

# **Les diagrammes de composants**

# Table des matières

|   |           |
|---|-----------|
| <b>I. Diagramme de composants en théorie</b>    | <b>3</b>  |
| <b>II. Exercice : Quiz</b>                      | <b>10</b> |
| <b>III. Diagramme de composants en pratique</b> | <b>11</b> |
| <b>IV. Exercice : Quiz</b>                      | <b>15</b> |
| <b>V. Essentiel</b>                             | <b>16</b> |
| <b>VI. Auto-évaluation</b>                      | <b>16</b> |
| A. Exercice .....                               | 16        |
| B. Test .....                                   | 17        |
| <b>Solutions des exercices</b>                  | <b>19</b> |

# I. Diagramme de composants en théorie

**Durée : 1 h**

**Pré-Requis :** avoir des connaissances en UML

**Environnement de travail :** un pc connecté à internet

## Contexte

Un diagramme de composants va représenter les relations entre les différents composants d'un système. Cette représentation est faite dans une vue que l'on appelle « *statique* ». C'est-à-dire que les composants sont décrits à un instant T et ne peuvent évoluer dans le diagramme. Peu importe ce qui se passe avant ou après le déroulement des interactions, notre diagramme reste fixe. Si nous voulons montrer un changement majeur, il faudra refaire un diagramme de composants.

Les composants, dans le contexte UML, sont des éléments que l'on dit modulaires d'un système. En clair, tous les composants sont indépendants, autonomes et encapsulent des structures complexes. En d'autres termes, les composants ne dépendent d'aucun autre et présentent chacun leur propre structure.

Les éléments encapsulés entrent en contact avec d'autres composants uniquement par le biais des interfaces. Ils peuvent alors mettre à disposition leur(s) interface(s) ou utiliser les interfaces d'autres composants, dans le but d'accéder à leurs fonctions ou leurs services. Sachez qu'il existe deux types d'interfaces.

Pour finir, le diagramme de composants permet de modéliser l'architecture de notre système.

## L'architecture d'un système

Une Architecture Système (AS) est un modèle conceptuel qui définit la structure, le comportement et les différents points de vue d'un système. On essaie, le plus souvent, de chercher quelque chose de modulaire et flexible afin que l'évolution de notre système ne soit pas bloquée par un système trop rigide. La description de cette architecture est organisée de telle sorte à comprendre le système au travers de ses structures et de ses comportements. C'est-à-dire de savoir quels sont les éléments qui le composent et comment ces éléments communiquent entre eux. Elle permet de comprendre un système complexe et de le rendre plus flexible aux évolutions. En termes de programmation, rien n'est jamais figé.

En effet, nous avons tous les jours de plus en plus de données, de fonctionnalités, les langages se mettent à jour constamment et cela nécessite de développer un système qui peut évoluer avec le temps sans se retrouver bloqué, à nous, dans une architecture trop rigide.

## Qu'est-ce qu'un composant logiciel ?

Un composant logiciel est une boîte noire qui offre des services. Une boîte noire contient toutes les informations du logiciel, mais elle n'est pas à la portée de tous. Prenons un exemple facile à comparer : la boîte noire dans les avions. Après le crash d'un avion, les personnes chargées d'analyser le site du crash recherchent au plus vite la boîte noire, afin de la décrypter et d'observer à l'intérieur ce qu'il s'est passé.

Ces services sont accessibles au travers d'interfaces. On peut considérer cette notion d'interface comme l'ensemble des signatures des méthodes accessibles pour le composant. Un composant peut être ici un objet, une fonction, une librairie, ou bien d'autres choses. On connaît les fonctions qui sont contenues dans notre composant. On sait aussi quels sont les paramètres qui sont attendus, et quel résultat devrait sortir en fin de processus. Mais on ne sait pas forcément comment tout cela fonctionne.

Un exemple assez simple renvoie au fait que la plupart des personnes ne savent pas comment fonctionne une fonction « *print()* » en Python ou comment tourne la fonction « *console.log()* » en JavaScript. Pourtant, ce sont des fonctions que l'on utilise souvent, voire très souvent dans la programmation. Nous savons comment il faut appeler la fonction, on sait notamment ce que la fonction attend comme paramètre, et ce qui sort une fois la fonction exécutée.

Si un composant dépend lui-même du résultat d'un autre composant, il y aura une interface décrivant les services attendus.

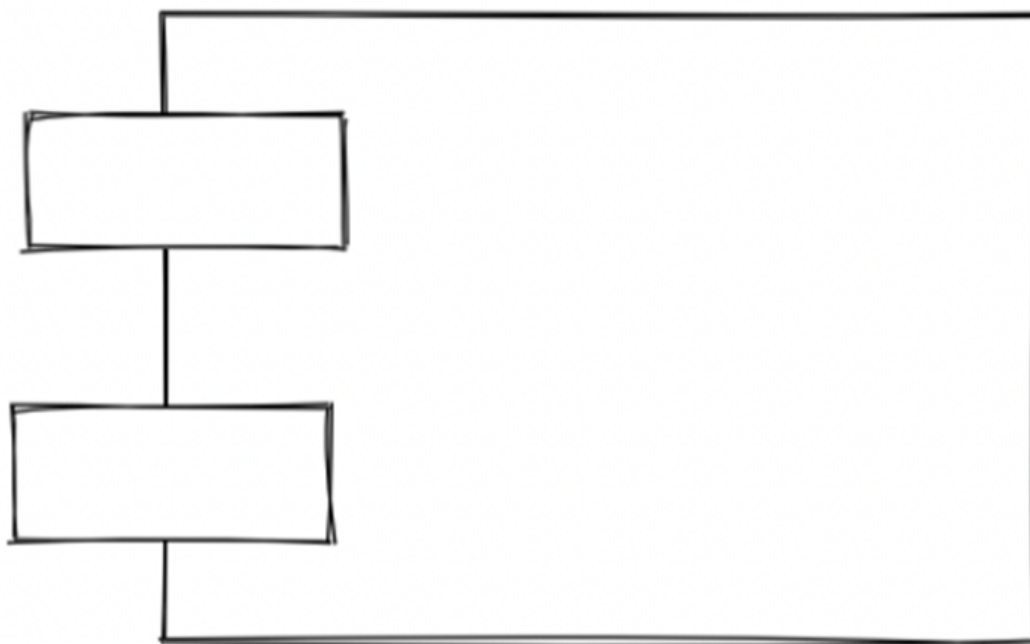
### Les formes et symboles utilisés dans les diagrammes de composants UML

Les diagrammes de composants vont s'étendre de la vue sommaire, autrement dit d'une vue d'ensemble, à des modèles qui seront plus détaillés et plus complexes. Il faut se familiariser avec les différents symboles UML, afin de pouvoir rapidement identifier les éléments dont il est question. Commençons par les symboles de base.

Symbole de composant :

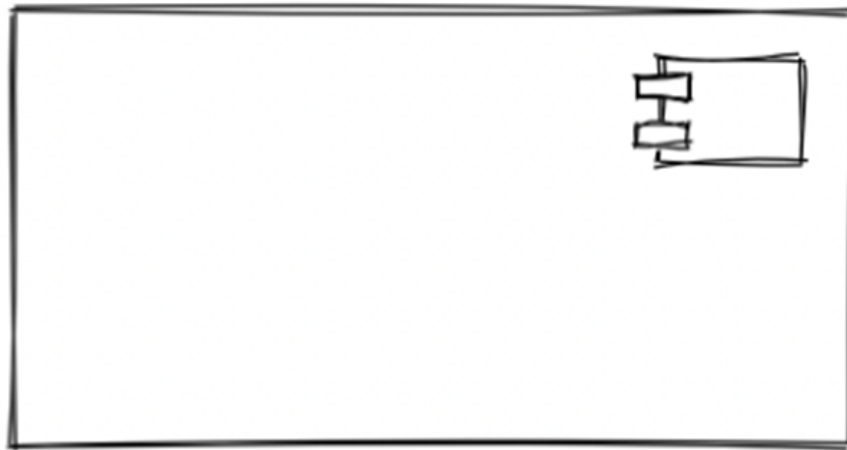
Le composant va fournir et consommer un comportement via les interfaces, mais également via d'autres composants. Il existe deux façons de modéliser un composant en UML.

Avec UML 1.0, le composant va être modélisé sous les traits d'un bloc de forme rectangulaire avec deux plus petits rectangles qui dépassent sur le côté. Ce sont ces deux rectangles sur le côté qui permettent de démarquer notre composant de l'ensemble de notre diagramme.



