

Simulation d'une chaîne de production

But du laboratoire

Ce laboratoire a pour objectifs:

- d'appliquer et implémenter plusieurs concepts orientés objet
 - Héritage, polymorphisme, interface, encapsulation.
- de comprendre et appliquer des patrons de conception
 - Observateur
 - Stratégie
- d'analyser et extraire le contenu d'un fichier XML
 - concept de node, tag elements, etc...

Un dernier but est de vous apprendre à planifier le travail pour un projet d'envergure, en décomposant le problème en tâches et en fixant les dates pour des événements jalons (milestones) afin de terminer le travail dans le temps alloué.

Description du laboratoire

En industrie, la fabrication de plusieurs produits manufacturiers se fait en suivant une série d'opérations successives ou parallèles. L'organisation de ces opérations constitue une chaîne de production que le manufacturier vise généralement à optimiser. Une chaîne de production est composée de plusieurs postes, nommés usines de production. Chaque usine de production assemble ou produit un composant spécifique. Pour produire un composant, une usine de production peut nécessiter des composants produits par d'autres usines. Par exemple, dans une chaîne de montage d'avions, une usine de production assemble le groupe motopropulseur alors qu'une autre usine produit le poste de pilotage. Une autre usine de production assemble la cellule de l'avion et cela en utilisant d'autres composants tels que le fuselage, les ailes et le train d'atterrissage. Ces composants sont eux-mêmes produits par d'autres usines.

Ce laboratoire consiste à concevoir et implémenter une application qui simule une chaîne de production simplifiée. En effet, pour maximiser le flux de production et la fiabilité de la chaîne, les manufacturiers ont recours à des outils de simulation. Un outil de simulation permet de modéliser la chaîne de production et ses divers paramètres (exemple: types d'équipements, intervenants robots ou personnel, période de simulation, etc.). Le manufacturier peut alors réaliser plusieurs simulations avec différents paramètres et ainsi identifier la configuration de la chaîne qui permet d'atteindre ses objectifs de productivité.

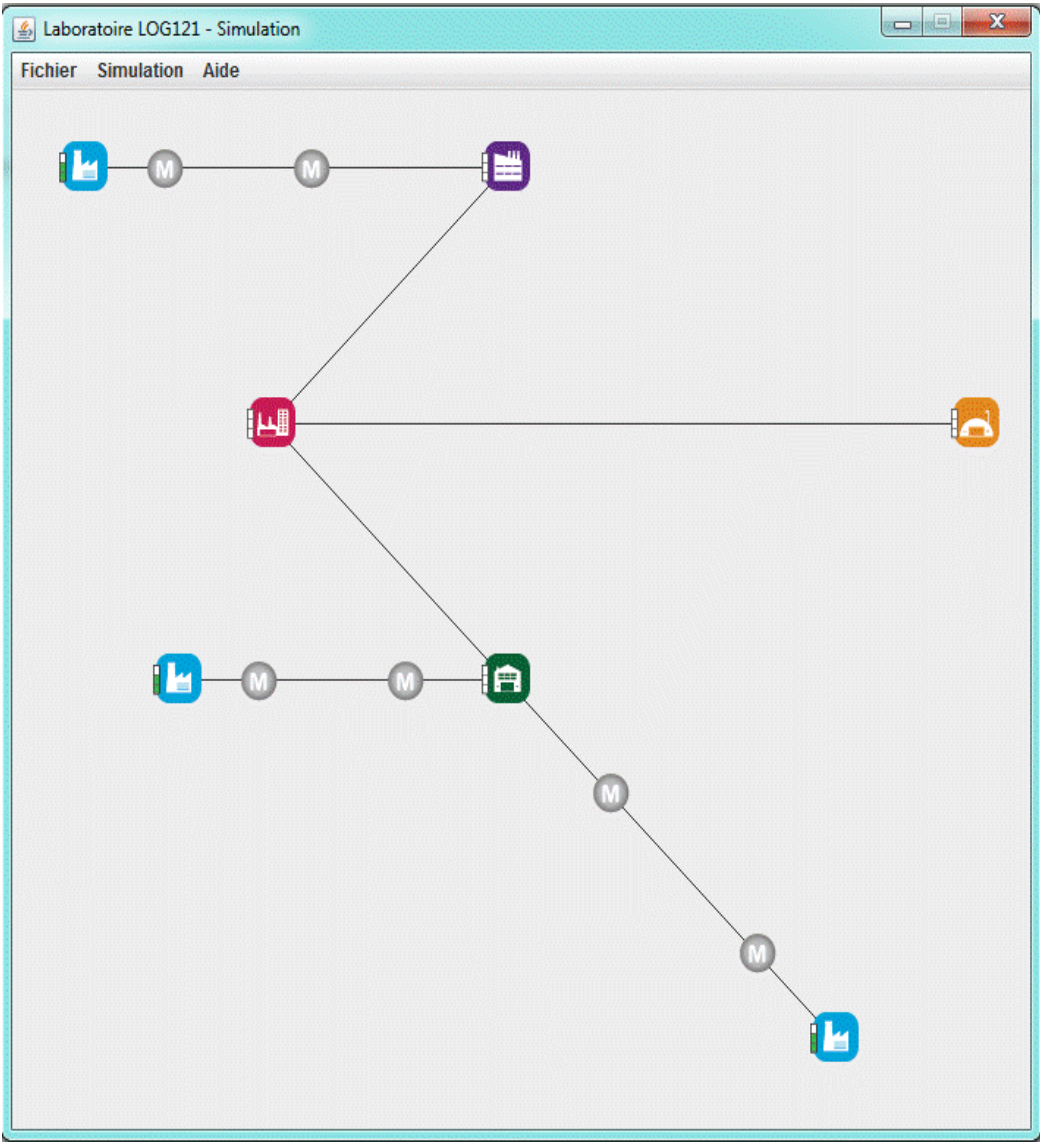
Pour les besoins de simulation, la chaîne de production sera représentée par un réseau. Dans ce réseau, chaque nœud correspond à une usine de production. Un arc dirigé, appelé chemin, relie deux usines de production lorsque l'usine de destination consomme des composants produits par l'usine de départ. Les composants assemblés sont transmis d'une usine à une autre à travers les chemins appropriés.

