# Gestion de Stock

1.0

Généré par Doxygen 1.15.0

# Chapitre 1

# Hiérarchie de répertoires

## 1.1 Répertoires

# Chapitre 2

# Index des espaces de nommage

## 2.1 Liste des paquetages

Liste des paquetages avec une brève description (si disponible) :

# Chapitre 3

# Index hiérarchique

## 3.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

# Chapitre 4

# Index des classes

## 4.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

# Chapitre 5

# Index des fichiers

## 5.1 Liste des fichiers

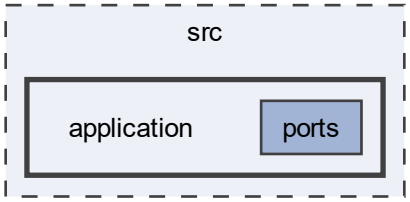
Liste de tous les fichiers avec une brève description :

# Chapitre 6

# Documentation des répertoires

## 6.1 Répertoire de référence de src/application

Graphe des dépendances de répertoires pour application:




**Répertoires**

— répertoire ports

**Fichiers**

— fichier AddProductUseCase.java

— fichier DeleteProductUseCase.java

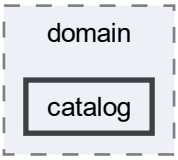
— fichier SearchProductUseCase.java

— fichier SellProductUseCase.java

— fichier SortProductsUseCase.java

## 6.2 Répertoire de référence de src/domain/catalog

Graphe des dépendances de répertoires pour catalog:

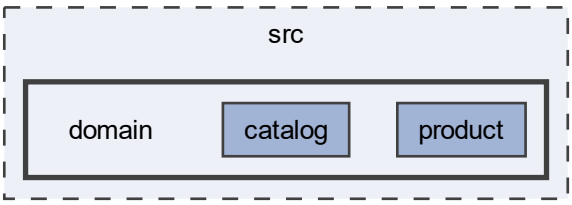



### Fichiers

- fichier Brand.java
- fichier Catalog.java
- fichier ProductLine.java

## 6.3 Répertoire de référence de src/domain

Graphe des dépendances de répertoires pour domain:

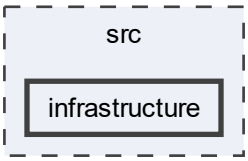



### Répertoires

- répertoire catalog
- répertoire product

## 6.4 Répertoire de référence de src/infrastructure

Graphe des dépendances de répertoires pour infrastructure:




**Fichiers**

— fichier InMemoryProductRepository.java

— fichier ProductIdGenerator.java

## 6.5 Répertoire de référence de src/application/ports

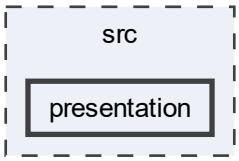
Graphe des dépendances de répertoires pour ports:


**Fichiers**

— fichier ProductRepository.java

## 6.6 Répertoire de référence de src/presentation

Graphe des dépendances de répertoires pour presentation:




### Fichiers

— fichier ConsoleInterface.java

## 6.7 Répertoire de référence de src/domain/product

Graphe des dépendances de répertoires pour product:


### Fichiers

— fichier ContainerProduct.java

— fichier Product.java

— fichier SimpleProduct.java

## 6.8 Répertoire de référence de src

### Répertoires

— répertoire application

— répertoire domain

— répertoire infrastructure

— répertoire presentation

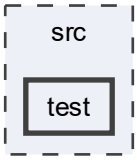
— répertoire test

### Fichiers

— fichier Main.java

## 6.9 Répertoire de référence de src/test

Graphe des dépendances de répertoires pour test:




### Fichiers

— fichier TestRunner.java

# Chapitre 7

# Documentation des espaces de nommage

## 7.1 Paquetage application

### Paquetages

— package ports

### Classes

— class AddProductUseCase

*Use case responsable de l'ajout d'un produit dans le stock.*

— class DeleteProductUseCase

*Cas d'usage pour la suppression d'un produit du stock.*

— class SearchProductUseCase

*Cas d'usage : Rechercher des produits dans le stock.*

— class SellProductUseCase

*Cas d'usage : Vendre un produit du stock.*

— class SortProductsUseCase

*Cas d'usage : Trier les produits du stock par ordre croissant de prix.*

## 7.2 Paquetage application.ports

### Classes

— interface ProductRepository

*Interface représentant le port d'accès aux données des produits.*

## 7.3 Paquetage domain.catalog

### Classes

— class Brand

*Représente une marque à laquelle peuvent être associées une ou plusieurs gammes de produits.*

— class Catalog

*Fournit un registre centralisé pour la gestion des marques (`Brand`) et des gammes de produits (`ProductLine`).*

— class ProductLine

*Représente une gamme de produits.*

## 7.4 Paquetage domain.product

### Classes

— class ContainerProduct

*Représente un produit conteneur, c'est-à-dire un produit composé qui contient une certaine quantité d'un autre produit.*

— class Product

*Représente un produit abstrait appartenant à un catalogue.*

— class SimpleProduct

*Représente un produit simple du catalogue.*

## 7.5 Paquetage infrastructure

### Classes

— class InMemoryProductRepository

*Implémentation en mémoire de l'interface `ProductRepository`.*

— class ProductIdGenerator

*Générateur d'identifiants uniques pour les produits.*

## 7.6 Paquetage presentation

### Classes

— class ConsoleInterface

*Interface utilisateur en ligne de commande pour la gestion de stock.*

## 7.7 Paquetage test

### Classes

— class TestRunner

*Classe de test manuelle de l'application de gestion de stock.*

# Chapitre 8

# Documentation des classes

## 8.1 Référence de la classe application.AddProductUseCase

Use case responsable de l'ajout d'un produit dans le stock.

#### Fonctions membres publiques

- AddProductUseCase (ProductRepository productRepository)

  *Construit le cas d'usage d'ajout de produit avec le dépôt spécifié.*

- void execute (Product product)

  *Ajoute un produit au stock.*

### 8.1.1 Description détaillée

Use case responsable de l'ajout d'un produit dans le stock.

Si un produit avec le même identifiant existe déjà dans le référentiel, son stock est incrémenté et son prix mis à jour. Le nom, la marque et la gamme d'origine sont conservés.

Cette logique assure une unicité forte sur l'identifiant produit, en cohérence avec la gestion automatique des identifiants côté infrastructure.

Cette classe fait partie de la couche application et dépend d'un port `ProductRepository`.

**Auteur**

Lucas

**Version**

1.2

### 8.1.2 Documentation des constructeurs et destructeur

#### 8.1.2.1 AddProductUseCase()

```
application.AddProductUseCase.AddProductUseCase (
            ProductRepository productRepository)
```

Construit le cas d'usage d'ajout de produit avec le dépôt spécifié.

**Paramètres**

| | |
|---|---|
| *productRepository* | le dépôt de persistance des produits |

### 8.1.3 Documentation des fonctions membres

#### 8.1.3.1 execute()

```
void application.AddProductUseCase.execute (
            Product product)
```

Ajoute un produit au stock.

Si l'identifiant du produit existe déjà : le stock existant est incrémenté le prix est mis à jour avec la nouvelle valeur le nom, la marque et la gamme existants sont conservés

Sinon, le produit est inséré tel quel.

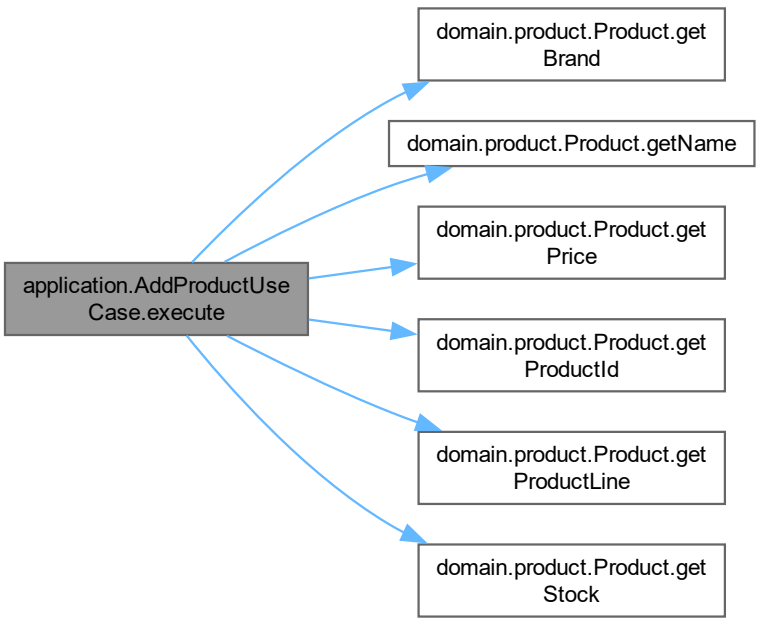
**Paramètres**

| | |
|---|---|
| *product* | le produit à enregistrer ou mettre à jour |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si le produit est nul ou invalide |

Voici le graphe d'appel pour cette fonction :




La documentation de cette classe a été générée à partir du fichier suivant :
— src/application/AddProductUseCase.java

## 8.2 Référence de la classe domain.catalog.Brand

Représente une marque à laquelle peuvent être associées une ou plusieurs gammes de produits.