**Symbole de composant  
UML 1.0**

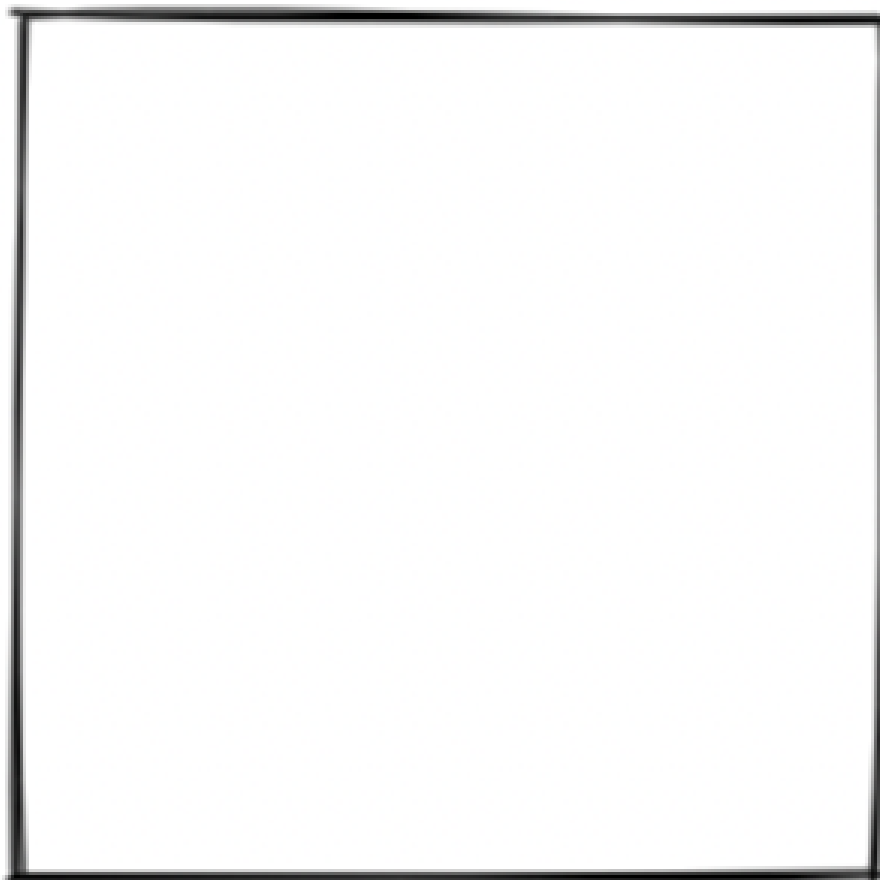
Avec UML 2.0, le composant est modélisé d'un bloc rectangulaire avec une petite image de la forme du composant que l'on fait avec UML 1.0. Cette modélisation est un peu plus subtile, mais peut se présenter plus esthétique.



Symbole de composant UML 2.0

Symbole de nœud :

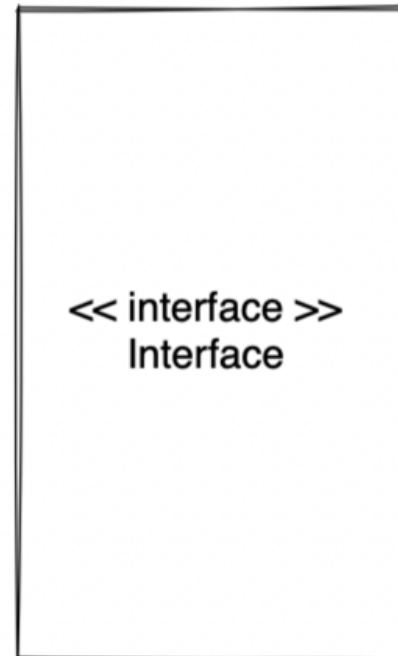
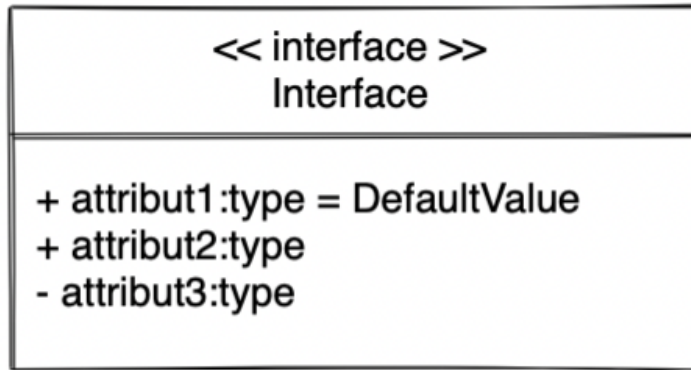
Ce symbole représente des objets matériels ou des objets logiciels étant situés à un niveau au-dessus des composants. Il possède une taille similaire à celle de notre composant. **Attention à ne pas le confondre avec le symbole de port que nous allons voir ci-dessous.** Un symbole de port est plus petit et sera relié par un lien, nous verrons cela d'ici peu.



### Symbole de nœud

Symbole d'interface :

Le symbole d'interface indique les entrées ou les données, qu'un composant va recevoir ou fournir. On peut représenter ces interfaces par des notes (texte) ou des symboles.



### Symbole d'interface

Symbole de port :

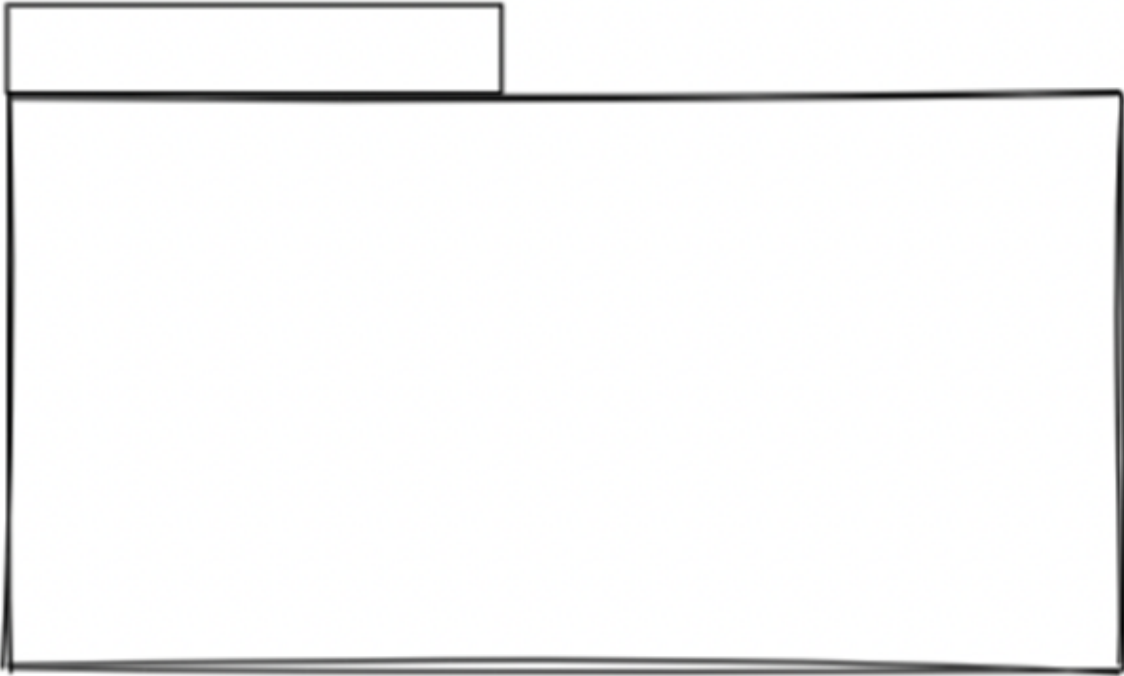
Le symbole de port sert à spécifier un point d'interaction entre un composant et son environnement. On symbolise les ports par un carré. Le port est utilisé lorsque le composant sur lequel il est inséré délègue les interfaces à une classe interne.



### Symbole de port

**Symbole de paquetage :**

Ce symbole permet de regrouper plusieurs éléments d'un système. Les packages peuvent englober plusieurs éléments.

**Symbole de paquetage****Symbole de dépendance :**

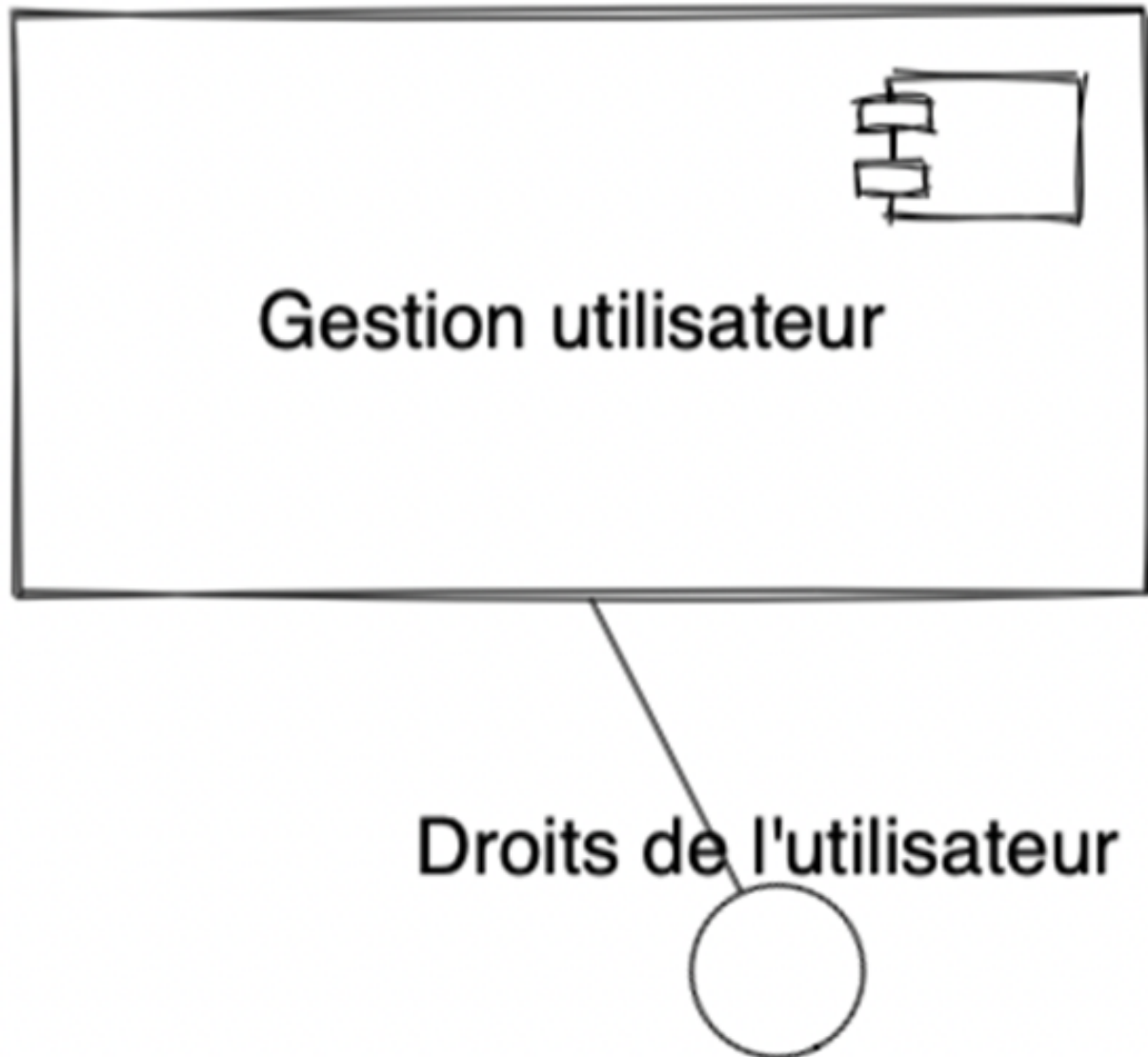
Ce symbole montre les relations entre différentes parties de notre système. Comme son nom l'indique, cela va indiquer des relations de dépendance. Bien que l'on puisse afficher plus de détails dans la relation entre deux composants, grâce à la notation d'une interface fournie ou d'une interface requise.

