IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

L'essentiel d'UML

A. Beugnard & J.C. Royer

Diaporama – UE-IR-S6 –
Session 3
1er semestre 2025

Un programme

```
1  /**
2  * Version simple de la classe Product. Un produit possède un nom. Il
3  * est possible de récupérer ce nom et de transformer un produit en
4  * chaîne de caractères .
5  * @authors F.Dagnat
6  */
7  public class Product {
8  /** le nom du produit sous forme d'une chaîne de caractères */
9  private String name;
10 /** un constructeur qui prend en paramètre le nom du nouveau produit */
11 public Product(String name) { this.name = name; }
12 /** rend une chaîne de caractères qui est le nom du produit */
13 public String getName() { return name; }
14 /** rend une chaîne de caractères qui décrit le produit */
15 public String toString() { return "Produit " + name; }
16 /** Une méthode main qui teste cette classe */
17 public static void main(String[] args) {
18 Product p1 = new Product("p1");
19 System.out.println(p1);
20 new Product("");
21 Product p3 = new Product("p3");
22 System.out.println(p3);
23 System.out.println("Le nom de p3 est " + p3.getName());
24 }
25 }
```

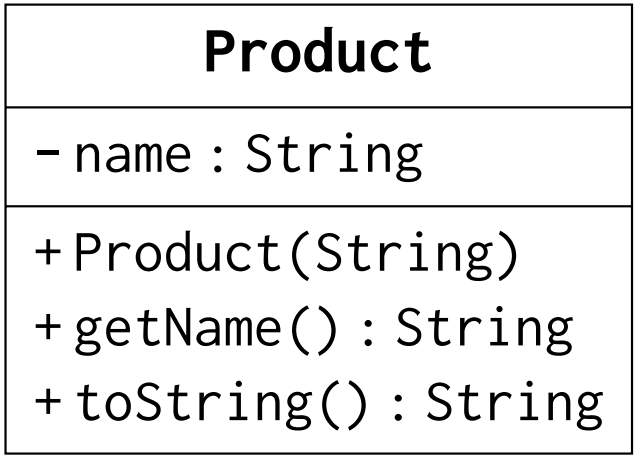
Un programme indenté


```
1  /**
2  * Version simple de la classe Product. Un produit possède un nom. Il
3  * est possible de récupérer ce nom et de transformer un produit en
4  * chaîne de caractères .
5  * @authors F.Dagnat
6  */
7  public class Product {
8    /** le nom du produit sous forme d'une chaîne de caractères */
9    private String name;
10   /** un constructeur qui prend en paramètre le nom du nouveau produit */
11   public Product(String name) { this.name = name; }
12   /** rend une chaîne de caractères qui est le nom du produit */
13   public String getName() { return name; }
14   /** rend une chaîne de caractères qui décrit le produit */
15   public String toString() { return "Produit " + name; }
16   /** Une méthode main qui teste cette classe */
17   public static void main(String[] args) {
18     Product p1 = new Product("p1");
19     System.out.println(p1);
20     new Product("");
21     Product p3 = new Product("p3");
22     System.out.println(p3);
23     System.out.println("Le nom de p3 est " + p3.getName());
24   }
25 }
```