#### Fonctions membres publiques

— Brand (String brandId, String name)

*Construit une instance de Brand avec un identifiant et un nom.*

— String getBrandId ()

*Retourne l'identifiant unique de la marque.*

— String getName ()

*Retourne le nom de la marque.*

— boolean equals (Object o)

*Vérifie l'égalité entre deux marques sur la base de leur identifiant.*

— int hashCode ()

*Calcule le hashcode basé sur l'identifiant de la marque.*

— String toString ()

*Fournit une représentation textuelle lisible de la marque.*

### 8.2.1 Description détaillée

Représente une marque à laquelle peuvent être associées une ou plusieurs gammes de produits.

Chaque marque est identifiée de manière unique par un identifiant stable (brandId) et possède un nom affiché lisible par un humain.

**Auteur**

Lucas

**Version**

1.1

### 8.2.2 Documentation des constructeurs et destructeur

#### 8.2.2.1 Brand()

```
domain.catalog.Brand.Brand (
            String brandId,
            String name)
```

Construit une instance de Brand avec un identifiant et un nom.

**Paramètres**

| | |
|---|---|
| *brandId* | identifiant unique de la marque (non nul) |
| *name* | nom lisible de la marque (non nul) |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si l'un des paramètres est nul |

Voici le graphe des appelants de cette fonction :


### 8.2.3 Documentation des fonctions membres

#### 8.2.3.1 equals()

```
boolean domain.catalog.Brand.equals (
            Object o)
```

Vérifie l'égalité entre deux marques sur la base de leur identifiant.

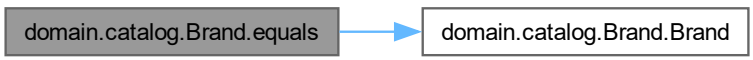
**Paramètres**

| | |
|---|---|
| *o* | objet à comparer |

**Renvoie**

true si les identifiants sont identiques, false sinon

Voici le graphe d'appel pour cette fonction :



#### 8.2.3.2 getBrandId()

```
String domain.catalog.Brand.getBrandId ()
```

Retourne l'identifiant unique de la marque.

**Renvoie**

identifiant de la marque

#### 8.2.3.3 getName()

```
String domain.catalog.Brand.getName ()
```

Retourne le nom de la marque.

**Renvoie**

nom de la marque

#### 8.2.3.4 hashCode()

```
int domain.catalog.Brand.hashCode ()
```

Calcule le hashcode basé sur l'identifiant de la marque.

**Renvoie**

valeur de hachage

#### 8.2.3.5 toString()

```
String domain.catalog.Brand.toString ()
```

Fournit une représentation textuelle lisible de la marque.

**Renvoie**

chaîne descriptive de la marque

La documentation de cette classe a été générée à partir du fichier suivant :

— src/domain/catalog/Brand.java

## 8.3 Référence de la classe domain.catalog.Catalog

Fournit un registre centralisé pour la gestion des marques (`Brand`) et des gammes de produits (`ProductLine`).

**Fonctions membres publiques**

- Brand createBrand (String id, String name)

  *Crée une marque si elle n'existe pas encore dans le registre.*

- ProductLine createProductLine (String lineId, String name, Brand brand)

  *Crée une gamme de produits rattachée à une marque si elle n'existe pas déjà.*

- List< Brand > getAllBrands ()

  *Retourne toutes les marques actuellement enregistrées dans le catalogue.*

- List< ProductLine > getAllProductLines ()

  *Retourne toutes les gammes de produits enregistrées.*

- Optional< Brand > findBrandById (String id)

  *Recherche une marque à partir de son identifiant.*

- Optional< ProductLine > findProductLineById (String id)

  *Recherche une gamme à partir de son identifiant.*

- List< ProductLine > findLinesByBrand (Brand brand)

  *Retourne toutes les gammes associées à une marque donnée.*

## 8.3.1 Description détaillée

Fournit un registre centralisé pour la gestion des marques (`Brand`) et des gammes de produits (`ProductLine`).

Ce composant encapsule la logique de création, d'accès et de regroupement des entités métier liées au catalogue produit.

**Auteur**

Lucas

**Version**

1.2

## 8.3.2 Documentation des fonctions membres

### 8.3.2.1 createBrand()

```
Brand domain.catalog.Catalog.createBrand (
            String id,
            String name)
```

Crée une marque si elle n'existe pas encore dans le registre.

Deux marques avec le même identifiant sont considérées comme identiques. Cette méthode est idempotente.

**Paramètres**

| | |
|---|---|
| *id* | identifiant unique de la marque |
| *name* | nom lisible de la marque |

**Renvoie**

la marque existante ou nouvellement créée

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si l'identifiant est nul |

#### 8.3.2.2 createProductLine()

```
ProductLine domain.catalog.Catalog.createProductLine (
            String lineId,
            String name,
            Brand brand)
```

Crée une gamme de produits rattachée à une marque si elle n'existe pas déjà.

Deux gammes avec le même identifiant sont considérées comme identiques. Cette méthode est idempotente.

**Paramètres**

| | |
|---|---|
| *lineId* | identifiant unique de la gamme |
| *name* | nom de la gamme |
| *brand* | marque associée |

**Renvoie**

la gamme existante ou nouvellement créée

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si l'identifiant ou la marque sont nuls |

#### 8.3.2.3 findBrandById()

```
Optional< Brand > domain.catalog.Catalog.findBrandById (
            String id)
```

Recherche une marque à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *id* | identifiant de la marque |

**Renvoie**

un Optional contenant la marque si elle est trouvée

#### 8.3.2.4 findLinesByBrand()

```
List< ProductLine > domain.catalog.Catalog.findLinesByBrand (
            Brand brand)
```

Retourne toutes les gammes associées à une marque donnée.

**Paramètres**

| | |
|---|---|
| *brand* | instance de la marque cible |

**Renvoie**

liste des gammes rattachées à cette marque

#### 8.3.2.5 findProductLineById()

```
Optional< ProductLine > domain.catalog.Catalog.findProductLineById (
            String id)
```

Recherche une gamme à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *id* | identifiant de la gamme |

**Renvoie**

un Optional contenant la gamme si elle est trouvée

#### 8.3.2.6 getAllBrands()

```
List< Brand > domain.catalog.Catalog.getAllBrands ()
```

Retourne toutes les marques actuellement enregistrées dans le catalogue.

**Renvoie**

liste des marques

#### 8.3.2.7 getAllProductLines()

```
List< ProductLine > domain.catalog.Catalog.getAllProductLines ()
```

Retourne toutes les gammes de produits enregistrées.

**Renvoie**

liste des gammes

La documentation de cette classe a été générée à partir du fichier suivant :

— src/domain/catalog/Catalog.java

## 8.4 Référence de la classe presentation.ConsoleInterface

Interface utilisateur en ligne de commande pour la gestion de stock.

**Fonctions membres publiques**

— void start ()

*Lance le menu principal en boucle jusqu'à la demande de sortie.*

### 8.4.1 Description détaillée

Interface utilisateur en ligne de commande pour la gestion de stock.

Permet l'ajout, la suppression, la recherche, la vente et l'affichage de produits. Sert de point d'entrée principal de la couche de présentation.

Cette classe orchestre les cas d'utilisation métier à partir des saisies utilisateur.

**Auteur**

Lucas

**Version**

1.1

### 8.4.2 Documentation des fonctions membres

#### 8.4.2.1 start()

```
void presentation.ConsoleInterface.start ()
```

Lance le menu principal en boucle jusqu'à la demande de sortie.

Voici le graphe des appelants de cette fonction :

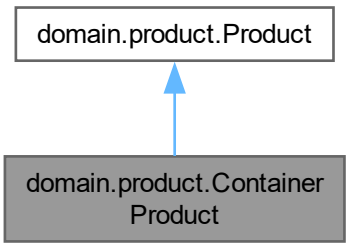

La documentation de cette classe a été générée à partir du fichier suivant :

— src/presentation/ConsoleInterface.java

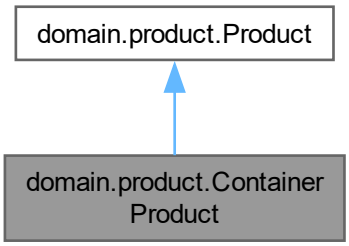
## 8.5 Référence de la classe domain.product.ContainerProduct

Représente un produit conteneur, c'est-à-dire un produit composé qui contient une certaine quantité d'un autre produit.