Note importante: Il est recommandé de lire l'énoncé en entier avant d'entamer la conception et l'implémentation du laboratoire.

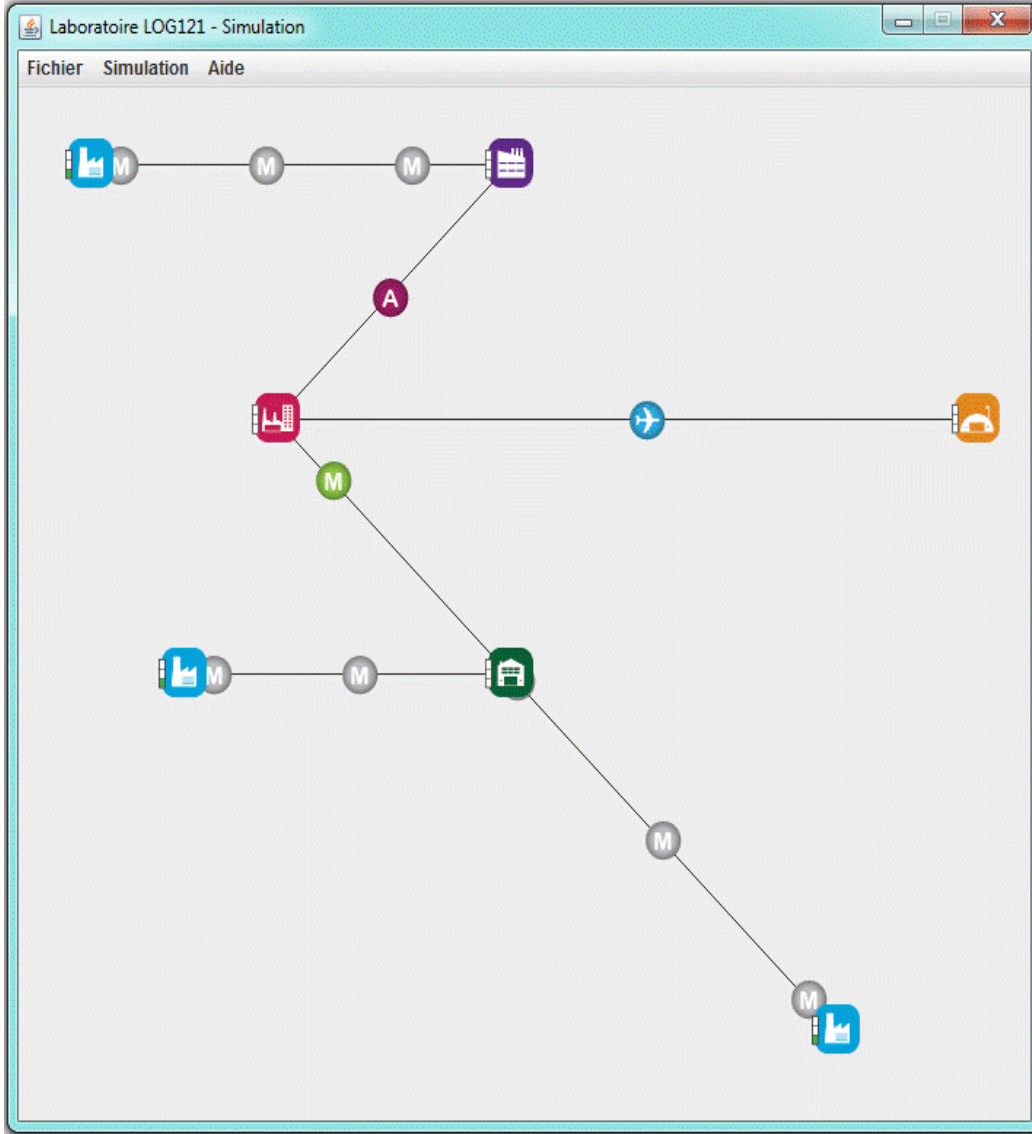
Description de la simulation

L'objectif de ce laboratoire est de simuler une chaîne de production d'avions. Celle-ci inclut des usines de production et un entrepôt dans lequel les avions produits sont emmagasinés.

La chaîne sera représentée par un réseau comme dans la figure suivante.



La figure suivante montre des composants transigeant entre toutes les usines.



Un fichier de configuration vous est fourni sous format XML. Ce fichier contient les éléments essentiels pour la création de la chaîne de production. Le fichier de configuration spécifie toutes les informations reliées aux paramètres de production des usines et aux paramètres de visualisation de la simulation. Dans ce cas-ci, la chaîne simulée inclut des usines de production et un entrepôt dans lequel les avions produits sont emmagasinés. Des détails sur les types d'usine présents sur le réseau ou les composants qui y transigent sont donnés dans les sous-sections suivantes.

La simulation sera dirigée par une horloge, autrement dit une unité de temps fixe sera utilisée. À chaque fois que cette unité de temps, appelée **tour**, est écoulée, vous devez explorer l'état de tous les éléments de la chaîne (les usines, l'entrepôt et les composants produits) et synchroniser votre simulation avec cet état.

Paramètres de la chaîne de production

Selon le type d'usine, la production se fait à différents intervalles de temps et sous certaines conditions. Ces informations sont données dans la balise "metadonnees" du fichier de configuration.

Pour chaque type d'usines, les informations suivantes sont données :