Un programme indenté et coloré


```java
/**
* Version simple de la classe Product. Un produit possède un nom. Il
* est possible de récupérer ce nom et de transformer un produit en
* chaîne de caractères .
* @authors F.Dagnat
*/
public class Product {
  /** le nom du produit sous forme d'une chaîne de caractères */
  private String name;
  /** un constructeur qui prend en paramètre le nom du nouveau produit */
  public Product(String name) { this.name = name; }
  /** rend une chaîne de caractères qui est le nom du produit */
  public String getName() { return name; }
  /** rend une chaîne de caractères qui décrit le produit */
  public String toString () { return "Produit " + name; }
  /** Une méthode main qui teste cette classe */
  public static void main(String[] args) {
    Product p1 = new Product("p1");
    System.out.println(p1);
    new Product("");
    Product p3 = new Product("p3");
    System.out.println(p3);
    System.out.println("Le nom de p3 est " + p3.getName());
  }
}
```

```
1  /**
2  * Version simple de la classe Product. Un produit possède un nom. Il
3  * est possible de récupérer ce nom et de transformer un produit en
4  * chaîne de caractères .
5  * @authors F.Dagnat
6  */
7  public class Product {
8    /** le nom du produit sous forme d'une chaîne de caractères */
9    private String name;
10   /** un constructeur qui prend en paramètre le nom du nouveau produit */
11   public Product(String name) { ... }
12   /** rend une chaîne de caractères qui est le nom du produit */
13   public String getName() { ... }
14   /** rend une chaîne de caractères qui décrit le produit */
15   public String toString () { ... }
16   /** Une méthode main qui teste cette classe */
17   public static void main(String[] args) { ... }
18 }
```


Product
- name : String
+ Product(String)
+ getName() : String
+ toString() : String

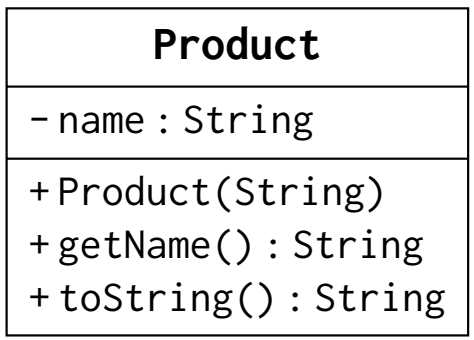

Product

Analyse
(Comprendre le problème)

Conception
(Trouver une solution au problème)

Réalisation
(de la solution)

Product

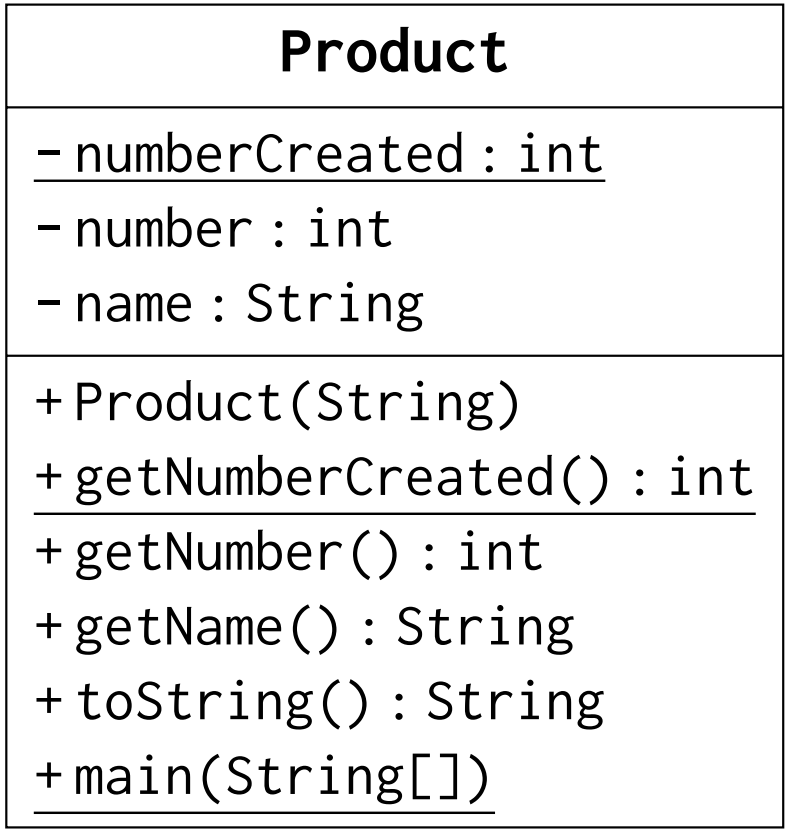



```java
/**
 * Version simple de la classe Product. Un produit possède un nom. Il
 * est possible de récupérer ce nom et de transformer un produit en
 * chaîne de caractères .
 * @authors F.Dagnat
 */
public class Product {
  /** le nom du produit sous forme d'une chaîne de caractères */
  private String name;
  /** un constructeur qui prend en paramètre le nom du nouveau produit */
  public Product(String name) { this.name = name; }
  /** rend une chaîne de caractères qui est le nom du produit */
  public String getName() { return name; }
  /** rend une chaîne de caractères qui décrit le produit */
  public String toString() { return "Produit " + name; }
  /** Une méthode main qui teste cette classe */
  public static void main(String[] args) {
    Product p1 = new Product("p1");
    System.out.println(p1);
    new Product("");
    Product p3 = new Product("p3");
    System.out.println(p3);
    System.out.println("Le nom de p3 est " + p3.getName());
  }
}
```