Graphe d'héritage de domain.product.ContainerProduct:



Graphe de collaboration de domain.product.ContainerProduct:




### Fonctions membres publiques

- ContainerProduct (String id, String name, double price, int stock, Product containedProduct, int contained↩ Quantity, Brand brand, ProductLine productLine)

  *Construit un produit conteneur avec son contenu et ses attributs de base.*

- Product getContainedProduct ()

  *Retourne le produit contenu dans ce conteneur.*

- int getContainedQuantity ()

  *Retourne la quantité du produit contenu.*

## Fonctions membres publiques hérités de domain.product.Product

- Product (String id, String name, double price, int stock, Brand brand, ProductLine productLine)

  *Construit un produit générique avec validation des contraintes métier.*

- String getProductId ()

  *Retourne l'identifiant unique du produit.*

- String getName ()

  *Retourne le nom du produit.*

- double getPrice ()

  *Retourne le prix unitaire du produit.*

- int getStock ()

  *Retourne la quantité en stock.*

- Brand getBrand ()

*Retourne la marque associée au produit.*

— ProductLine getProductLine ()

*Retourne la gamme à laquelle appartient le produit.*

— String toString ()

### 8.5.1 Description détaillée

Représente un produit conteneur, c'est-à-dire un produit composé qui contient une certaine quantité d'un autre produit.

Ce type de produit est utile pour les lots, packs ou combinaisons de produits.

Hérite des propriétés de base d'un `Product`.

**Auteur**

Lucas

**Version**

1.1

### 8.5.2 Documentation des constructeurs et destructeur

#### 8.5.2.1 ContainerProduct()

```
domain.product.ContainerProduct.ContainerProduct (
            String id,
            String name,
            double price,
            int stock,
            Product containedProduct,
            int containedQuantity,
            Brand brand,
            ProductLine productLine)
```

Construit un produit conteneur avec son contenu et ses attributs de base.

**Paramètres**

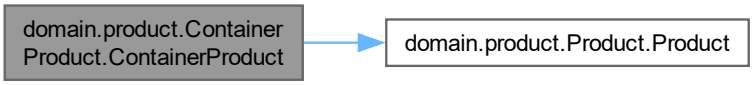
| | |
|---|---|
| *id* | identifiant unique du conteneur |
| *name* | nom du conteneur |
| *price* | prix du conteneur (peut être différent du produit contenu) |
| *stock* | quantité en stock du conteneur |
| *containedProduct* | produit contenu (ne peut pas être nul) |
| *containedQuantity* | quantité du produit contenu (strictement positive) |
| *brand* | marque associée |

| | |
|---|---|
| *productLine* | gamme associée |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si le produit contenu est nul ou si la quantité est invalide |

Voici le graphe d'appel pour cette fonction :




### 8.5.3 Documentation des fonctions membres

#### 8.5.3.1 getContainedProduct()

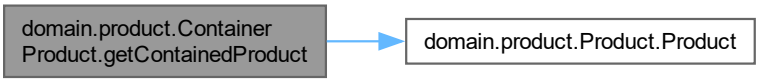
```
Product domain.product.ContainerProduct.getContainedProduct ()
```

Retourne le produit contenu dans ce conteneur.

**Renvoie**

produit encapsulé

Voici le graphe d'appel pour cette fonction :




#### 8.5.3.2 getContainedQuantity()

```
int domain.product.ContainerProduct.getContainedQuantity ()
```

Retourne la quantité du produit contenu.

**Renvoie**

quantité encapsulée

La documentation de cette classe a été générée à partir du fichier suivant :
— src/domain/product/ContainerProduct.java

## 8.6 Référence de la classe application.DeleteProductUseCase

Cas d'usage pour la suppression d'un produit du stock.

#### Fonctions membres publiques

— DeleteProductUseCase (ProductRepository productRepository)

*Construit le cas d'usage avec le référentiel produit injecté.*

— void execute (String productId)

*Supprime le produit correspondant à l'identifiant spécifié.*

### 8.6.1 Description détaillée

Cas d'usage pour la suppression d'un produit du stock.

Cette classe encapsule la logique métier liée à la suppression d'un produit existant. Elle effectue les validations nécessaires avant d'interagir avec le dépôt de persistance.

Ce use case appartient à la couche application et dépend du port `ProductRepository`.

**Auteur**

Lucas

**Version**

1.2

### 8.6.2 Documentation des constructeurs et destructeur

#### 8.6.2.1 DeleteProductUseCase()

```
application.DeleteProductUseCase.DeleteProductUseCase (
            ProductRepository productRepository)
```

Construit le cas d'usage avec le référentiel produit injecté.

**Paramètres**

| | |
|---|---|
| *productRepository* | le dépôt produit à utiliser pour les opérations |

### 8.6.3 Documentation des fonctions membres

#### 8.6.3.1 execute()

```
void application.DeleteProductUseCase.execute (
            String productId)
```

Supprime le produit correspondant à l'identifiant spécifié.

Avant suppression, vérifie que l'identifiant est valide et que le produit existe réellement.

**Paramètres**

| | |
|---|---|
| *productId* | identifiant unique du produit à supprimer |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si l'ID est nul, vide ou que le produit n'existe pas |

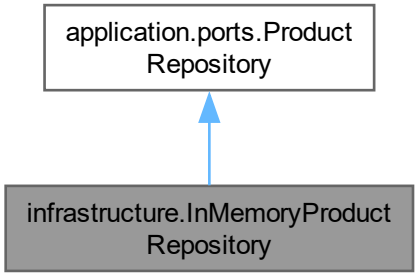
La documentation de cette classe a été générée à partir du fichier suivant :

— src/application/DeleteProductUseCase.java

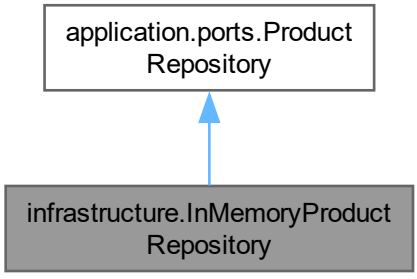
## 8.7 Référence de la classe infrastructure.InMemoryProductRepository

Implémentation en mémoire de l'interface `ProductRepository`.

Graphe d'héritage de infrastructure.InMemoryProductRepository:




Graphe de collaboration de infrastructure.InMemoryProductRepository:



**Fonctions membres publiques**

— void save (Product product)

*Enregistre ou met à jour un produit dans le stockage.*

— boolean existsById (String productId)

*Vérifie si un produit existe à partir de son identifiant.*

— Product findById (String productId)

*Recherche un produit par son identifiant.*

— List< Product > findAll ()

*Retourne l'ensemble des produits enregistrés.*

— List< Product > search (String keyword)

*Recherche les produits contenant le mot-clé dans leur nom.*

— void deleteById (String productId)

*Supprime un produit du stockage à partir de son identifiant.*

## 8.7.1 Description détaillée

Implémentation en mémoire de l'interface `ProductRepository`.

Cette classe fournit une solution de persistance temporaire, idéale pour les phases de test, de prototypage ou d'exécution sans base de données.

Les données sont stockées dans une Map et sont perdues à l'arrêt de l'application.

**Auteur**

Lucas

**Version**

1.1

## 8.7.2 Documentation des fonctions membres

### 8.7.2.1 deleteById()

```
void infrastructure.InMemoryProductRepository.deleteById (
            String productId)
```

Supprime un produit du stockage à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | identifiant du produit à supprimer |

Implémente application.ports.ProductRepository.

#### 8.7.2.2 existsById()

```
boolean infrastructure.InMemoryProductRepository.existsById (
            String productId)
```

Vérifie si un produit existe à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | identifiant du produit |

**Renvoie**

true si le produit existe, false sinon

Implémente application.ports.ProductRepository.

#### 8.7.2.3 findAll()

```
List< Product > infrastructure.InMemoryProductRepository.findAll ()
```

Retourne l'ensemble des produits enregistrés.

**Renvoie**

liste de tous les produits

Implémente application.ports.ProductRepository.

#### 8.7.2.4 findById()

```
Product infrastructure.InMemoryProductRepository.findById (
            String productId)
```

Recherche un produit par son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | identifiant du produit |

**Renvoie**

le produit correspondant, ou null si absent

Implémente application.ports.ProductRepository.