- Les chemins vers les fichiers des icônes représentant chaque usine.
- Les composants reçus en entrée par l'usine et la quantité nécessaire de chaque composant pour démarrer la production. Certaines usines ne nécessitent pas de composants en entrée

(ex : usine-matiere dans le fichier de configuration). D'autres usines peuvent nécessiter différents types de composants avec différentes quantités (ex : usine-assemblage dans le fichier de configuration).

- Les composants produits en sortie par l'usine.
- L'intervalle de production lequel spécifie la durée que prend l'usine pour fabriquer ou monter le composant qu'elle donne en sortie.

L'entrepôt est un cas particulier d'usine où il n'y a pas de production. On entrepose dedans les avions produits par la chaîne. Donc, l'entrepôt n'a pas de composants en sortie ni d'intervalle de production. Par contre l'entrepôt a une limite de capacité d'entreposage qu'il ne peut dépasser. Cette capacité est spécifiée dans le fichier de configuration comme attribut du type d'entrée que l'entrepôt reçoit, à savoir les avions.

Le nombre d'avions dans l'entrepôt diminue quand une vente a lieu. Différentes stratégies de vente peuvent être utilisées et il faut gérer le flux de la chaîne de production en conséquence. Dans notre simulation, nous considérons les deux stratégies suivantes:

- La vente se fait selon une fonction de probabilité aléatoire.
- La vente se fait à des intervalles fixes; par exemple après avoir produit un certain nombre d'avions.

Dans tous cas, si l'entrepôt est rempli, il faut arrêter la production et attendre qu'un avion soit vendu et donc déplacé de l'entrepôt. Pour éviter une telle situation, il faut gérer le flux de la chaîne de production selon l'état de l'entrepôt pour que les usines produisent de façon optimale.