**Symbole de  
dépendance**

## Les différentes interfaces du diagramme

Interface fournie :

Une interface fournie permet de fournir un service aux autres composants logiciels. Dans cet exemple :



### Exemple d'interface fournie

Le module de gestion des utilisateurs va ici fournir les droits d'un utilisateur qui pourront être utilisés pour donner, ou non, un accès à certaines fonctionnalités.

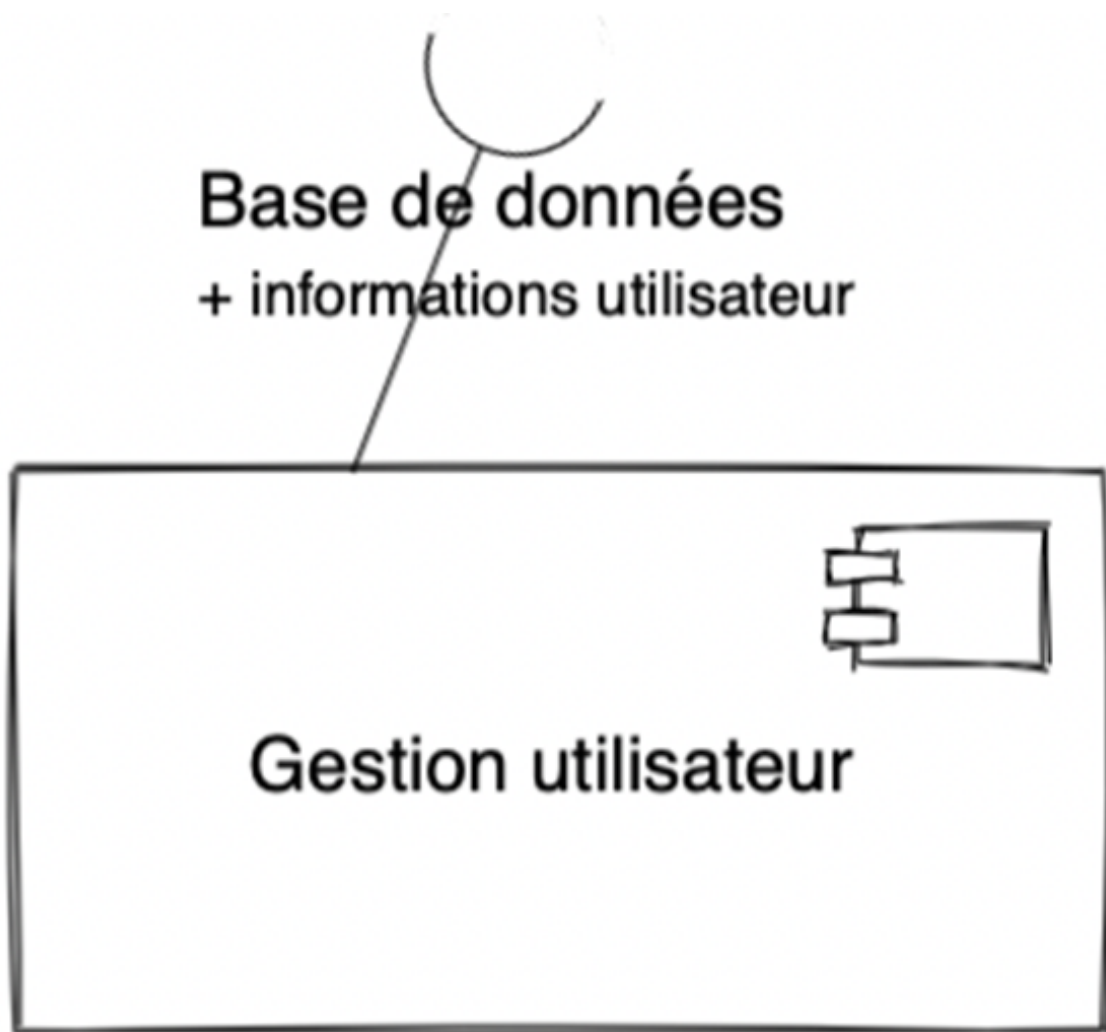
Le composant en lui-même peut être traité comme une boîte noire, car nous ne savons pas forcément comment les données seront transférées par exemple.

Une interface fournie, comme nous pouvons l'observer, est représentée par une ligne droite qui part du composant et qui contient un cercle à l'extrémité. En bref, ce composant produit des informations qui seront alors utilisées par l'interface requise d'un autre composant.



Interface requise :

Une interface requise permet à un composant de procéder à une demande d'information de la part d'un autre composant. Dans cet exemple :



**Exemple d'interface  
requise**

Le module de gestion des utilisateurs va avoir besoin d'information provenant d'une base de données, pour prendre connaissance des informations sur un ou des utilisateurs.

Une interface requise se caractérise par une ligne droite partant du composant et qui contient un demi-cercle à son extrémité. En bref, ce symbole va servir aux interfaces où un composant va avoir besoin de recueillir des informations, dans le but d'accomplir correctement sa fonction.

À première vue, les diagrammes des composants peuvent paraître complexes, mais ils sont d'une valeur inestimable pour la conception de votre système. Ces diagrammes vont aider vos équipes à :

- Imaginer la structure physique du système,
- Faire ressortir les composants du système et à leurs relations,
- Mettre en évidence le comportement de chaque service vis-à-vis de l'interface.

## Exercice : Quiz

[solution n°1 p.21]

### Question 1

Un diagramme de composants représente une vue dynamique.

- ☐ Vrai
- ☐ Faux

### Question 2

Lors du développement d'une architecture système, on doit faire en sorte que celle-ci soit rigide et difficilement modulaire.

- ☐ Vrai
- ☐ Faux

### Question 3

On place une interface requise sur un composant lorsque celui-ci est entièrement indépendant et n'a besoin d'aucun autre composant pour vivre.

- ☐ Vrai
- ☐ Faux

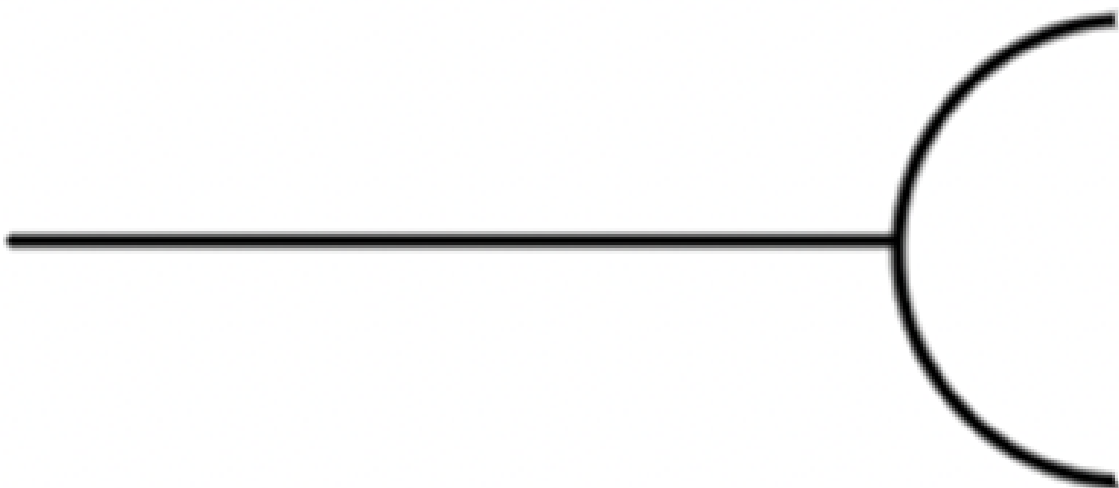
### Question 4

Une interface fournie :

- ☐ Apporte un service aux autres composants logiciels
- ☐ Permet à un composant de procéder à une demande d'information de la part d'un autre composant

### Question 5

Que représente ce symbole ?