#### 8.7.2.5 save()

```
void infrastructure.InMemoryProductRepository.save (
            Product product)
```

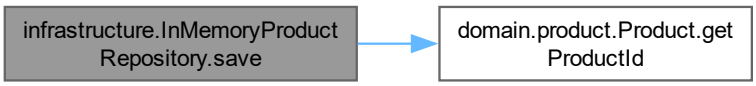
Enregistre ou met à jour un produit dans le stockage.

**Paramètres**

| | |
|---|---|
| *product* | produit à sauvegarder |

Implémente application.ports.ProductRepository.

Voici le graphe d'appel pour cette fonction :




#### 8.7.2.6 search()

```
List< Product > infrastructure.InMemoryProductRepository.search (
            String keyword)
```

Recherche les produits contenant le mot-clé dans leur nom.

La recherche est insensible à la casse.

**Paramètres**

| | |
|---|---|
| *keyword* | mot-clé à rechercher |

**Renvoie**

liste des produits correspondants

Implémente application.ports.ProductRepository.

La documentation de cette classe a été générée à partir du fichier suivant :
— src/infrastructure/InMemoryProductRepository.java

## 8.8 Référence de la classe Main

Point d'entrée de l'application de gestion de stock.

**Fonctions membres publiques statiques**

— static void main (String[ ] args)

*Méthode principale du programme.*

### 8.8.1 Description détaillée

Point d'entrée de l'application de gestion de stock.

Lance l'interface en ligne de commande permettant les opérations de consultation et de modification du catalogue produit.

Cette classe initialise le contrôleur principal. Elle pourra également servir de point d'ancrage à d'autres interfaces futures (web, graphique, API, etc.).

**Auteur**

Lucas

**Version**

1.0

### 8.8.2 Documentation des fonctions membres

#### 8.8.2.1 main()

```
void Main.main (
            String[] args)  [static]
```

Méthode principale du programme.

Initialise l'interface console et démarre la boucle interactive.

**Paramètres**

| | |
|---|---|
| *args* | arguments passés en ligne de commande (non utilisés) |

Voici le graphe d'appel pour cette fonction :


La documentation de cette classe a été générée à partir du fichier suivant :
— src/Main.java

## 8.9 application.SearchProductUseCase.Mode Référence de l'énumération

Modes de recherche disponibles.

**Attributs publics**

- ID
- NAME
- BOTH

### 8.9.1 Description détaillée

Modes de recherche disponibles.

### 8.9.2 Documentation des données membres

#### 8.9.2.1 BOTH

```
application.SearchProductUseCase.Mode.BOTH
```

#### 8.9.2.2 ID

```
application.SearchProductUseCase.Mode.ID
```

#### 8.9.2.3 NAME

```
application.SearchProductUseCase.Mode.NAME
```

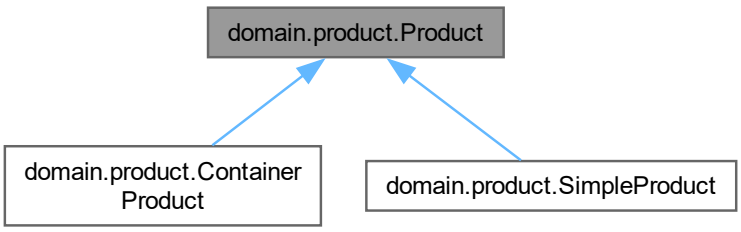
La documentation pour cette énumération a été générée à partir des fichiers suivants :

- src/application/SearchProductUseCase.java

## 8.10 Référence de la classe domain.product.Product

Représente un produit abstrait appartenant à un catalogue.

Graphe d'héritage de domain.product.Product:



**Fonctions membres publiques**

- Product (String id, String name, double price, int stock, Brand brand, ProductLine productLine)

  *Construit un produit générique avec validation des contraintes métier.*

- String getProductId ()

  *Retourne l'identifiant unique du produit.*

- String getName ()

  *Retourne le nom du produit.*

- double getPrice ()

  *Retourne le prix unitaire du produit.*

- int getStock ()

  *Retourne la quantité en stock.*

- Brand getBrand ()

  *Retourne la marque associée au produit.*

- ProductLine getProductLine ()

  *Retourne la gamme à laquelle appartient le produit.*

- String toString ()

### 8.10.1 Description détaillée

Représente un produit abstrait appartenant à un catalogue.

Cette classe sert de base pour les types de produits concrets (ex: SimpleProduct). Elle encapsule les propriétés communes à tous les produits : identifiant, nom, prix, stock, ainsi que les informations de marque et de gamme.

Cette classe est abstraite et ne peut être instanciée directement.

**Auteur**

Lucas

**Version**

1.1

### 8.10.2 Documentation des constructeurs et destructeur

#### 8.10.2.1 Product()

```
domain.product.Product.Product (
            String id,
```

```
String name,
double price,
int stock,
Brand brand,
ProductLine productLine)
```

Construit un produit générique avec validation des contraintes métier.

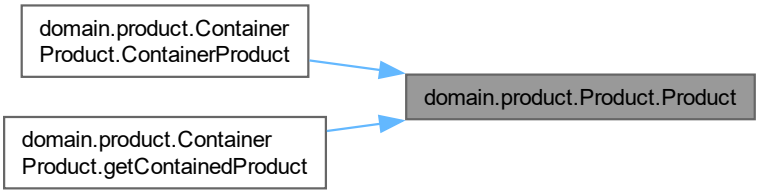
**Paramètres**

| | |
|---|---|
| *id* | identifiant unique du produit (non nul) |
| *name* | nom du produit (non nul) |
| *price* | prix unitaire du produit (>= 0) |
| *stock* | quantité disponible en stock (>= 0) |
| *brand* | marque à laquelle appartient le produit (non nulle) |
| *productLine* | gamme à laquelle appartient le produit (non nulle) |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si un paramètre est invalide |

Voici le graphe des appelants de cette fonction :




## 8.10.3 Documentation des fonctions membres

### 8.10.3.1 getBrand()

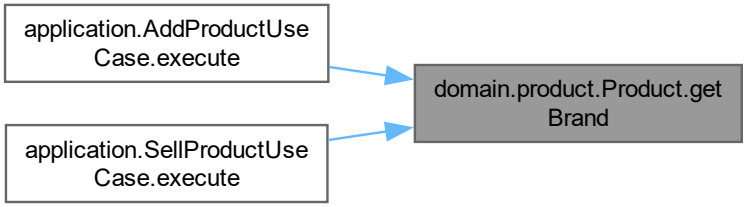
```
Brand domain.product.Product.getBrand ()
```

Retourne la marque associée au produit.

**Renvoie**

marque

Voici le graphe des appelants de cette fonction :




#### 8.10.3.2 getName()

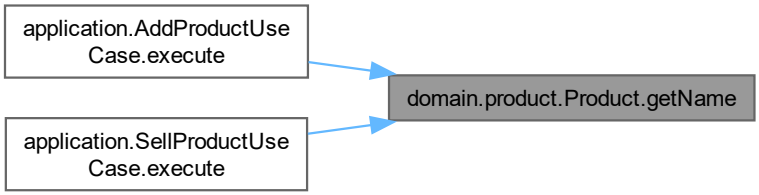
```
String domain.product.Product.getName ()
```

Retourne le nom du produit.

**Renvoie**

nom du produit

Voici le graphe des appelants de cette fonction :




#### 8.10.3.3 getPrice()

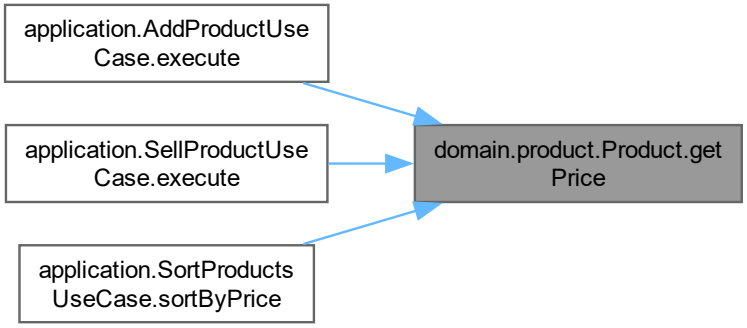
```
double domain.product.Product.getPrice ()
```

Retourne le prix unitaire du produit.