Paramètres de la visualisation de la simulation

Les usines et l'entrepôt sont affichés sous forme d'icônes dans la simulation. De plus, une usine affiche un indicateur qui spécifie le temps écoulé dans la production alors que l'entrepôt affiche la quantité de produits qu'il contient.

Prendre note que chaque usine possède quatre variantes de son icône, une pour chaque état de son indicateur.

Indicateur

Signification



Vide (pour une usine, cela signifie qu'elle n'a pas commencé la production. Pour un entrepôt, cela signifie qu'il est vide)



Un-tiers (pour une usine, cela signifie qu'un tiers du temps nécessaire à la production a été écoulé. Pour un entrepôt, cela signifie qu'il est à un tiers de sa capacité de stockage)



Deux-tiers (pour une usine, cela signifie que deux tiers du temps nécessaire à la production a été écoulé. Pour un entrepôt, cela signifie qu'il est à deux tiers de sa capacité de stockage)



Plein (pour une usine, cela signifie qu'elle a fini la production. Pour un entrepôt, cela signifie qu'il est plein)

Le chemin vers les fichiers des icônes des usines sont donnés dans le fichier de configuration.

Les composants produits ou reçus par les usines et l'entrepôt ont chacun une icône les représentant :

Composant Icône

Métal



Aile



Moteur



Avion



L'ensemble des icônes des usines et des composants vous sont fournies dans le dossier */ressources* du squelette qui vous est fourni [ici](#).

Travail à réaliser

Votre travail consiste à:

- Concevoir et implémenter les classes de base correspondant aux différents types d'usine et aux différents types de composants produits par les usines;
- Concevoir et implémenter les classes représentant le réseau, ses nœuds et les chemins les reliant.
- Concevoir et implémenter les classes permettant de lire le fichier de configuration et de créer les éléments du réseau et de la simulation.
- Choisir et appliquer les patrons appropriés pour répondre aux contraintes décrites dans la simulation.

Contraintes de conception

Vous devez faire une conception préliminaire avant de commencer à implémenter le laboratoire. La conception doit être orientée objet et facile à étendre. Vos classes doivent être cohésives et vous devez minimiser le couplage entre les classes.

De plus, votre conception doit être organisée en différents packages pour faciliter la compréhension et la maintenance de l'application.

Utilisation obligatoire de l'héritage et du polymorphisme : Il est évident que les différents types d'usines ont un nombre de propriétés communes. Il en est de même pour les composants produits par les usines. De plus, les usines et les composants ont aussi quelques propriétés similaires. Une utilisation judicieuse du polymorphisme vous permettra de réutiliser des propriétés qui sont communes entre les usines et les composants.

Utilisation obligatoire du patron Observateur : L'entrepôt ayant une capacité limitée de stockage, il faut arrêter la production dans le cas où il est rempli et la reprendre quand il y a une vente d'un avion. Vous devez donc obligatoirement appliquer le patron Observateur pour minimiser le couplage entre les classes concernées.

Utilisation obligatoire du patron stratégie : Plusieurs stratégies peuvent être employées pour simuler une vente d'avion. Vous devez implémenter les deux stratégies de vente données dans la description de la simulation.

Détails d'implémentation du laboratoire

Squelette pour l'application :

Vous devez utiliser ce [squelette](#) pour une application Swing ayant un fil (Thread) qui exécute la simulation de la production en faisant appel à vos propres classes.

Fichier de configuration :

Le fichier de configuration de la simulation vous est fourni sous format XML. Celui-ci se trouve dans le dossier */ressources* du squelette qui vous est fourni.

La balise "metadonnees" du fichier de configuration regroupe la description des paramètres des différentes usines et de l'entrepôt. Comme expliqué plus haut dans la section "Description de la simulation", ces paramètres vous informent sur les détails d'implémentation des classes usines et entrepôts. Les informations données sous la balise "metadonnees" sont communes à plusieurs simulations.

Attention : Les valeurs des attributs des éléments listés dans cette balise ne doivent pas être "hard-codés" dans les classes mais récupérés à même le fichier de configuration lorsqu'une simulation est lancée.

La balise "simulation" contient les instances d'usines de la chaîne de production qu'on veut simuler. Ces instances ont toutes un attribut *x*, un attribut *y* et un attribut *id*. Les attributs *x* et *y* vous serviront à placer l'usine ou l'entrepôt dans l'espace. L'attribut *id* est un identifiant de l'instance. Ces identifiants sont utilisés pour créer les chemins reliant les différentes instances d'usines de la simulation.

