Table des matières

[*I-* *Modélisation :* 2](#_Toc165473882)

[1. Convoyeur d'entrée : 2](#_Toc165473883)

[Les pièces arrivent à l'usine de production par un convoyeur d'entrée. Ce convoyeur achemine les pièces jusqu'à la station de traitement. Il est modélisé par une file d'attente nommée convoyeurDEntree de type Queue<Piece>. 2](#_Toc165473884)

[2. Station de traitement : 2](#_Toc165473885)

[3. Convoyeur de sortie : 2](#_Toc165473886)

[4. Station de stockage : 2](#_Toc165473887)mm

[5. Choix des structures de données : 3](#_Toc165473888)

[*II-* *Classes :* 3](#_Toc165473889)

[Piece 3](#_Toc165473890)

[BoiteStockage 3](#_Toc165473891)

[StationDeTraitement 4](#_Toc165473892)

[StationDeStockage 5](#_Toc165473893)

[*III-* *Enumération :* 6](#_Toc165473894)

[EtatPiece 6](#_Toc165473895)

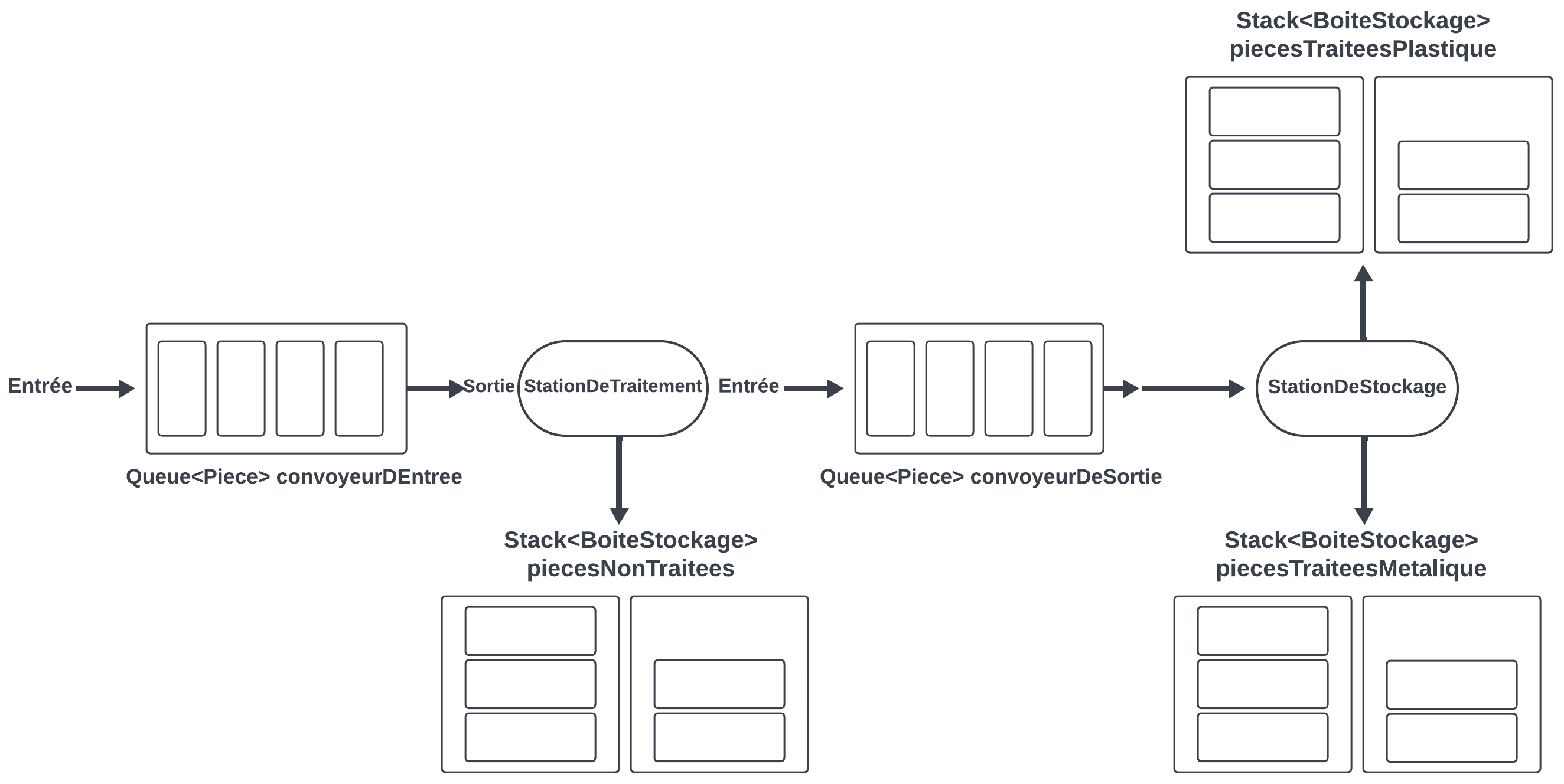
[Materiaux 7](#_Toc165473896)

[TypeMateriaux 7](#_Toc165473897)

[EtatRemplissage 7](#_Toc165473898)

# *Modélisation :*

Le schéma suivant présente la modélisation de processus de production :



## Convoyeur d'entrée :

## Les pièces arrivent à l'usine de production par un convoyeur d'entrée. Ce convoyeur achemine les pièces jusqu'à la station de traitement. Il est modélisé par une file d'attente nommée convoyeurDEntree de type Queue<Piece>.