**Renvoie**

prix du produit

Voici le graphe des appelants de cette fonction :




#### 8.10.3.4 getProductId()

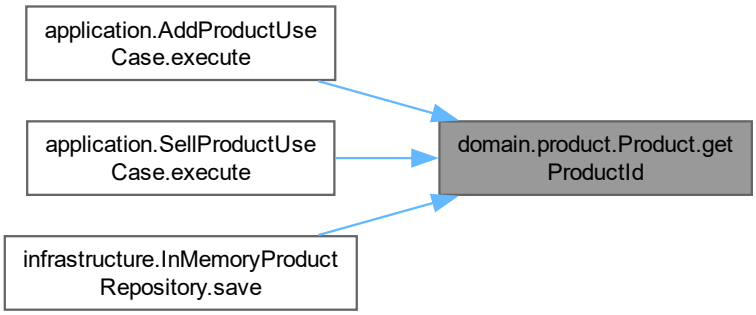
```
String domain.product.Product.getProductId ()
```

Retourne l'identifiant unique du produit.

**Renvoie**

identifiant du produit

Voici le graphe des appelants de cette fonction :



#### 8.10.3.5 getProductLine()

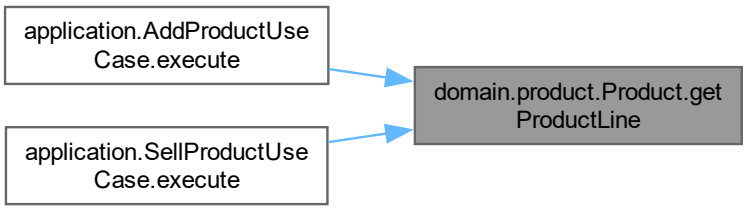
```
ProductLine domain.product.Product.getProductLine ()
```

Retourne la gamme à laquelle appartient le produit.

**Renvoie**

gamme

Voici le graphe des appelants de cette fonction :




#### 8.10.3.6 getStock()

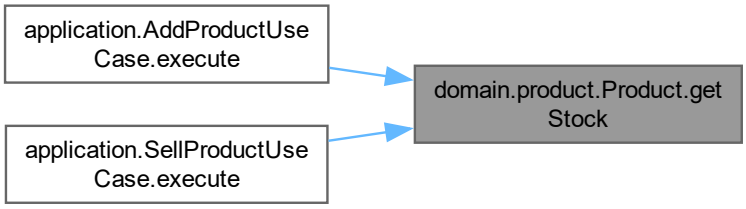
```
int domain.product.Product.getStock ()
```

Retourne la quantité en stock.

**Renvoie**

stock actuel

Voici le graphe des appelants de cette fonction :



#### 8.10.3.7 toString()

```
String domain.product.Product.toString ()
```

La documentation de cette classe a été générée à partir du fichier suivant :

— src/domain/product/Product.java

## 8.11 Référence de la classe infrastructure.ProductIdGenerator

Générateur d'identifiants uniques pour les produits.

**Fonctions membres publiques statiques**

— static synchronized String generateId ()

*Génère un nouvel identifiant unique sous forme de chaîne.*

— static synchronized void initializeFrom (int lastUsedId)

*Initialise manuellement l'état interne du générateur.*

### 8.11.1 Description détaillée

Générateur d'identifiants uniques pour les produits.

Garantit l'unicité des IDs via un incrément interne.

Utilisé pour éviter la duplication d'identifiants produits lors de l'ajout de nouveaux éléments au stock.

Ce générateur est thread-safe via synchronisation explicite.

**Auteur**

Lucas

**Version**

1.1

### 8.11.2 Documentation des fonctions membres

#### 8.11.2.1 generateId()

```
synchronized String infrastructure.ProductIdGenerator.generateId ()  [static]
```

Génère un nouvel identifiant unique sous forme de chaîne.

**Renvoie**

identifiant produit incrémental (ex: "1", "2", "3", ...)

#### 8.11.2.2 initializeFrom()

```
synchronized void infrastructure.ProductIdGenerator.initializeFrom (
            int lastUsedId)  [static]
```

Initialise manuellement l'état interne du générateur.

Permet d'éviter les collisions si des produits préexistants sont rechargés depuis une base externe ou un fichier.

**Paramètres**

| | |
|---|---|
| *lastUsedId* | identifiant le plus élevé actuellement utilisé |

La documentation de cette classe a été générée à partir du fichier suivant :
— src/infrastructure/ProductIdGenerator.java

## 8.12 Référence de la classe domain.catalog.ProductLine

Représente une gamme de produits.

#### Fonctions membres publiques

— ProductLine (String lineId, String name, Brand brand)

*Construit une nouvelle gamme de produits associée à une marque.*

— String getLineId ()

*Retourne l'identifiant de la gamme.*

— String getName ()

*Retourne le nom de la gamme.*

— Brand getBrand ()

*Retourne la marque associée à la gamme.*

— boolean equals (Object o)

— int hashCode ()

— String toString ()

### 8.12.1 Description détaillée

Représente une gamme de produits.

Chaque gamme est liée à une marque unique et possède un identifiant et un nom. Les objets ProductLine sont considérés comme égaux s'ils partagent le même identifiant.

### 8.12.2 Documentation des constructeurs et destructeur

#### 8.12.2.1 ProductLine()

```
domain.catalog.ProductLine.ProductLine (
            String lineId,
            String name,
            Brand brand)
```

Construit une nouvelle gamme de produits associée à une marque.

**Paramètres**

| | |
|---|---|
| *lineId* | identifiant unique de la gamme |
| *name* | nom de la gamme |
| *brand* | marque associée à cette gamme |

Voici le graphe des appelants de cette fonction :


### 8.12.3 Documentation des fonctions membres

#### 8.12.3.1 equals()

```
boolean domain.catalog.ProductLine.equals (
            Object o)
```

Voici le graphe d'appel pour cette fonction :


#### 8.12.3.2 getBrand()

```
Brand domain.catalog.ProductLine.getBrand ()
```

Retourne la marque associée à la gamme.

**Renvoie**

objet Brand lié à cette gamme

#### 8.12.3.3 getLineId()

```
String domain.catalog.ProductLine.getLineId ()
```

Retourne l'identifiant de la gamme.

**Renvoie**

identifiant de la gamme

#### 8.12.3.4 getName()

```
String domain.catalog.ProductLine.getName ()
```

Retourne le nom de la gamme.

**Renvoie**

nom de la gamme

#### 8.12.3.5 hashCode()

```
int domain.catalog.ProductLine.hashCode ()
```

#### 8.12.3.6 toString()

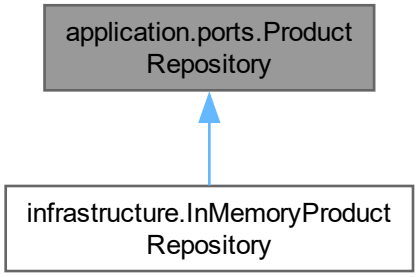
```
String domain.catalog.ProductLine.toString ()
```

La documentation de cette classe a été générée à partir du fichier suivant :
— src/domain/catalog/ProductLine.java

## 8.13 Référence de l'interface application.ports.ProductRepository

Interface représentant le port d'accès aux données des produits.

Graphe d'héritage de application.ports.ProductRepository:



**Fonctions membres publiques**

— void save (Product product)

*Sauvegarde un produit dans le référentiel.*

— boolean existsById (String productId)

*Vérifie si un produit existe à partir de son identifiant.*

— Product findById (String productId)

*Recherche un produit par son identifiant.*

— List< Product > findAll ()

*Récupère la liste complète des produits disponibles.*

— List< Product > search (String keyword)

*Recherche les produits correspondant à un mot-clé.*

— void deleteById (String productId)

*Supprime un produit à partir de son identifiant.*

## 8.13.1 Description détaillée

Interface représentant le port d'accès aux données des produits.

Définit les opérations de persistance nécessaires pour interagir avec la couche infrastructure sans en dépendre directement. Ce port permet une architecture découplée, facilitant les tests, l'évolution et le changement de technologies de stockage.

## 8.13.2 Documentation des fonctions membres

### 8.13.2.1 deleteById()

```
void application.ports.ProductRepository.deleteById (
            String productId)
```

Supprime un produit à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | L'identifiant du produit à supprimer. |

Implémenté dans infrastructure.InMemoryProductRepository.

#### 8.13.2.2 existsById()

```
boolean application.ports.ProductRepository.existsById (
            String productId)
```

Vérifie si un produit existe à partir de son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | L'identifiant unique du produit. |

**Renvoie**

true si le produit existe, false sinon.

Implémenté dans infrastructure.InMemoryProductRepository.

#### 8.13.2.3 findAll()

```
List< Product > application.ports.ProductRepository.findAll ()
```

Récupère la liste complète des produits disponibles.

**Renvoie**

Une liste de tous les produits.