- ☐ Un port
- ☐ Une interface fournie
- ☐ Une dépendance
- ☐ Une interface requise

### III. Diagramme de composants en pratique

Dans cette seconde partie, nous allons voir ensemble comment constituer un diagramme de composants. Voyons dans un premier temps les étapes à respecter.

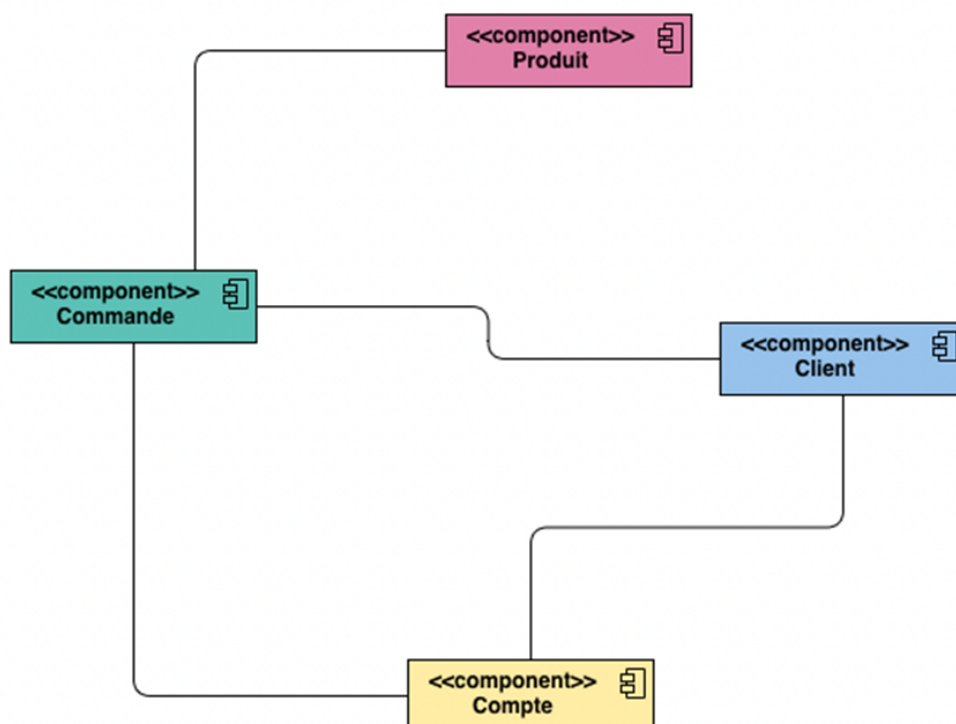
D'abord, il faut déterminer quel est l'objectif de notre diagramme de composants. Il faut identifier quels sont les items (documents, fichiers, etc.) qu'il va falloir représenter dans notre diagramme. Ensuite, au fil des relations que l'on va établir entre les éléments que nous avons pu identifier dans l'étape précédente, il faut alors ajouter les composants dans notre diagramme, les positionner, puis les regrouper entre eux selon les relations que l'on souhaite mettre en place. D'autres éléments comme les interfaces, des objets, des classes, des dépendants, etc. peuvent venir compléter notre diagramme.

Pour finir, si on le souhaite, on peut ajouter des notes à notre diagramme afin de mettre en lumière certains détails, pour qu'ils soient facilement compris par quelqu'un qui souhaite consulter notre diagramme.

#### Un diagramme de composants pour un système d'achat en ligne

Pour respecter nos étapes, nous allons dans un premier temps définir quels sont les éléments dont nous aurons besoin. Dans un processus d'achat en ligne, les éléments entrant en compte sont notamment la commande, le produit en lui-même. Sans oublier le client qui doit procéder à la commande et qui doit pouvoir régler son achat grâce à son compte client dans lequel sont, notamment, répertoriées ses informations bancaires.

Produit, client, compte bancaire du client : bien, nous avons nos éléments. Passons à la prochaine étape. Nous allons créer une disposition de notre diagramme et relier chaque composant entre eux.



### Mise en place des composants dans un diagramme de composants

Ici, nous avons placé chaque composant comme on le souhaite. Vous pouvez utiliser une disposition différente de celle-ci, mais attention à vous y retrouver lors de l'ajout d'informations. Puis nous avons fait un lien entre chacun d'entre eux. Le composant commande est associé au composant produit, au composant client ainsi qu'au composant du compte de notre client. Le composant client et le composant compte sont également associés l'un avec l'autre.

Une fois les différentes associations faites entre les différents composants, nous pouvons désormais apporter quelques précisions sur la nature de chacune d'elles et comment elles interagissent.

Pour ce qui est du composant Commande :

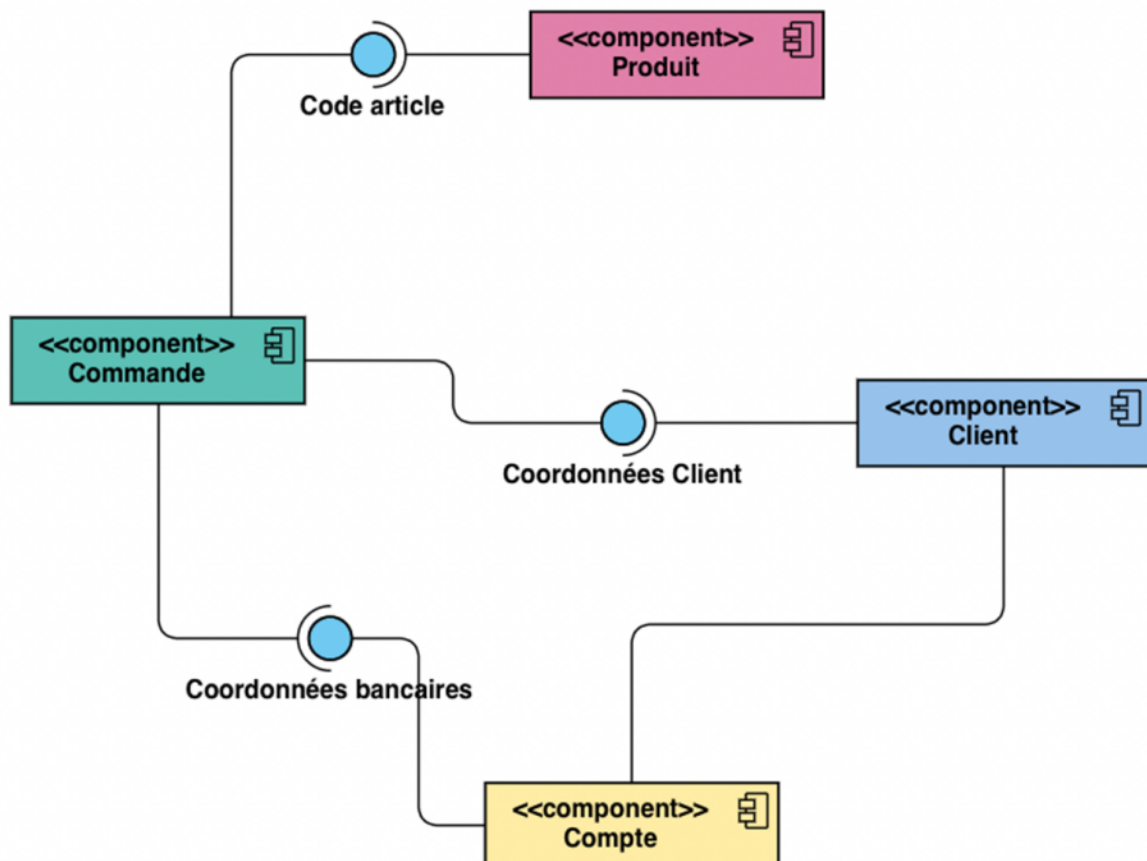
- Il a **besoin** du code provenant du composant produit afin de savoir précisément de quel produit le client a commandé.
- Il a **besoin** des détails du composant client (nom, prénom, adresse, etc.) afin de pouvoir expédier la commande à la bonne destination.
- Il a **besoin** des coordonnées bancaires du composant client afin de procéder au paiement.

Le composant Produit va **fournir** le code de chaque produit.

Le composant Client va **fournir** ses informations personnelles.

Le composant Compte va **fournir** les informations du compte afin de procéder au paiement.

En mettant bout à bout tous ces besoins, nous en arrivons à un diagramme comme celui-ci :



**Mise en place des interfaces fournies et requises dans un diagramme de composants**

Ici, nous avons procédé ensemble à la création d'un diagramme de composants relativement simple et sans trop de fioritures. Cependant, il existe des diagrammes qui nécessitent bien plus de réflexion et de temps pour être créés.