Pour réfléchir, les diagrammes ; pour réaliser, le code

Abstraction	Boîte	$>$	Java
Lisibilité	Boîte	$>$	Java
Facilité à écrire	Boîte	$>$	Java
Facilité à manipuler	Boîte	$>$	Java
Détails	Java	$>$	Boîte
Précisions	Java	$>$	Boîte
Exécutable	Java	$>$	Boîte

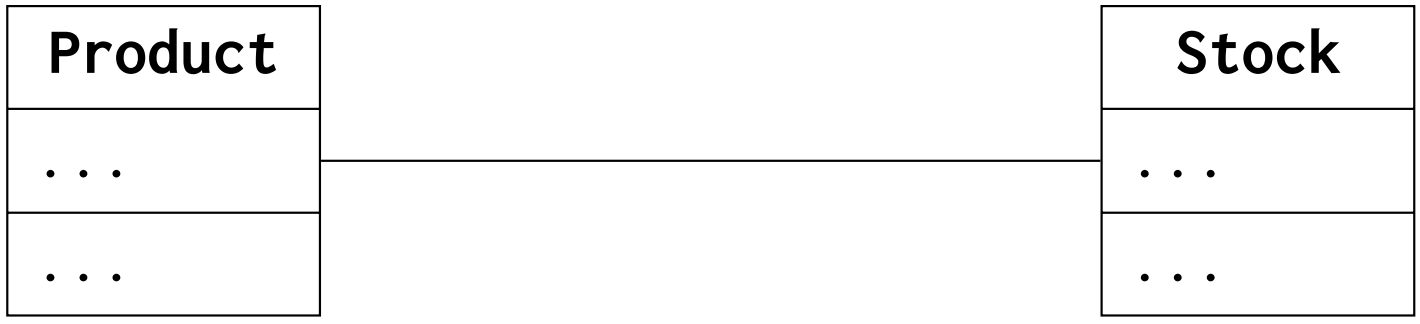
UML (*Unified Modeling Language*) permet de faire des « plans » de logiciel.


Product
- numberCreated : int
- number : int
- name : String
+ Product(String)
+ getNumberCreated() : int
+ getNumber() : int
+ getName() : String
+ toString() : String
+ main(String[])

Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)

Comment exprimer qu'un Produit est dans un Stock ?

Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)