Implémenté dans infrastructure.InMemoryProductRepository.

#### 8.13.2.4 findById()

```
Product application.ports.ProductRepository.findById (
            String productId)
```

Recherche un produit par son identifiant.

**Paramètres**

| | |
|---|---|
| *productId* | L'identifiant du produit recherché. |

**Renvoie**

Le produit correspondant, ou null s'il n'existe pas.

Implémenté dans infrastructure.InMemoryProductRepository.

#### 8.13.2.5 save()

```
void application.ports.ProductRepository.save (
            Product product)
```

Sauvegarde un produit dans le référentiel.

**Paramètres**

| | |
|---|---|
| *product* | Le produit à enregistrer. |

Implémenté dans infrastructure.InMemoryProductRepository.

#### 8.13.2.6 search()

```
List< Product > application.ports.ProductRepository.search (
            String keyword)
```

Recherche les produits correspondant à un mot-clé.

La recherche peut s'appliquer sur le nom, l'identifiant ou d'autres critères selon l'implémentation.

**Paramètres**

| | |
|---|---|
| *keyword* | Le mot-clé à utiliser pour la recherche. |

**Renvoie**

Une liste de produits correspondant au critère.

Implémenté dans infrastructure.InMemoryProductRepository.

La documentation de cette interface a été générée à partir du fichier suivant :

— src/application/ports/ProductRepository.java

## 8.14 Référence de la classe application.SearchProductUseCase

Cas d'usage : Rechercher des produits dans le stock.

### Classes

— enum Mode

*Modes de recherche disponibles.*

### Fonctions membres publiques

— SearchProductUseCase (ProductRepository productRepository)

— List< Product > execute (String keyword, Mode mode)

*Recherche les produits selon le mode spécifié.*

### 8.14.1 Description détaillée

Cas d'usage : Rechercher des produits dans le stock.

Permet une recherche ciblée selon l'identifiant, le nom ou les deux. Recherche insensible à la casse et robuste aux chaînes vides.

**Auteur**

Lucas

**Version**

1.2

### 8.14.2 Documentation des constructeurs et destructeur

#### 8.14.2.1 SearchProductUseCase()

```
application.SearchProductUseCase.SearchProductUseCase (
            ProductRepository productRepository)
```

### 8.14.3 Documentation des fonctions membres

#### 8.14.3.1 execute()

```
List< Product > application.SearchProductUseCase.execute (
            String keyword,
            Mode mode)
```

Recherche les produits selon le mode spécifié.

Si aucun mot-clé valide n'est fourni, retourne l'intégralité des produits.

**Paramètres**

| | |
|---|---|
| *keyword* | Mot-clé utilisé pour filtrer les produits |
| *mode* | Mode de recherche (ID, NAME ou BOTH) |

**Renvoie**

Liste filtrée des produits correspondant au critère

La documentation de cette classe a été générée à partir du fichier suivant :
— src/application/SearchProductUseCase.java

## 8.15 Référence de la classe application.SellProductUseCase

Cas d'usage : Vendre un produit du stock.

**Fonctions membres publiques**

— SellProductUseCase (ProductRepository productRepository)

*Constructeur du cas d'utilisation.*

— void execute (String productId, int quantity)

*Exécute la vente d'un produit en diminuant son stock.*

## 8.15.1 Description détaillée

Cas d'usage : Vendre un produit du stock.

Ce cas d'utilisation vérifie que le produit existe, que la quantité demandée est disponible, et met à jour le stock après la vente.

Ce comportement est atomique et garantit l'intégrité du stock.

**Auteur**

Lucas

**Version**

1.2

## 8.15.2 Documentation des constructeurs et destructeur

### 8.15.2.1 SellProductUseCase()

```
application.SellProductUseCase.SellProductUseCase (
            ProductRepository productRepository)
```

Constructeur du cas d'utilisation.

**Paramètres**

| | |
|---|---|
| *productRepository* | Référentiel des produits (port d'accès aux données) |

## 8.15.3 Documentation des fonctions membres

### 8.15.3.1 execute()

```
void application.SellProductUseCase.execute (
            String productId,
            int quantity)
```

Exécute la vente d'un produit en diminuant son stock.

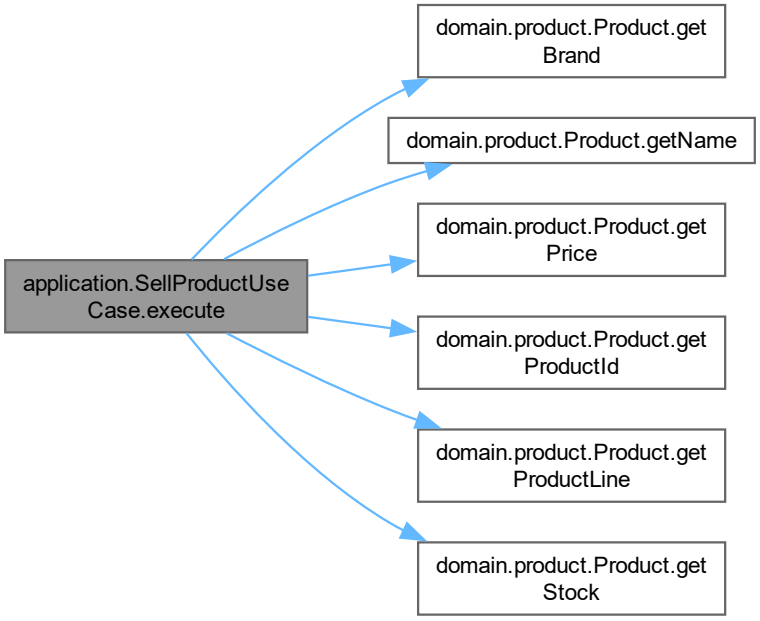
**Paramètres**

| | |
|---|---|
| *productId* | Identifiant du produit à vendre |
| *quantity* | Quantité à vendre (doit être strictement positive) |

**Exceptions**

| | |
|---|---|
| *IllegalArgumentException* | si l'identifiant est inconnu, si le stock est insuffisant ou si la quantité demandée est invalide |

Voici le graphe d'appel pour cette fonction :



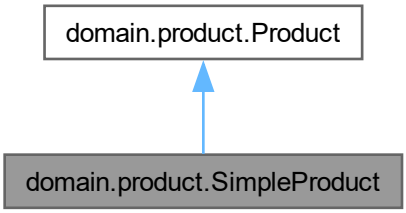

La documentation de cette classe a été générée à partir du fichier suivant :

— src/application/SellProductUseCase.java

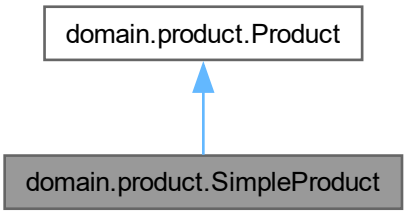
## 8.16 Référence de la classe domain.product.SimpleProduct

Représente un produit simple du catalogue.

Graphe d'héritage de domain.product.SimpleProduct:




Graphe de collaboration de domain.product.SimpleProduct:




### Fonctions membres publiques

— SimpleProduct (String id, String name, double price, int stock, Brand brand, ProductLine productLine)

*Construit une instance de produit simple avec ses attributs fondamentaux.*

## Fonctions membres publiques hérités de domain.product.Product

— Product (String id, String name, double price, int stock, Brand brand, ProductLine productLine)

*Construit un produit générique avec validation des contraintes métier.*

— String getProductId ()

*Retourne l'identifiant unique du produit.*

— String getName ()

*Retourne le nom du produit.*

— double getPrice ()

*Retourne le prix unitaire du produit.*

— int getStock ()

*Retourne la quantité en stock.*

— Brand getBrand ()

*Retourne la marque associée au produit.*

— ProductLine getProductLine ()

*Retourne la gamme à laquelle appartient le produit.*

— String toString ()

## 8.16.1 Description détaillée

Représente un produit simple du catalogue.

Il s'agit d'un produit unitaire, non composite, dérivé de la classe abstraite `Product`. Ce type de produit ne contient ni sous-produits ni structure hiérarchique.

Cette classe est utilisée pour les cas standards de gestion de stock.

**Auteur**

Lucas

**Version**

1.1

## 8.16.2 Documentation des constructeurs et destructeur

### 8.16.2.1 SimpleProduct()

```
domain.product.SimpleProduct.SimpleProduct (
            String id,
            String name,
            double price,
            int stock,
            Brand brand,
            ProductLine productLine)
```

Construit une instance de produit simple avec ses attributs fondamentaux.