**Un diagramme de composants pour un système d'achat en ligne au moment d'une commande**

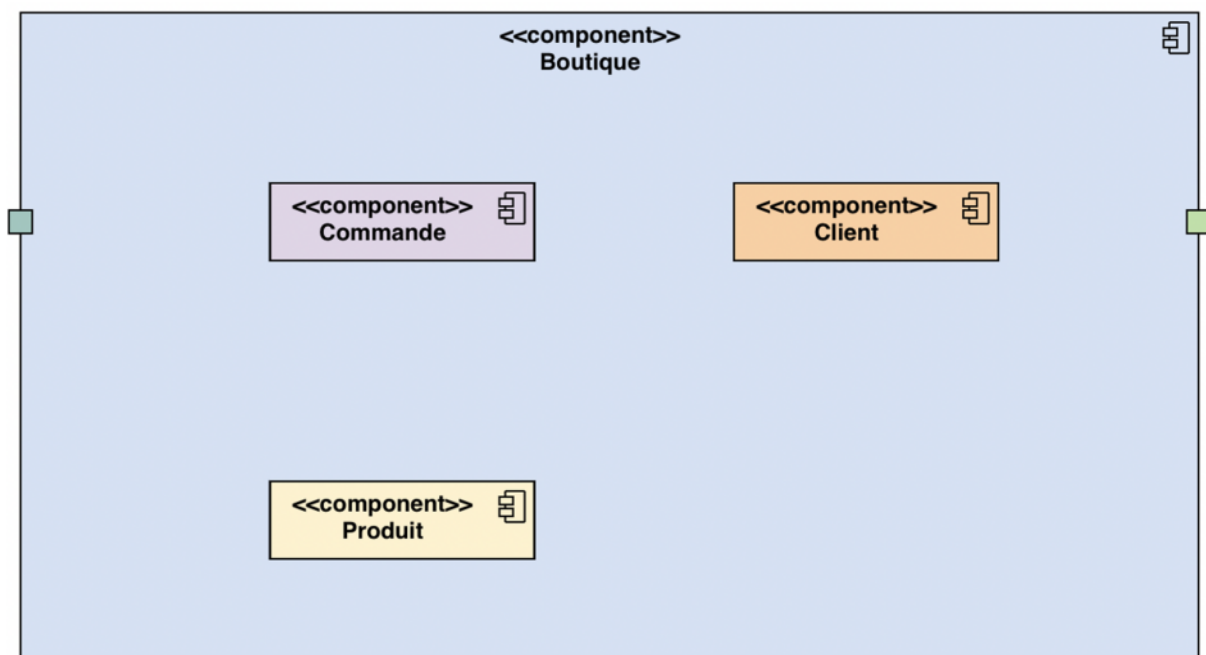
Pour garder le même type de système, il est possible de ne prendre en compte que le moment où une nouvelle commande arrive dans la boutique, en considérant le composant Commande, le composant Client et le composant Produit. Le tout ferait partie d'un seul et même grand composant Boutique, qui lui-même ferait partie d'un diagramme de composant plus complexe. Autrement dit, nous avons un composant Boutique et, en son sein, nous allons retrouver le composant Client, le composant Produit ainsi que le composant Commande.

En résumé pour comprendre notre diagramme de composant :

- Le composant Client :
  - Requier le compte (les informations) qui est fourni par une classe externe.
  - Fournit les coordonnées du client au composant Commande.
- Le composant Commande :
  - Requier les coordonnées du client.
  - Requier de connaître le(s) produit(s) commandé(s).
  - Fournit la commande à une classe externe.
- Le composant Produit va, quant à lui, fournir les produits commandés par le client via le composant commande.

Pour commencer, nous avons un seul et même grand composant que l'on appelle Boutique.

Dans ce grand composant, nous avons plusieurs composants : le composant Commande, le composant Client et le composant Produit. Prenons uniquement ces informations et commençons par mettre sur papier une esquisse de notre diagramme :



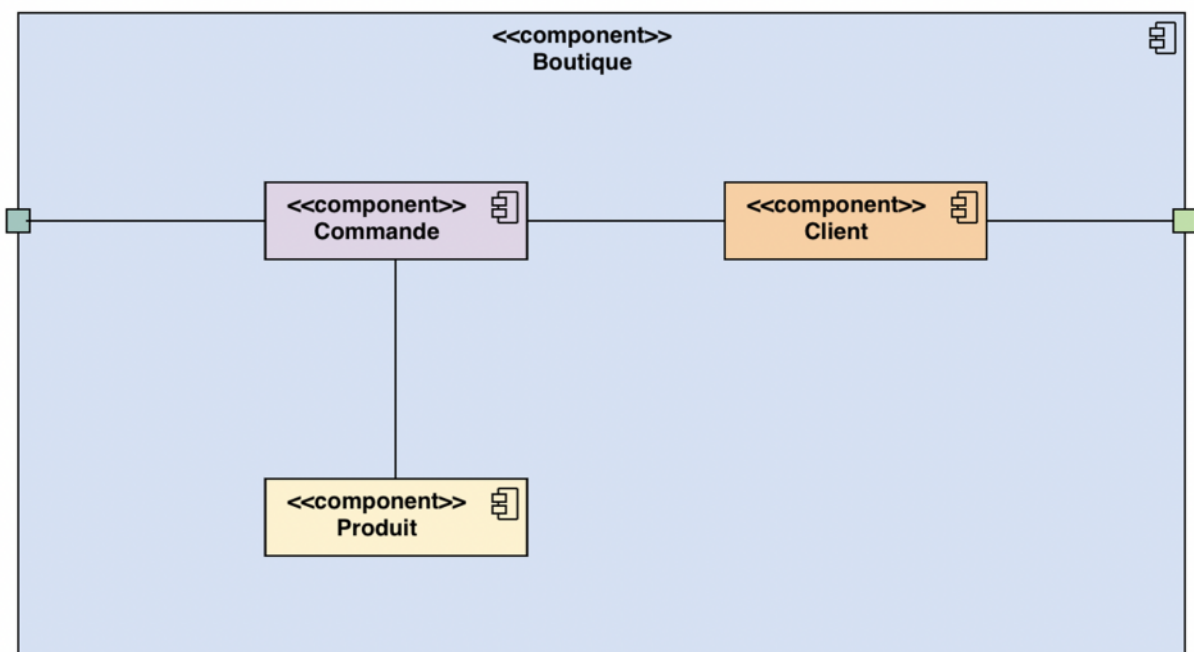
### Exemple d'un diagramme de composants dans un composant

Ici, nous avons instauré dans un premier temps notre composant Boutique, puis nous avons positionné nos différents composants à l'intérieur. Il ne faut pas oublier les symboles de port, afin d'indiquer qu'il délègue les interfaces à une classe interne.

Les symboles qui représentent un port vont être placés quelque part sur la limite de notre composant boutique, mais il faut faire attention à correctement relier les composants qui vont interagir avec l'extérieur de notre composant Boutique.

Comme dans l'exemple précédent, vous pouvez utiliser une disposition différente de celle-ci, mais attention à vous y retrouver lors de l'ajout d'informations.

Vient le moment de faire les liens entre les composants. On sait que la commande va interagir avec le composant Produit et le composant Client. On va donc relier les composants Produit et Client entre eux, afin de marquer l'interaction. On sait également que notre composant Client et notre composant Commande vont tous les deux déléguer leurs interfaces à des classes externes. Dans le cas où un composant délègue son interface à une classe externe, nous pouvons donc les relier chacun à un port :

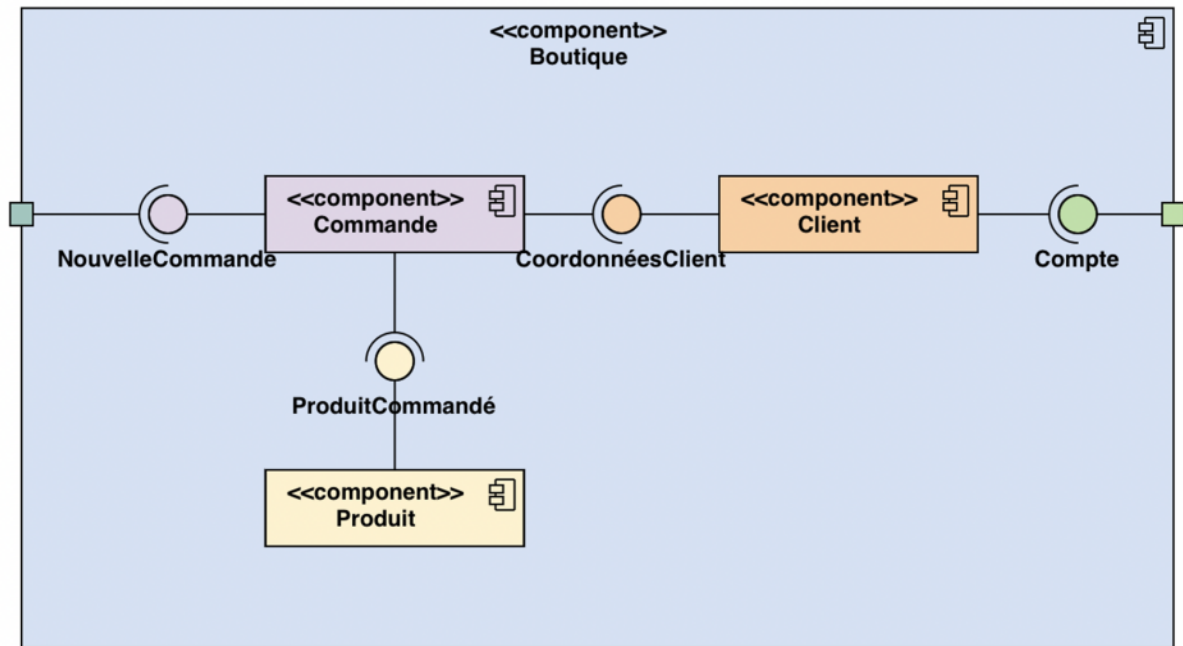


### Mise en place des symboles des liens

Comme le montre notre exemple ci-dessus, nous avons alors relié le composant Commande à un port externe, puis nous avons procédé de la même manière pour le composant Client.