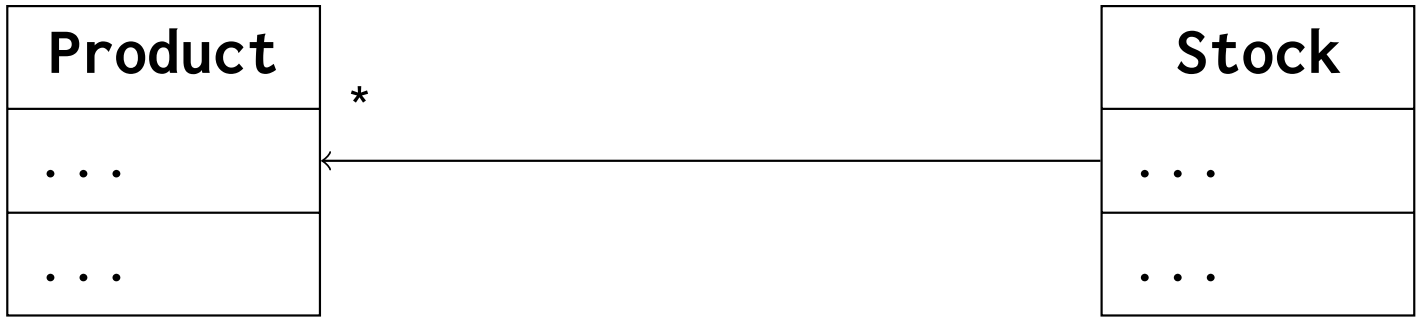


Comment exprimer qu'un Produit est dans un Stock ?

Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)

Comment exprimer qu'un Produit est dans un Stock ?

Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)



et qu'un Stock peut contenir plusieurs Produits ?

Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)

Comment exprimer qu'un Produit est dans un Stock ?
et qu'un Stock peut contenir plusieurs Produits ?

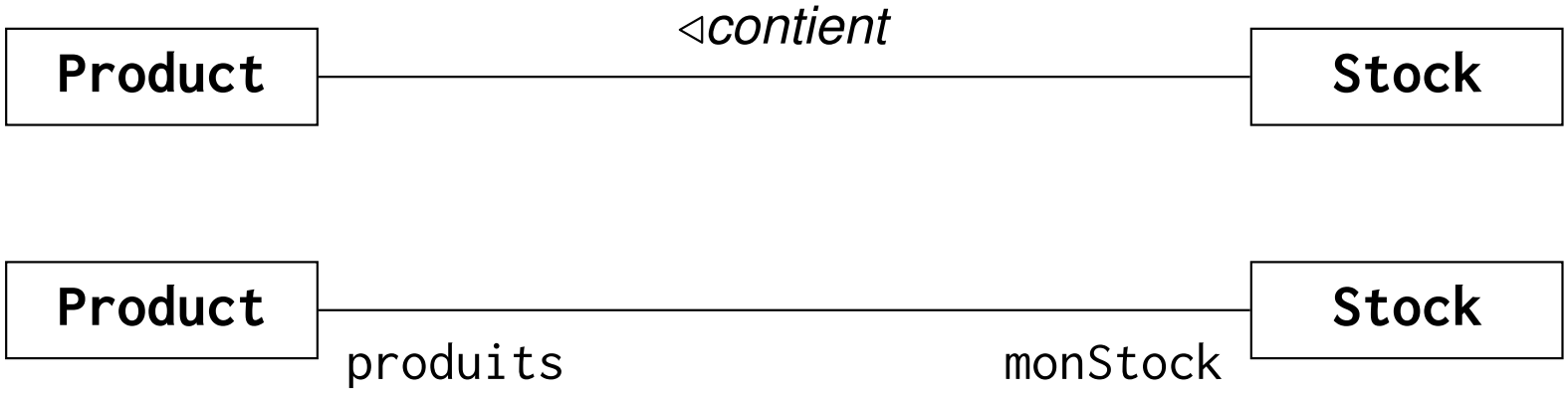
Un produit est dans un stock (java)