**Paramètres**

| | |
|---|---|
| *id* | identifiant unique du produit |

| | |
|---|---|
| *name* | nom du produit |
| *price* | prix unitaire (>= 0) |
| *stock* | quantité disponible (>= 0) |
| *brand* | marque associée au produit (non nulle) |
| *productLine* | gamme à laquelle appartient le produit (non nulle) |

La documentation de cette classe a été générée à partir du fichier suivant :
— src/domain/product/SimpleProduct.java

## 8.17 Référence de la classe application.SortProductsUseCase

Cas d'usage : Trier les produits du stock par ordre croissant de prix.

### Fonctions membres publiques

— SortProductsUseCase (ProductRepository repository)

*Initialise le cas d'utilisation avec le référentiel des produits.*

— List< Product > sortByPrice ()

*Trie les produits en mémoire par prix croissant.*

— List< Product > sortByPriceSelectionSort ()

*Variante manuelle du tri utilisant l'algorithme de tri par sélection.*

### 8.17.1 Description détaillée

Cas d'usage : Trier les produits du stock par ordre croissant de prix.

Ce cas d'utilisation propose deux approches :
- Tri rapide via l'API Java standard (Comparator + sort)
- Tri manuel via l'algorithme de tri par sélection (à des fins pédagogiques ou de benchmarking)

Cette classe n'effectue aucun tri en place dans le dépôt : elle retourne une nouvelle liste triée.

**Auteur**

Lucas

**Version**

1.1

### 8.17.2 Documentation des constructeurs et destructeur

#### 8.17.2.1 SortProductsUseCase()

```
application.SortProductsUseCase.SortProductsUseCase (
            ProductRepository repository)
```

Initialise le cas d'utilisation avec le référentiel des produits.

**Paramètres**

| | |
|---|---|
| *repository* | Interface d'accès aux produits |

### 8.17.3 Documentation des fonctions membres

#### 8.17.3.1 sortByPrice()

```
List< Product > application.SortProductsUseCase.sortByPrice ()
```

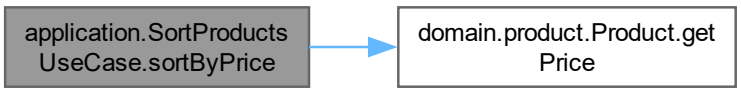
Trie les produits en mémoire par prix croissant.

Utilise `java.util.Collections#sort(List, java.util.Comparator)` pour des performances optimales.

**Renvoie**

liste de produits triée par prix

Voici le graphe d'appel pour cette fonction :




#### 8.17.3.2 sortByPriceSelectionSort()

```
List< Product > application.SortProductsUseCase.sortByPriceSelectionSort ()
```

Variante manuelle du tri utilisant l'algorithme de tri par sélection.

Moins efficace que `sortByPrice()`, mais utile pour l'analyse de performance ou démonstration.

**Renvoie**

liste triée par prix en utilisant un algorithme bas-niveau

La documentation de cette classe a été générée à partir du fichier suivant :

— src/application/SortProductsUseCase.java

## 8.18 Référence de la classe test.TestRunner

Classe de test manuelle de l'application de gestion de stock.

**Fonctions membres publiques statiques**

— static void main (String[ ] args)

*Point d'entrée principal pour exécuter les tests unitaires manuels.*

## 8.18.1 Description détaillée

Classe de test manuelle de l'application de gestion de stock.

Contient un ensemble de tests unitaires simples exécutés depuis la ligne de commande, sans utilisation de framework externe.

Cette classe couvre :

- La création de produits
- L'ajout, la recherche et la suppression de produits
- La génération automatique d'identifiants
- La recherche par nom

**Auteur**

Lucas

**Version**

1.0

## 8.18.2 Documentation des fonctions membres

### 8.18.2.1 main()

```
void test.TestRunner.main (
            String[] args)  [static]
```

Point d'entrée principal pour exécuter les tests unitaires manuels.

Chaque méthode de test est appelée et une confirmation s'affiche si elle passe.

La documentation de cette classe a été générée à partir du fichier suivant :

— src/test/TestRunner.java

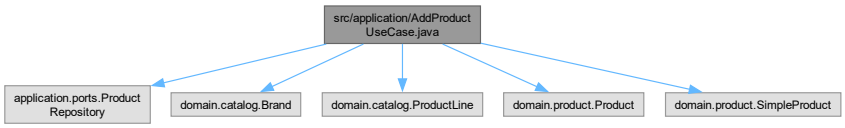
# Chapitre 9

# Documentation des fichiers

## 9.1 Référence du fichier src/application/AddProductUseCase.java

```
import application.ports.ProductRepository;
import domain.catalog.Brand;
import domain.catalog.ProductLine;
import domain.product.Product;
import domain.product.SimpleProduct;
```

Graphe des dépendances par inclusion de AddProductUseCase.java:




### Classes

— class application.AddProductUseCase

*Use case responsable de l'ajout d'un produit dans le stock.*

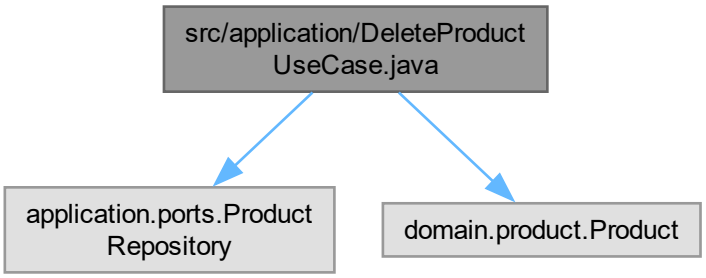
### Paquetages

— package application

## 9.2 Référence du fichier src/application/DeleteProductUseCase.java

```
import application.ports.ProductRepository;
import domain.product.Product;
```

Graphe des dépendances par inclusion de DeleteProductUseCase.java:




### Classes

— class application.DeleteProductUseCase

*Cas d'usage pour la suppression d'un produit du stock.*

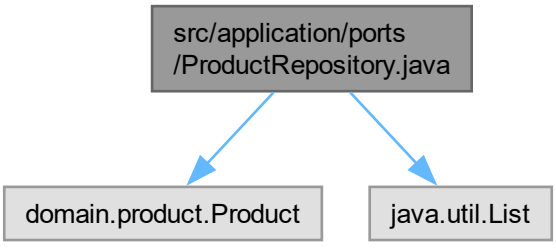
### Paquetages

— package application

## 9.3 Référence du fichier src/application/ports/ProductRepository.java

```
import domain.product.Product;
import java.util.List;
```

Graphe des dépendances par inclusion de ProductRepository.java:



### Classes

— interface application.ports.ProductRepository

*Interface représentant le port d'accès aux données des produits.*

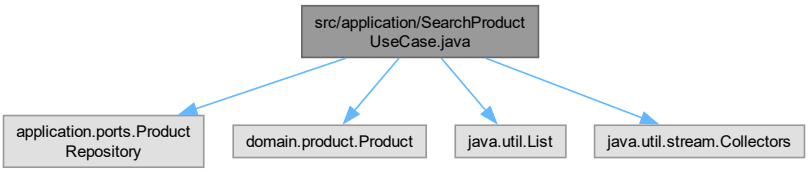
### Paquetages

— package application.ports

## 9.4 Référence du fichier src/application/SearchProductUseCase.java

```
import application.ports.ProductRepository;
import domain.product.Product;
import java.util.List;
import java.util.stream.Collectors;
```

Graphe des dépendances par inclusion de SearchProductUseCase.java:




### Classes

— class application.SearchProductUseCase

*Cas d'usage : Rechercher des produits dans le stock.*

— enum application.SearchProductUseCase.Mode

*Modes de recherche disponibles.*

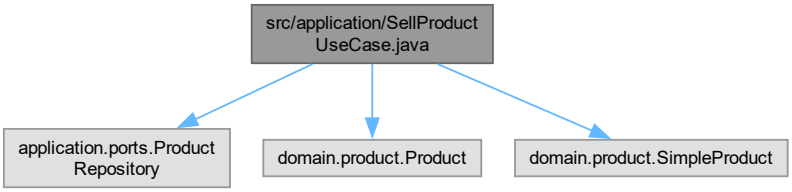
### Paquetages

— package application

## 9.5 Référence du fichier src/application/SellProductUseCase.java

```
import application.ports.ProductRepository;
import domain.product.Product;
import domain.product.SimpleProduct;
```

Graphe des dépendances par inclusion de SellProductUseCase.java:




### Classes

— class application.SellProductUseCase

*Cas d'usage : Vendre un produit du stock.*

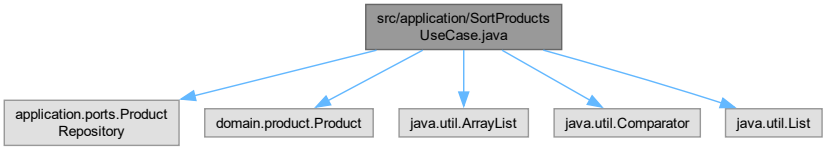
### Paquetages

— package application

## 9.6 Référence du fichier src/application/SortProductsUseCase.java

```
import application.ports.ProductRepository;
import domain.product.Product;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
```

Graphe des dépendances par inclusion de SortProductsUseCase.java:



### Classes

— class application.SortProductsUseCase

*Cas d'usage : Trier les produits du stock par ordre croissant de prix.*

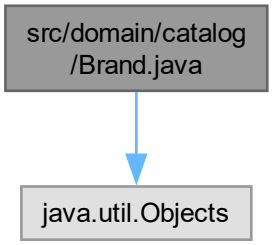
### Paquetages

— package application

## 9.7 Référence du fichier src/domain/catalog/Brand.java

```
import java.util.Objects;
```

Graphe des dépendances par inclusion de Brand.java:




### Classes

— class domain.catalog.Brand

*Représente une marque à laquelle peuvent être associées une ou plusieurs gammes de produits.*

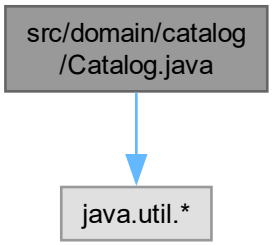
### Paquetages

— package domain.catalog

## 9.8 Référence du fichier src/domain/catalog/Catalog.java

```
import java.util.*;
```

Graphe des dépendances par inclusion de Catalog.java:




### Classes

— class domain.catalog.Catalog

*Fournit un registre centralisé pour la gestion des marques (`Brand`) et des gammes de produits (`ProductLine`).*

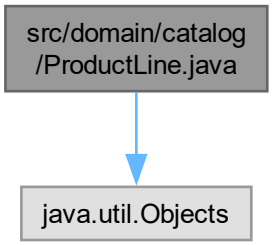
### Paquetages

— package domain.catalog

## 9.9 Référence du fichier src/domain/catalog/ProductLine.java

```
import java.util.Objects;
```

Graphe des dépendances par inclusion de ProductLine.java:



### Classes

— class domain.catalog.ProductLine

*Représente une gamme de produits.*

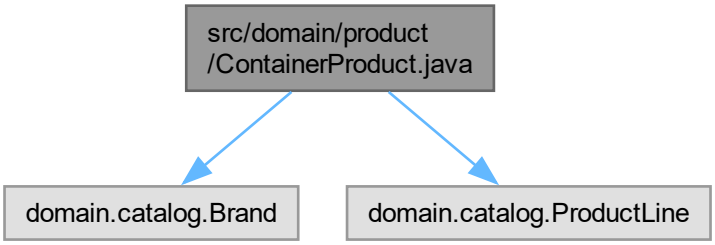
### Paquetages

— package domain.catalog

## 9.10 Référence du fichier src/domain/product/ContainerProduct.java

```
import domain.catalog.Brand;
import domain.catalog.ProductLine;
```

Graphe des dépendances par inclusion de ContainerProduct.java:




### Classes

— class domain.product.ContainerProduct

*Représente un produit conteneur, c'est-à-dire un produit composé qui contient une certaine quantité d'un autre produit.*

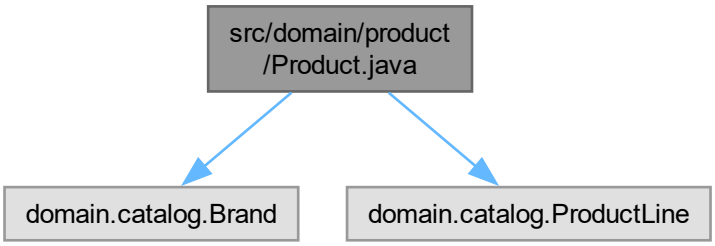
### Paquetages

— package domain.product

## 9.11 Référence du fichier src/domain/product/Product.java

```
import domain.catalog.Brand;
import domain.catalog.ProductLine;
```

Graphe des dépendances par inclusion de Product.java:




### Classes

— class domain.product.Product

*Représente un produit abstrait appartenant à un catalogue.*

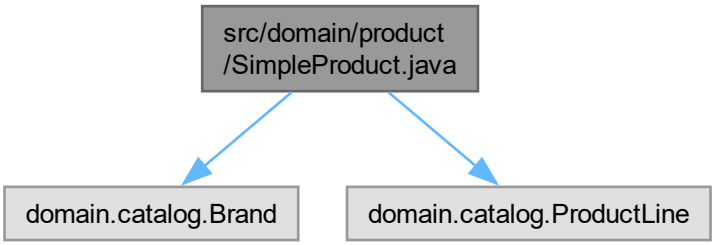
### Paquetages

— package domain.product

## 9.12 Référence du fichier src/domain/product/SimpleProduct.java

```
import domain.catalog.Brand;
import domain.catalog.ProductLine;
```

Graphe des dépendances par inclusion de SimpleProduct.java:



### Classes

— class domain.product.SimpleProduct

*Représente un produit simple du catalogue.*

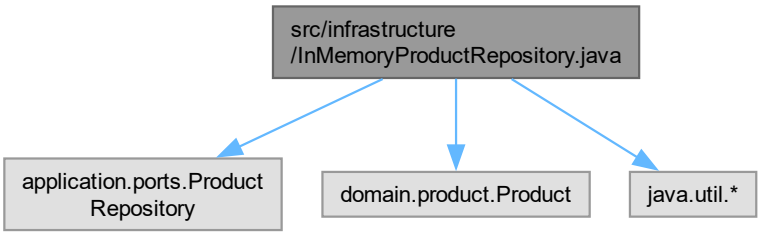
### Paquetages

— package domain.product

## 9.13 Référence du fichier src/infrastructure/InMemoryProductRepository.java

```
import application.ports.ProductRepository;
import domain.product.Product;
import java.util.*;
```

Graphe des dépendances par inclusion de InMemoryProductRepository.java:




### Classes

— class infrastructure.InMemoryProductRepository

*Implémentation en mémoire de l'interface `ProductRepository`.*

### Paquetages

— package infrastructure

## 9.14 Référence du fichier src/infrastructure/ProductIdGenerator.java

### Classes

— class infrastructure.ProductIdGenerator

*Générateur d'identifiants uniques pour les produits.*

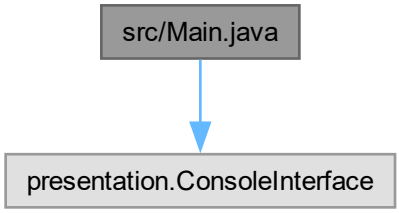
**Paquetages**

— package infrastructure

## 9.15 Référence du fichier src/Main.java

```
import presentation.ConsoleInterface;
```

Graphe des dépendances par inclusion de Main.java:




**Classes**

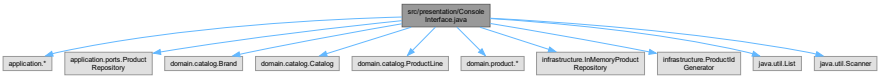
— class Main

*Point d'entrée de l'application de gestion de stock.*

## 9.16 Référence du fichier src/presentation/ConsoleInterface.java

```
import application.*;
import application.ports.ProductRepository;
import domain.catalog.Brand;
import domain.catalog.Catalog;
import domain.catalog.ProductLine;
import domain.product.*;
import infrastructure.InMemoryProductRepository;
import infrastructure.ProductIdGenerator;
import java.util.List;
import java.util.Scanner;
```

Graphe des dépendances par inclusion de ConsoleInterface.java:



### Classes

— class presentation.ConsoleInterface

*Interface utilisateur en ligne de commande pour la gestion de stock.*

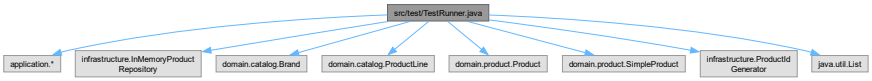
### Paquetages

— package presentation

## 9.17 Référence du fichier src/test/TestRunner.java

```
import application.*;
import infrastructure.InMemoryProductRepository;
import domain.catalog.Brand;
import domain.catalog.ProductLine;
import domain.product.Product;
import domain.product.SimpleProduct;
import infrastructure.ProductIdGenerator;
import java.util.List;
```

Graphe des dépendances par inclusion de TestRunner.java:




### Classes

— class test.TestRunner

*Classe de test manuelle de l'application de gestion de stock.*

### Paquetages

— package test