Une fois les liens entre les composants faits, on apporte alors plus de détails dans les interfaces utilisées :

- Le composant commande va fournir une nouvelle commande que le client aura demandé et la commande sera récupérée par un composant extérieur.
- Le composant Produit fournit au composant commande les produits qui ont été commandés.
- Le composant Client va fournir les coordonnées du client au composant commande afin que celle-ci soit payée, expédiée, etc.



**Mise en place des interfaces fournies et requises**

Ce deuxième diagramme de composants présente une vue détaillée d'un composant faisant partie d'un diagramme de composants plus complexe. Nous savons désormais qu'il est possible de détailler plus en profondeur nos diagrammes de composants, afin d'apporter plus de détails sur le fonctionnement attendu de notre future application / site web.

Par exemple, il serait possible d'aller détailler, si on le souhaite, le composant **Commande** afin de voir comment celui-ci fonctionne au sein de notre système. À l'inverse, on pourrait agrandir d'un niveau notre vue et voir le composant **Boutique** autour d'autres composants qui fonctionnent ensemble, comme dans notre exemple précédent.

## Exercice : Quiz

[solution n°2 p.22]

### Question 1

Deux composants doivent obligatoirement être reliés par une interface requise et une interface fournie.

- ☐ Vrai
- ☐ Faux

### Question 2

Il est possible d'imbriquer des composants dans un composant.

- ☐ Vrai
- ☐ Faux

### Question 3

Il est possible, dans un diagramme de composants, d'énumérer les interfaces, les méthodes ou les attributs d'un composant.

- ☐ Vrai
- ☐ Faux

#### Question 4

Un composant est considéré comme une boîte noire s'il est possible de voir ce qui se passe à l'intérieur de façon précise.

- ☐ Vrai
- ☐ Faux

#### Question 5

Il est possible d'intégrer des acteurs dans un diagramme de composants.

- ☐ Vrai
- ☐ Faux

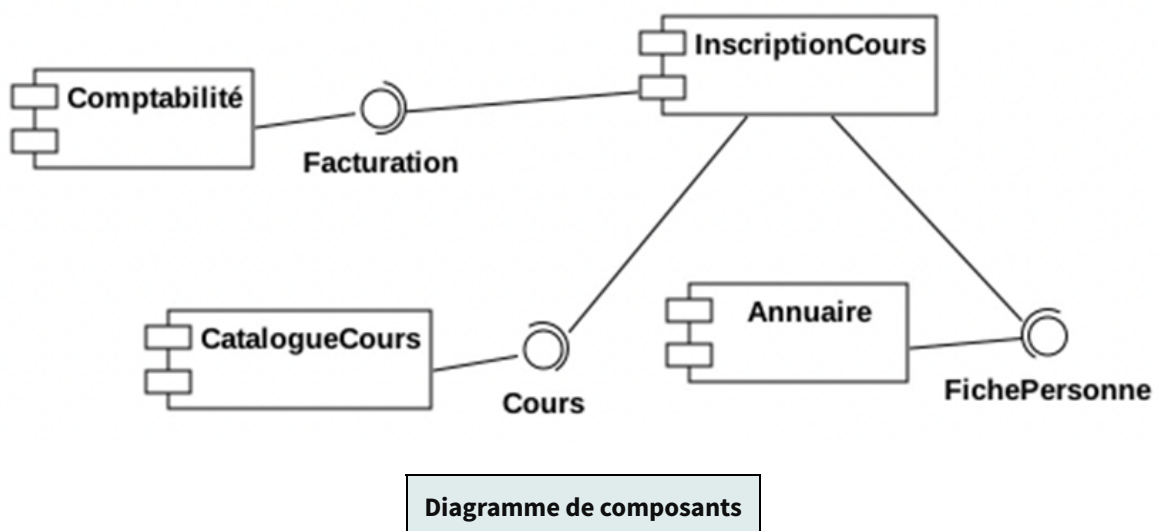
## V. Essentiel

La réalisation d'un diagramme de composants est constituée de plusieurs étapes. Tout d'abord, il faut déterminer l'objectif du diagramme et identifier les éléments tels que les fichiers, les documents que vous devez représenter sur le diagramme. Au fur et à mesure, ajoutez d'abord les composants à votre diagramme, en les regroupant dans d'autres composants comme vous le souhaitez. L'étape suivante consiste à ajouter d'autres éléments tels que des interfaces, des classes, des objets, des dépendances, etc. Enfin vous pouvez joindre des commentaires à différentes parties du diagramme de composants pour clarifier certains détails aux autres.

Si on vous fournit un diagramme déjà finalisé, pensez à prendre votre temps afin de le décortiquer, de saisir dans un premier temps l'idée globale, puis d'analyser petit à petit les éléments afin de saisir les subtilités que peut présenter un diagramme de composants.

## VI. Auto-évaluation

### A. Exercice





Source : Emmanuel Renaux<sup>1</sup>

Vous êtes en train de créer une nouvelle application. Votre responsable vous donne la responsabilité de prendre en charge le nouveau stagiaire toute la journée. Vous allez devoir lui expliquer le projet en cours.

**Question 1**

[solution n°3 p.24]

D'après le schéma, de quelle type d'application s'agit-il ? Quels sont les composants ?

**Question 2**

[solution n°4 p.24]

Quelles sont les relations entre les composants ?

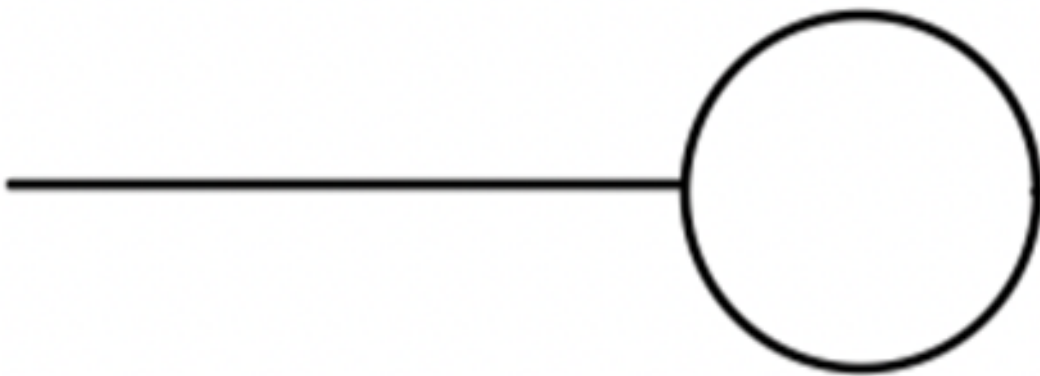
**B. Test**

**Exercice 1 : Quiz**

[solution n°5 p.24]

Question 1

Ce symbole représente l'interface fournie.



- ☐ Vrai
- ☐ Faux

Question 2

Cette combinaison de symboles représente une relation de dépendance entre deux composants.



- ☐ Vrai
- ☐ Faux

---

<sup>1</sup> <https://manurnx.wp.imt.fr/2017/01/26/modelisation-uml-dapplications-logicielles/>

Question 3

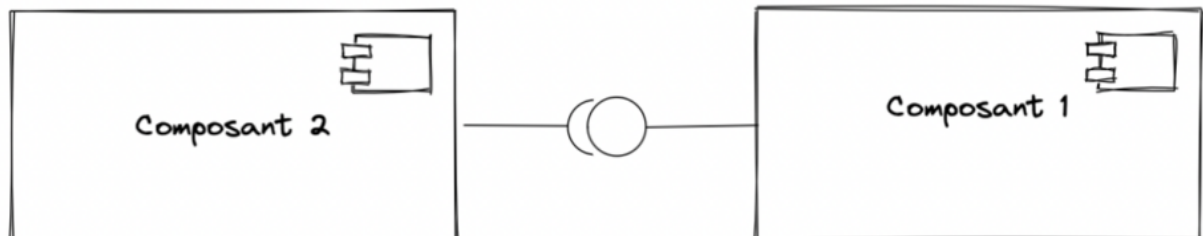
Est-il possible qu'un composant puisse à la fois avoir une interface fournie et une interface requise comme dans l'exemple suivant ?