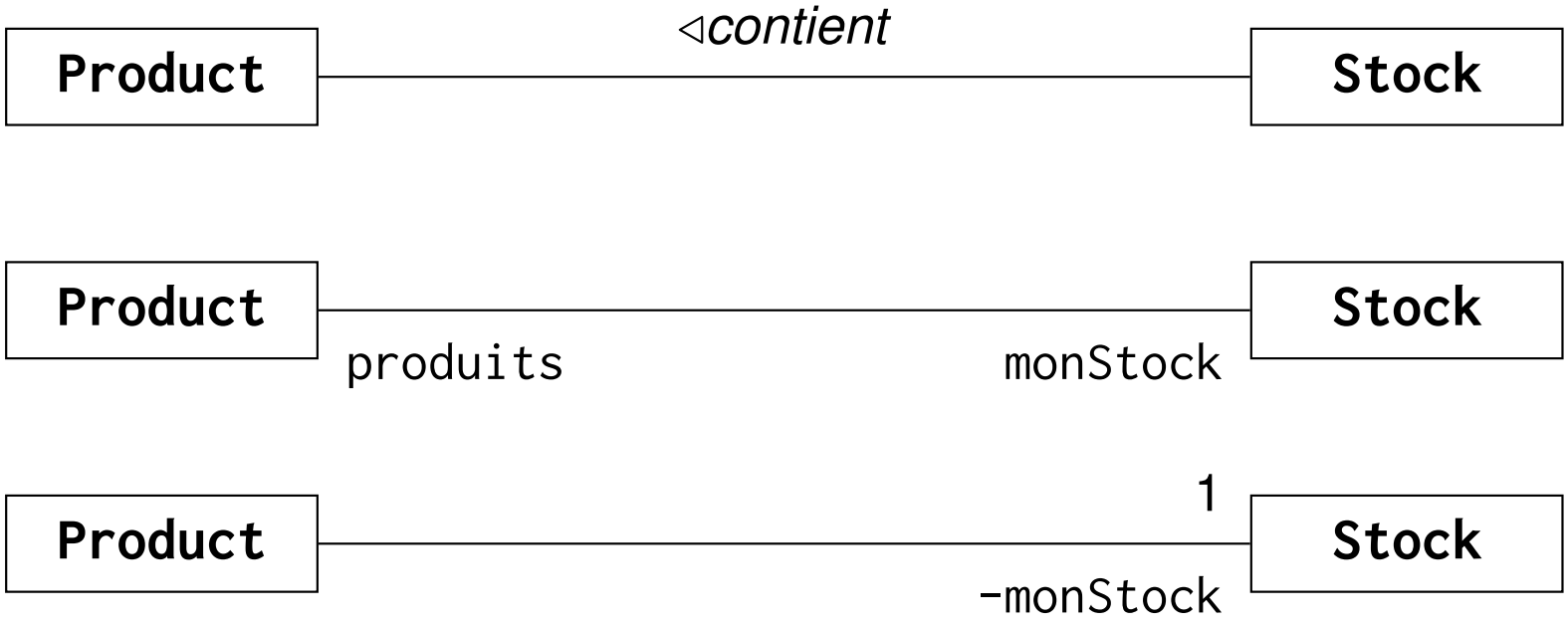

```java
public class Product {
  /** le nom du produit sous forme d'une chaîne de caractères */
  private String name;
  /** le stock dans lequel je suis rangé */
  private Stock monStock;
  /** un constructeur qui prend en paramètre le nom du nouveau produit */
  public Product(String name) { ... }
  /** rend une chaîne de caractères qui est le nom du produit */
  public String getName() { ... }
  /** Les méthodes pour manipuler monStock */
  // ...
  /** rend une chaîne de caractères qui décrit le produit */
  public String toString() { ... }
  /** Une méthode main qui teste cette classe */
  public static void main(String[] args) {...}
}
```