Pour la lecture du fichier XML et la récupération des éléments contenus dans le fichier, référez-vous aux explications et ressources données dans la section [Ressources pour la lecture du fichier XML](#).

Pour chaque élément récupéré du fichier de configuration et qui fait partie de la balise "simulation", vous devez automatiquement instancier la bonne classe et initialiser ses attributs à partir des valeurs dans le fichier XML.

Dessin du réseau

Une fois que vous avez créé les instances d'usines faisant partie de la simulation, vous devez dessiner le réseau correspondant à ces instances. Cette opération doit se faire à l'intérieur de la méthode *paint()* de la classe FenetrePrincipale. L'objet *Graphics*, passé en paramètre à la méthode, vous permettra de dessiner le réseau.

Une usine est représentée par un nœud affichant une des quatre icônes lui correspondant; l'icône initiale étant celle correspondant à l'icône indiquant l'état vide. Les chemins reliant les usines sont représentés par des lignes. Pour faciliter l'implémentation, vous devez être en mesure de récupérer les points de départ et d'arrivée d'un chemin pour identifier les usines que ce chemin relie.

Il est important de savoir que le canvas de Swing a un système d'axes où l'axe des ordonnées est inversé. Afin de simplifier l'implémentation du réseau, le système d'axes choisi sera celui du canvas. **Ainsi les Y positifs sont orientés vers le bas.**

Utilisation du réseau pour réaliser la simulation

Pour réaliser cette étape, il est recommandé de réduire la taille de votre réseau à deux usines et d'en ajouter graduellement. C'est ici que la notion de temps est prise en compte. Chaque élément de votre réseau devra agir en fonction d'un cycle d'horloge appelé un **tour**. L'aspect temporel de

la simulation doit être implémenté à l'intérieur de la méthode *doInBackground()* de la classe Environnement. Cette méthode sera appelée indéfiniment tant que la simulation sera active.

Dans cette méthode, il faut que les usines produisent lorsqu'elles rencontrent les critères nécessaires à la production. Pour cela, vous devez appeler une méthode qui informera l'usine qu'un tour est passé. Si le nombre de temps écoulé à la production est suffisant et que la quantité de composants nécessaires en entrée est disponible, un composant sera produit et ajouté au réseau.

Une fois que chaque usine a produit, on effectue le déplacement des composants (métal, aile, moteur et avion) sur le réseau. Ces derniers doivent contenir un vecteur de déplacement en x et en y (une vitesse). Les valeurs du vecteur seront ajoutées à la position du composant à chaque tour. De plus, la vitesse est calculée en fonction du chemin sur lequel le composant est envoyé; autrement dit la vitesse est calculée en fonction de la position de l'usine de départ et de l'usine d'arrivée que ce chemin relie.

Exemple de calcul de vitesse:

Position de l'usine 1: [x:10, y:10]

Position de l'usine 2: [x:10, y:25]

Doit correspondre à une vitesse de [dx:0, dy:1] pour le composant

Un exemple de déplacement de composant est fourni dans le squelette.

Voici quelques éléments à considérer afin de simplifier l'implémentation de cette fonctionnalité:

- La vitesse de déplacement de chaque entité est de ± 1 pixel en x et/ou en y.
- Les angles entre les usines sont toujours d'un facteur de 45 degrés (0, 45, 90, 135, 180, 225, 270, 315, 360)
- Grâce à ce qui a été mentionné précédemment, il est possible de calculer la rencontre (i.e., collision) entre deux entités en comparant simplement leur position en x et en y.

Les rencontres entre les composants et les usines doivent être calculées à chaque tour. Afin de simplifier votre implémentation, le réseau a été construit de manière à alimenter les usines avec les bons composants. Pas besoin de traiter le cas où le composant ne correspond pas au type que l'usine a besoin de recevoir. Lorsqu'un composant rencontre une usine ou un entrepôt, celui-ci est ajouté à la liste des composants disponibles de l'usine ou de l'entrepôt et doit être retiré de la simulation.

Optimisation de la production des usines

Il faut arrêter ou reprendre la production selon le nombre d'avions entreposés dans l'entrepôt. C'est pourquoi vous devez utiliser le patron observateur pour être en mesure d'informer les usines de l'état de l'entrepôt.

En plus, vous devez adapter cette implémentation de façon à réguler la production des usines de façon à optimiser le rythme de production de toute la chaîne selon la fréquence des ventes. Autrement dit, selon l'état de l'entrepôt, les usines doivent accélérer ou réduire leur rythme de production pour éviter l'arrêt complet de la chaîne. Le mécanisme pour réguler le rythme de production d'une usine est laissé à votre discrétion.