## Station de traitement :

La station de traitement, implémentée dans la classe **StationDeTraitement.java**, reçoit les pièces du **convoyeurDEntree**. Elle traite les pièces et les renvoie dans une nouvelle file d'attente. Les pièces non traitées sont quant à elles stockées dans une pile nommée **piecesNonTraitees** de type **Stack<BoiteStockage>.**

## Convoyeur de sortie :

Le convoyeur de sortie est modélisé par une file d'attente **Queue<Piece> convoyeurDeSortie**. Cette structure de données permet de gérer efficacement l'ordre d'arrivée des pièces traitées et d'assurer une livraison fluide à la station de stockage.

## Station de stockage :

La station de stockage, implémentée dans la classe **StationDeStockage.java**, reçoit les pièces du **convoyeurDeSortie**. Elle stocke les pièces dans deux piles de boites de stockage une pour les pièces métalliques et une pour les pièces plastiques.

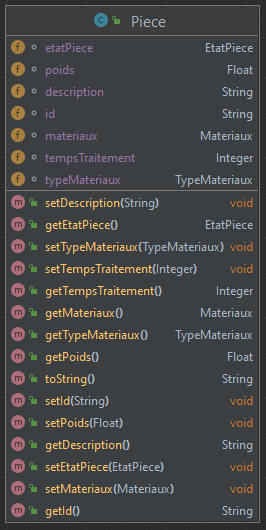
## Choix des structures de données :

Les files d'attente (Queue) sont utilisées pour représenter les convoyeurs, car elles modélisent bien le comportement FIFO (premier entré, premier sorti) des pièces dans le processus de production.

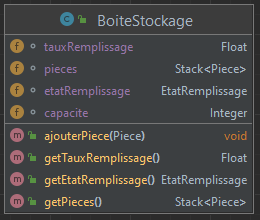
Les piles (Stack) sont utilisées pour stocker les pièces traitées et non traitées, car elles permettent un accès LIFO (dernier entré, premier sorti), ce qui est pratique pour le stockage et le traitement des pièces.

# *Classes :*

Piece : Cette classe représente une pièce individuelle avec ses attributs (id, description, materiaux, poids, etat de la piece, temps de traitement, type de matériaux).



BoiteStockage : Cette classe représente une boîte de stockage. Elle contient une pile pour stocker les pièces, un état de remplissage de la boîte, un taux de remplissage de la boîte, et une capacité maximale de la boîte. Elle contient également une méthode pour ajouter une pièce à la boîte.



StationDeTraitement :

La classe **StationDeTraitement** est une implémentation de l’interface **Station**. Elle représente une station de traitement dans notre système de gestion de production. Elle a plusieurs attributs, y compris les convoyeurs d’entrée et de sortie (des files d'attente), une pile pour les pièces non traitées, le temps de traitement maximum autorisé pour une pièce, la capacité maximale des boîtes de stockage, et un compteur pour le nombre de pièces traitées.

Constructeur : Le constructeur de cette classe initialise tous les attributs mentionnés ci-dessus.

Méthode lancerTraitement : Cette méthode lance le traitement des pièces. Elle parcourt toutes les pièces dans le convoyeur d’entrée. Si le temps de traitement d’une pièce dépasse le maximum autorisé, la pièce est considérée comme non traitée et est ajoutée à la pile de stockage des pièces non traitées. Sinon, la pièce est considérée comme traitée, le nombre de pièces traitées est incrémenté, et la pièce est ajoutée à la file d’attente de sortie.

public void lancerTraitement()

Méthodes statistiques : Cette méthode affiche le nombre de pièces traitées.

public void statistiques()

Méthode ajouterPieceNonTraitee : Cette méthode ajoute une pièce non traitée à la pile de stockage. Si la pile de stockage des pièces non traitées est vide ou si la dernière boîte est pleine, une nouvelle boîte est ajoutée à la pile et la pièce est stockée dans cette boîte. Sinon, la pièce est stockée dans la dernière boîte ajoutée à la pile.

public void ajouterPieceNonTraitee(Piece piece)

Méthode getPiece : Cette méthode récupère une pièce de la file d’attente d’entrée.

public Piece getPiece()

En résumé, la classe **StationDeTraitement** est responsable du traitement des pièces. Elle vérifie le temps de traitement de chaque pièce, décide si la pièce est traitée ou non, et gère le stockage des pièces non traitées.

