule
- Utilisez le camel case pour les noms des classes, interfaces, méthodes et attributs.
- Les attributs doivent avoir une visibilité**.** Par défaut, les attributs sont *private***.** Des méthodes de type *get/set* doivent permettre de gérer l'accès depuis l'extérieur à ces variables. Attention, définissez ces méthodes pour chaque attribut, seulement si elles sont utiles.
- Définissez des constructeurs que lorsqu'ils sont nécessaires.
- Commentez au maximum votre code. Expliquez ce que font vos méthodes, ainsi que leurs caractéristiques un peu plus complexes. N'expliquez pas non plus tout et n'importe quoi. Soyez concis et allez à l'essentiel, indiquez ce qui est utile à la compréhension.