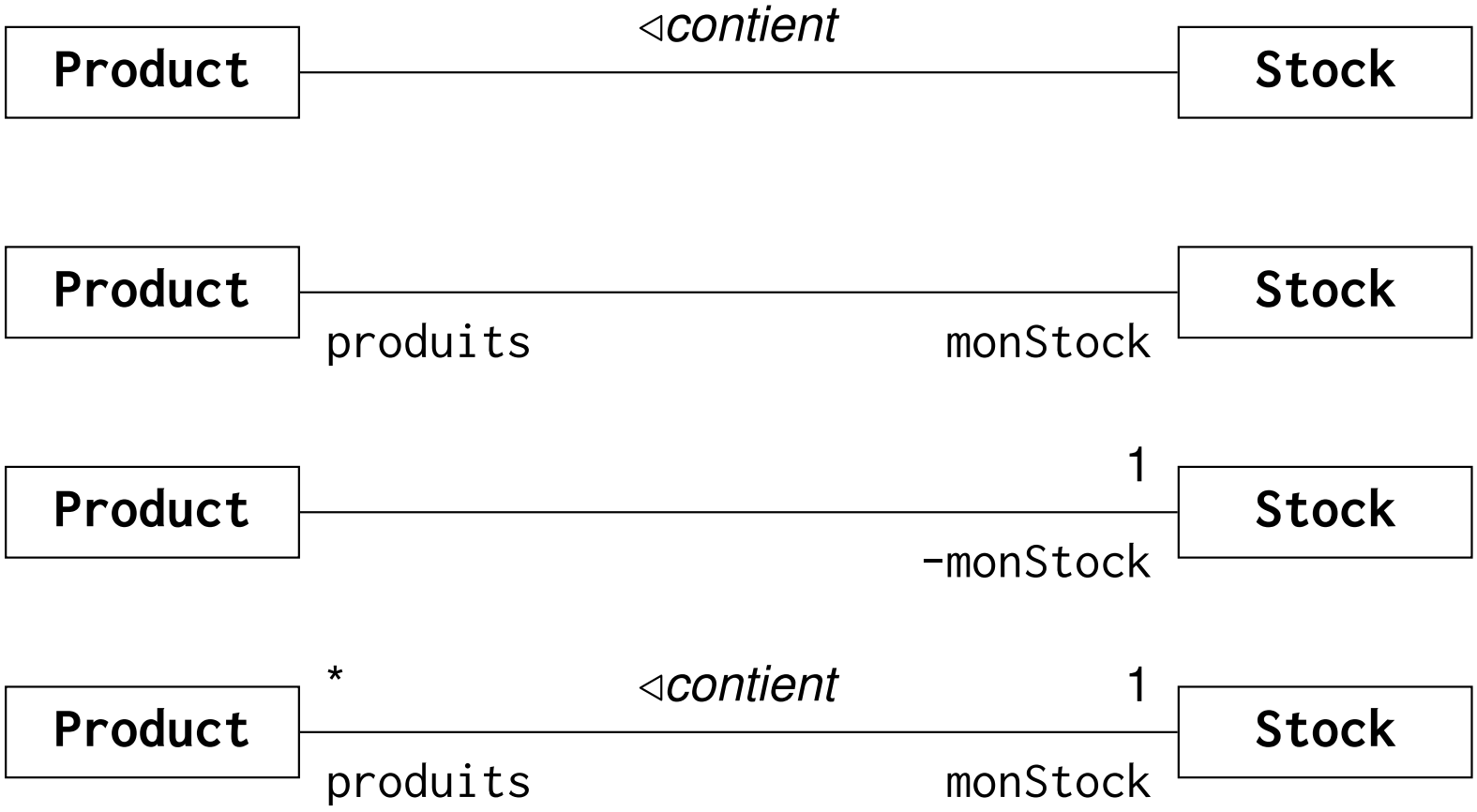
Un stock contient des produits (java)

```
1  public class Stock {
2    /** le tableau contenant les produits */
3    private Product[] produits;
4    /** le nombre de produits déposés */
5    private int size = 0;
6    /** un constructeur avec comme paramètre la tail le du stock
7    * @param s la tail le du stock */
8    public Stock(int s) { produits = new Product[s]; }
9    /** rajoute un nouveau produit dans le stock
10   * @param p le produit qui est rajouté */
11   public void add(Product p){
12     if (p==null) return;
13     produits[size++] = p;
14   }
15 }
```