Planification et déroulement du travail

Ce projet représente plusieurs défis. Donc, il faudra bien s'organiser afin d'atteindre le but dans le temps alloué (et d'avoir une bonne note!). Voici quelques stratégies pour vous aider à réussir :

- Décomposer le travail en sous tâches, par exemple, la conception et programmation du code pour :
 - Les classes d'usines et des composants;
 - L'analyse et la lecture du fichier de configuration et la création des usines et du réseau;
 - L'affichage à l'écran du réseau;
 - Le déroulement de la simulation;
 - ... (cette liste n'est pas complète et n'est qu'une suggestion).
- Faire un estimé de combien de temps ça va prendre pour chaque sous tâche.
- Fixer les dates pour des événements jalons. Par exemple, à la fin de la première semaine, la sous tâche X sera complétée.
- Commencer les sous tâches où vous vous sentez le moins à l'aise **le plus tôt possible**. Le but est d'éliminer les incertitudes le plus vite possible.
- Revoir votre plan au moins une fois par semaine pour l'ajuster au fur et à mesure que vous avancez.

Rapport de laboratoire

Le contenu du rapport est indiqué dans le gabarit sur le site de votre groupe.

Votre rapport doit comprendre:

- des diagrammes de classe en UML qui représentent votre conception.
- des diagrammes de séquence en UML illustrant les interactions entre les objets de votre solution.
 - Au minimum, un diagramme de séquences est exigé pour illustrer la dynamique du patron GoF "Observateur" dans le contexte de votre solution.

Veuillez aussi consulter la rubrique [Style de correction et Rapports de laboratoire](#) pour d'autres détails.

Pondération

Ce laboratoire compte pour 16%. Voilà comment les points sont alloués :

- 8 points = Fonctionnalité et présentation
- 8 points = Rapport de laboratoire

Date de remise et de présentation

Voir le site de chaque groupe pour la date de remise.

Vous devez respecter la procédure de remise des travaux.

La procédure de remise est expliquée sur le site de chaque groupe.

Ressources pour la lecture du fichier XML

Il existe différentes façons de lire (parser) un fichier XML. Nous vous expliquons brièvement ici comment lire le fichier XML en utilisant DOM (Document Object Model). Ce dernier est une API standardisée par le W3C. En utilisant DOM, le contenu d'un fichier XML peut être lu et modifié.

Un DOM représente le contenu du fichier XML sous forme d'une structure en arbre. De façon simple, les nœuds de l'arbre correspondent aux éléments du fichier XML et la hiérarchie des

nœuds du DOM reflètent les imbrications des éléments dans le fichier XML représenté par le DOM. Grâce à l'API DOM, vous pouvez donc parcourir les nœuds de l'arbre correspondant aux éléments du fichier XML.

Pour construire un DOM à partir du fichier XML, vous avez besoin d'importer les classes suivantes de la librairie Java :

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
```

La classe suivante pour les exceptions levées lors de la lecture du fichier XML:

```
import org.xml.sax.SAXException;
```

Les classes suivantes pour les définitions W3C du DOM et de son contenu :

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
```

Supposons que le chemin vers le fichier de configuration XML est stocké dans la chaîne de caractères nommée " filePath ". Vous devez en premier construire un Document (DOM) à partir du fichier XML comme suit:

```
//Instancier la Factory qui permet d'accéder à un parser (appelé DocumentBuilder)
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
//Récupérer le parser
DocumentBuilder db = dbf.newDocumentBuilder();
//Parser le fichier XML
Document doc = db.parse(new File(filePath));
doc.getDocumentElement().normalize();
```

Vous pouvez ensuite accéder aux éléments dans le document. L'API DOM vous donne plusieurs méthodes pour parcourir le document:

```
//accès à la racine du document
doc.getDocumentElement();
//Récupération d'un ensemble d'éléments ayant le même nom
NodeList nList = doc.getElementsByTagName("nomElementRecherché");
//La liste des nœuds peut être parcourue et on peut accéder aux attributs et contenu de
chaque élément de la liste.
```

Pour découvrir les autres méthodes offertes par l'API Java pour manipuler un document DOM, voilà des ressources additionnelles où vous pouvez trouver des exemples de lecture de fichier XML:

<https://docs.oracle.com/javase/tutorial/jaxp/dom/readingXML.html>

<https://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/>

https://www.tutorialspoint.com/java_xml/java_dom_parse_document.htm