- ☐ Oui
- ☐ Non

Question 4

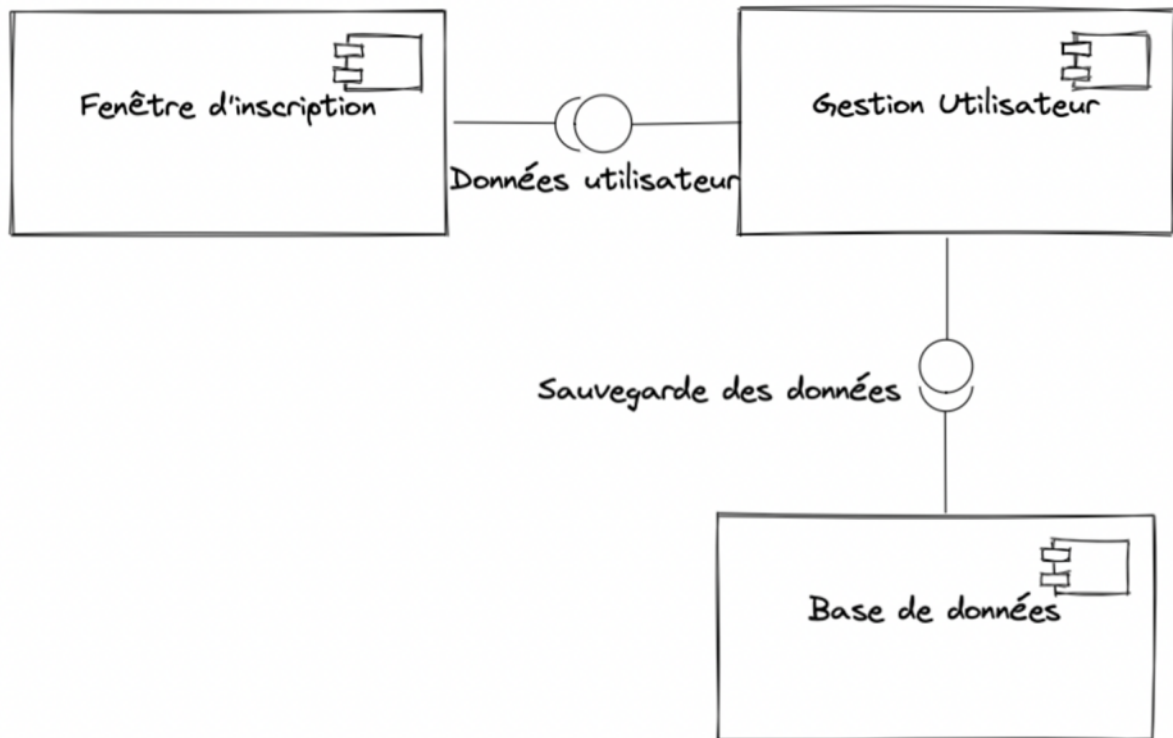
Dans cet exemple, le composant 2 fournit des informations au composant 1.



- ☐ Vrai
- ☐ Faux

Question 5

Une incohérence s'est-elle glissée dans le diagramme suivant ? Si oui, justifiez.




- ☐ Non, aucune incohérence n'a été relevée
- ☐ Le composant Fenêtre d'inscription devrait être relié à la base de données directement
- ☐ La sauvegarde des données ne dépend pas de la base de données
- ☐ La relation de dépendance entre la fenêtre d'inscription et la gestion utilisateur est mauvaise

## Solutions des exercices




**Exercice p. 10 Solution n°1****Question 1**

Un diagramme de composants représente une vue dynamique.

- ☐ Vrai
- ☒ Faux
-  Faux. Un diagramme représente une vue statique d'un système.


**Question 2**

Lors du développement d'une architecture système, on doit faire en sorte que celle-ci soit rigide et difficilement modulaire.

- ☐ Vrai
- ☒ Faux
-  Faux. On essaie de rendre l'architecture la plus souple et flexible possible, afin de durer le plus longtemps dans le temps et de pouvoir être mis à jour si nécessaire.


**Question 3**

On place une interface requise sur un composant lorsque celui-ci est entièrement indépendant et n'a besoin d'aucun autre composant pour vivre.

- ☐ Vrai
- ☒ Faux
-  Faux, une interface requise stipule que le composant a besoin d'un autre composant pour, par exemple, recevoir des données provenant de la base de données.

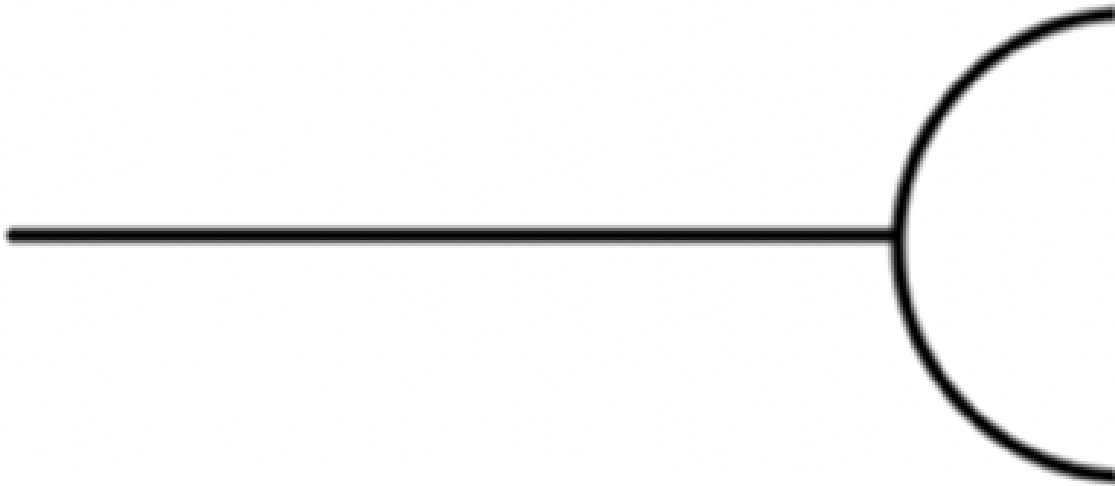
**Question 4**


Une interface fournie :

- ☒ Apporte un service aux autres composants logiciels
- ☐ Permet à un composant de procéder à une demande d'information de la part d'un autre composant
-  Une interface fournie apporte un service aux autres composants logiciels.

**Question 5**

Que représente ce symbole ?




- ☐ Un port
- ☐ Une interface fournie
- ☐ Une dépendance
- ☒ Une interface requise
-  Ce symbole représente bel et bien une interface requise.

### Exercice p. 15 Solution n°2


#### Question 1

Deux composants doivent obligatoirement être reliés par une interface requise et une interface fournie.

- ☐ Vrai
- ☒ Faux
-  C'est faux. Il est possible de relier deux composants à l'aide d'une flèche qui signifie une relation de dépendance. Les symboles d'interfaces fournissent cependant plus de précisions quant à la nature de la relation entre deux composants.

#### Question 2

Il est possible d'imbriquer des composants dans un composant.


- ☒ Vrai
- ☐ Faux
-  C'est vrai. Il est possible de placer un ou plusieurs composants au sein d'un seul et même composant. Un composant peut être constitué de plusieurs composants.

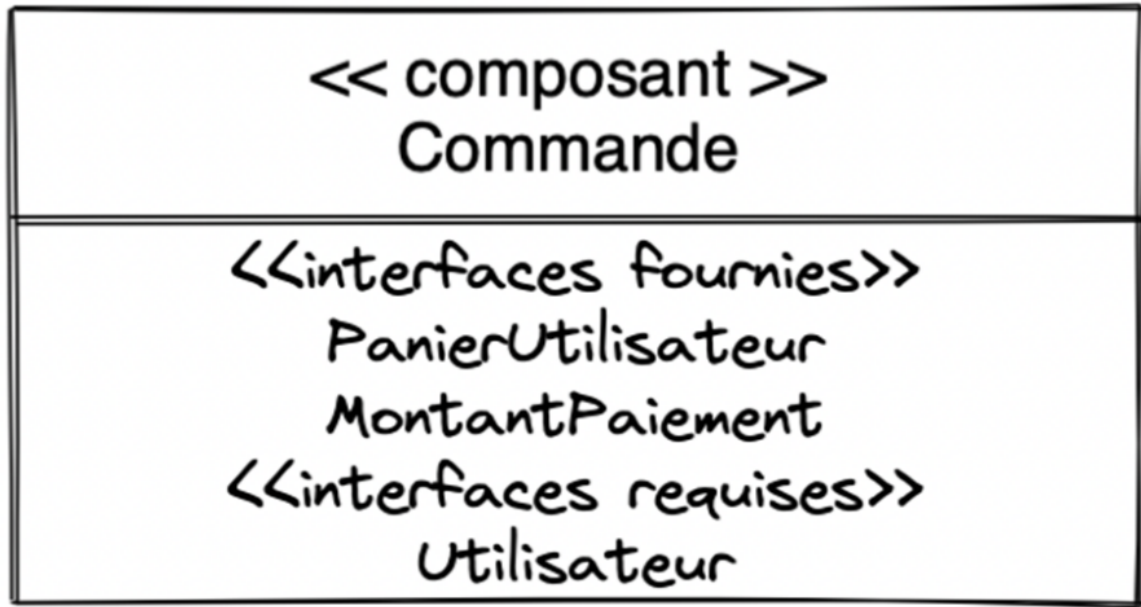
#### Question 3

Il est possible, dans un diagramme de composants, d'énumérer les interfaces, les méthodes ou les attributs d'un composant.

☒ Vrai

☐ Faux

 C'est vrai. Il est en effet possible d'énumérer des informations supplémentaires au sujet des composants, mais c'est toutefois optionnel. Par exemple :




#### Question 4

Un composant est considéré comme une boîte noire s'il est possible de voir ce qui se passe à l'intérieur de façon précise.

☐ Vrai

☒ Faux


 C'est faux, c'est tout le contraire. Le composant en lui-même peut être traité comme une boîte noire, car nous ne savons pas forcément comment les données seront transférées par exemple.

#### Question 5

Il est possible d'intégrer des acteurs dans un diagramme de composants.

☒ Vrai

☐ Faux

 Il est tout à fait possible, si nécessaire, d'intégrer des acteurs au sein de notre diagramme de composants.

**p. 17 Solution n°3**

Ce diagramme de composants représente un système d'inscription à des cours particuliers.