StationDeStockage :

Classe **StationDeStockage** : Cette classe implémente l’interface Station. Cette classe gère le stockage des pièces et représente une station de stockage. Elle a plusieurs attributs, y compris les convoyeurs de sortie (une file d'attente), deux piles pour les pièces traitées (une pour les pièces métalliques et une pour les pièces en plastiques), la capacité maximale des boîtes de stockage.

Elle contient des méthodes pour lancer le traitement des pièces, afficher les statistiques des pièces traitées, avertir lorsque la boîte de stockage est remplie à 80%, ajouter une pièce traitée en plastique ou en métal à la pile correspondante, et récupérer une pièce de la file d’attente.

Constructeur : Le constructeur de cette classe initialise tous les attributs mentionnés ci-dessus.

Méthode lancerTraitement : Cette méthode traite les pièces dans le convoyeur de sortie. Si une pièce est traitée, elle est ajoutée à la pile correspondante en fonction de son matériau.

public void lancerTraitement()

Méthodes statistiques : Cette méthode affiche les statistiques des pièces traitées.

public void statistiques()

Méthode avertir80PourcentPlein : Cette méthode avertit lorsque la boîte de stockage est remplie à 80%.

private void avertir80PourcentPlein()

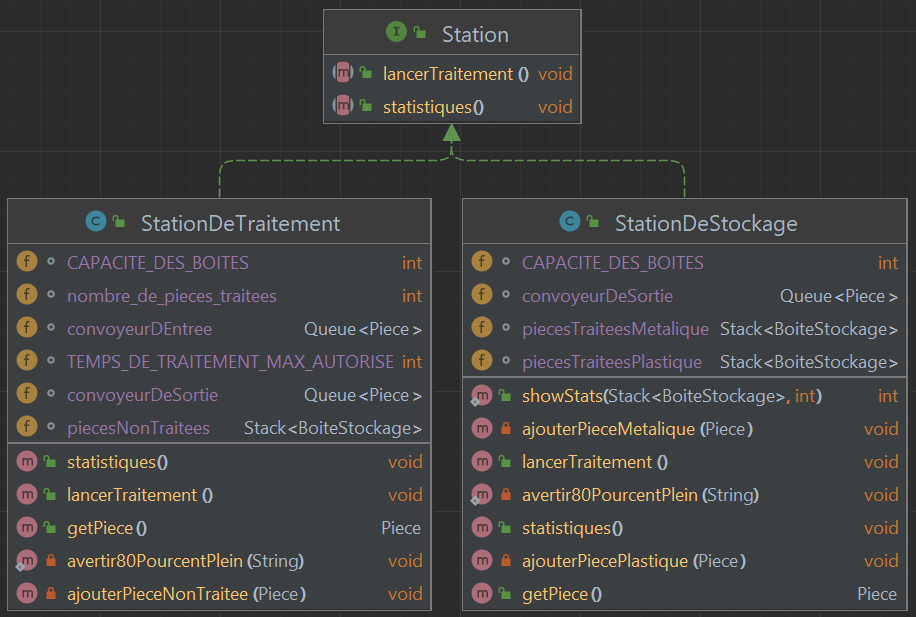
Méthodes ajouterPiecePlastique : Cette méthode ajoute une pièce métallique traitée à la pile des pièces métalliques traitées.

private void ajouterPiecePlastique(Piece piece)

Méthodes ajouterPieceMetalique : Cette méthode ajoute une pièce plastique traitée à la pile des pièces plastiques traitées.

private void ajouterPieceMetalique(Piece piece)

Les pièces sont d’abord traitées dans la **StationDeTraitement**. Ensuite, elles sont envoyées au convoyeur de sortie. La **StationDeStockage** récupère les pièces du convoyeur de sortie et les stocke dans les piles correspondantes en fonction de leur matériau. Si une boîte de stockage est remplie à 80%, un avertissement est émis. Si une boîte de stockage est pleine, une nouvelle boîte est ajoutée à la pile. Les statistiques des pièces traitées sont affichées à la fin.



# *Enumération :*

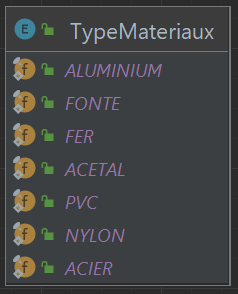
EtatPiece: avec les états suivants : NON\_TRAITEE, TRAITEE, EN\_COURS



Materiaux : avec les types suivants : METAL, PLASTIQUE.



TypeMateriaux : avec les types suivants : ACIER, FER, ALUMINIUM, FONTE, PVC, ACETAL, NYLON.



EtatRemplissage: avec les états suivants : PLEINE, PARTIELLEMENT\_PLEINE, VIDE.