Il est composé de 4 composants :

- Comptabilité
- InscriptionCours
- CatalogueCours
- Annuaire

**p. 17 Solution n°4**

Tous les composants sont reliés à un composant principal qui est l'inscription aux cours. Pour pouvoir inscrire une personne à un cours, le composant semble avoir besoin de plusieurs informations. On peut observer cela notamment avec les symboles d'interfaces requises qui partent du composant.

On peut supposer que pour pouvoir s'inscrire à un cours, il est nécessaire d'avoir réglé le prix de celui-ci, que le cours soit disponible dans le catalogue de cours et qu'une personne s'inscrive en fournissant ses coordonnées.

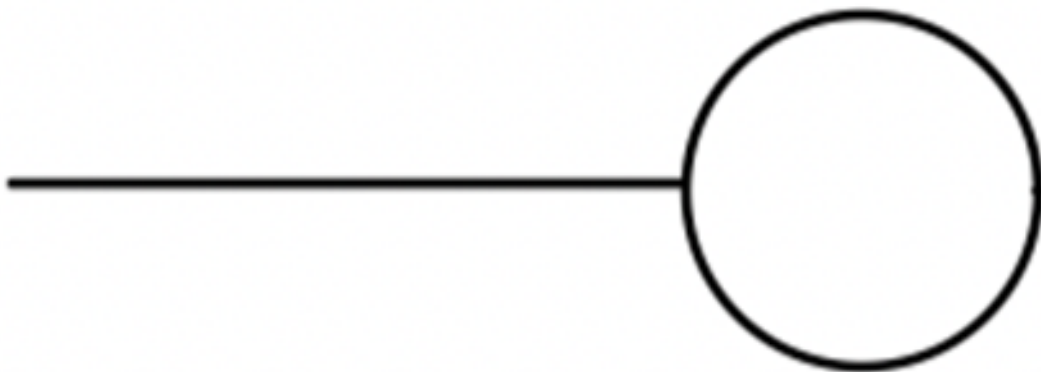
Le composant Comptabilité va fournir la ou les facturations au composant inscription.

Le composant Catalogue va fournir la liste de tous les cours disponibles au composant inscription.

Enfin, le composant Annuaire va fournir la ou les fiches des personnes inscrites pour un cours en question.

**Exercice p. 17 Solution n°5****Question 1**

Ce symbole représente l'interface fournie.



- ☐ Vrai
- ☒ Faux
- ☐ Faux. Ce symbole représente l'interface requise.



**Question 2**

Cette combinaison de symboles représente une relation de dépendance entre deux composants.



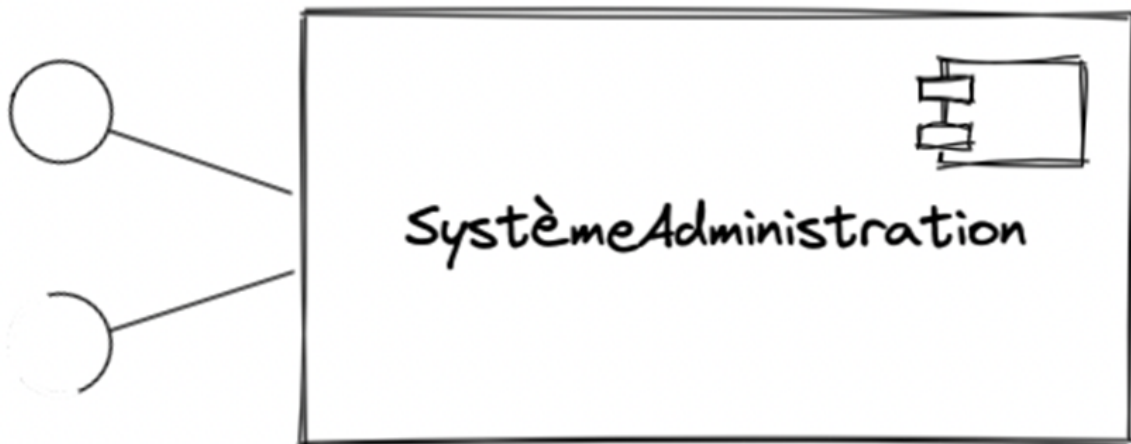
☒ Vrai

☐ Faux

**Q** C'est vrai. Bien que le symbole de dépendance soit représenté par une flèche, le fait de combiner une interface requise et une interface fournie signifie que les deux composants sont dépendants l'un de l'autre pour fonctionner correctement.

**Question 3**

Est-il possible qu'un composant puisse à la fois avoir une interface fournie et une interface requise comme dans l'exemple suivant ?



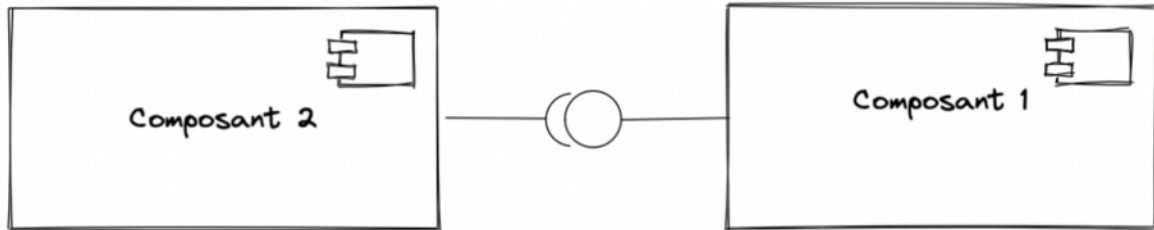
☒ Oui

☐ Non

**Q** Oui. Il est tout à fait possible d'avoir ce genre de cas de figure. Par rapport aux exemples que nous avons utilisés dans ce cours, les interfaces sont simplement placées du même côté.

**Question 4**

Dans cet exemple, le composant 2 fournit des informations au composant 1.



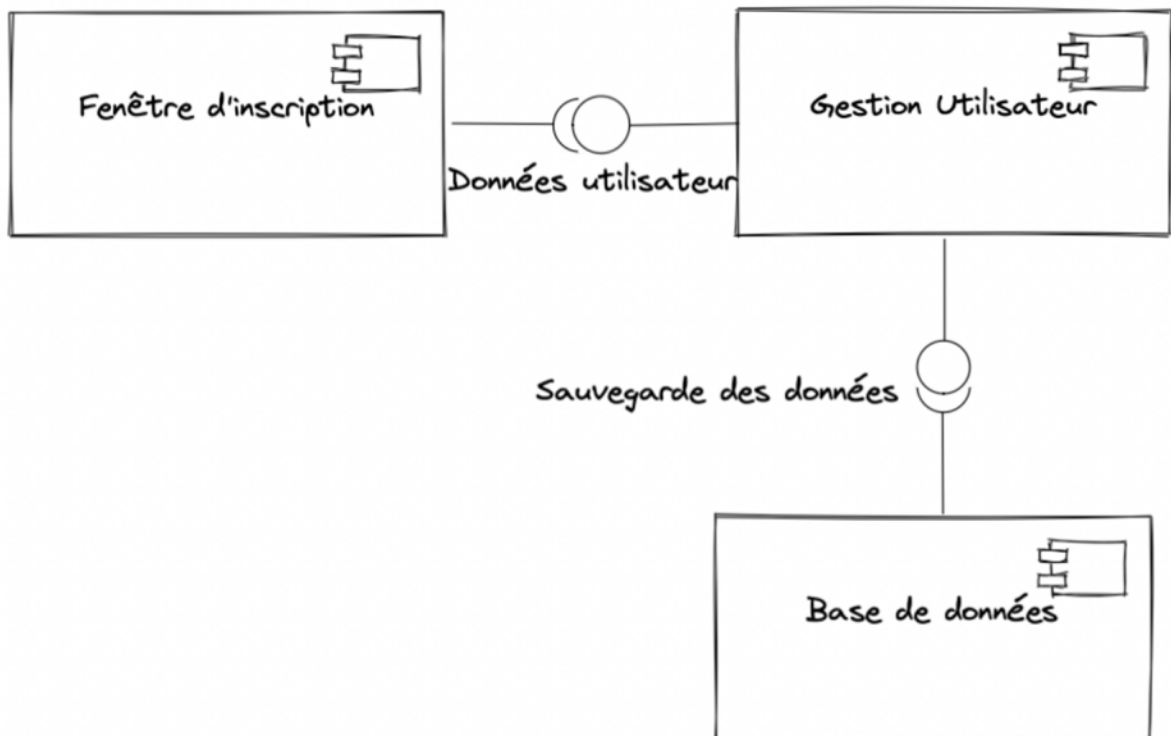
☐ Vrai

☒ Faux

Q Faux. Dans l'exemple ci-dessus, le composant 2 requiert des informations du composant 1. C'est donc ce dernier qui fournit des informations au composant 2.

### Question 5

Une incohérence s'est-elle glissée dans le diagramme suivant ? Si oui, justifiez.



☐ Non, aucune incohérence n'a été relevée

☐ Le composant Fenêtre d'inscription devrait être relié à la base de données directement

☐ La sauvegarde des données ne dépend pas de la base de données

☒ La relation de dépendance entre la fenêtre d'inscription et la gestion utilisateur est mauvaise

Q La relation de dépendance entre la fenêtre d'inscription et la gestion utilisateur est erronée. En effet, la fenêtre d'inscription ne peut pas avoir besoin des données utilisateurs car, comme son nom l'indique, c'est une fenêtre pour s'inscrire et, par conséquent, la fenêtre va **fournir** les données de l'utilisateur au composant gestion Utilisateur. Celui-ci va alors les fournir à la base de données.