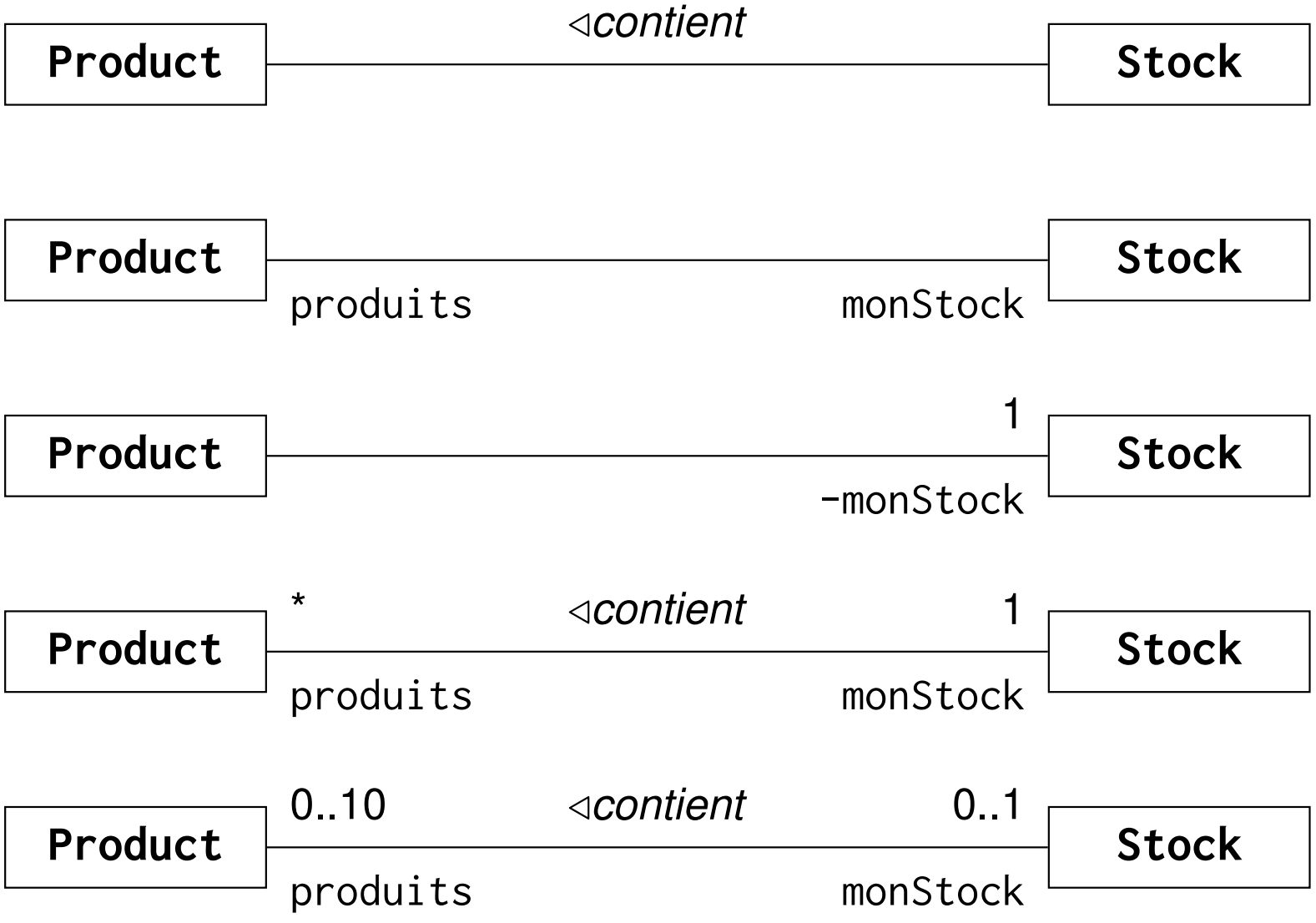

◁contient
Product
Stock


◁contient
Product
Stock
Product
Stock
produits
monStock


Product
◁contient
Stock
Product
produits
monStock
Stock
Product
1
-monStock
Stock


Product
◁contient
Stock
Product
produits
monStock
Stock
Product
1
-monStock
Stock
Product
*
◁contient
1
produits
monStock
Stock

De nombreuses variantes

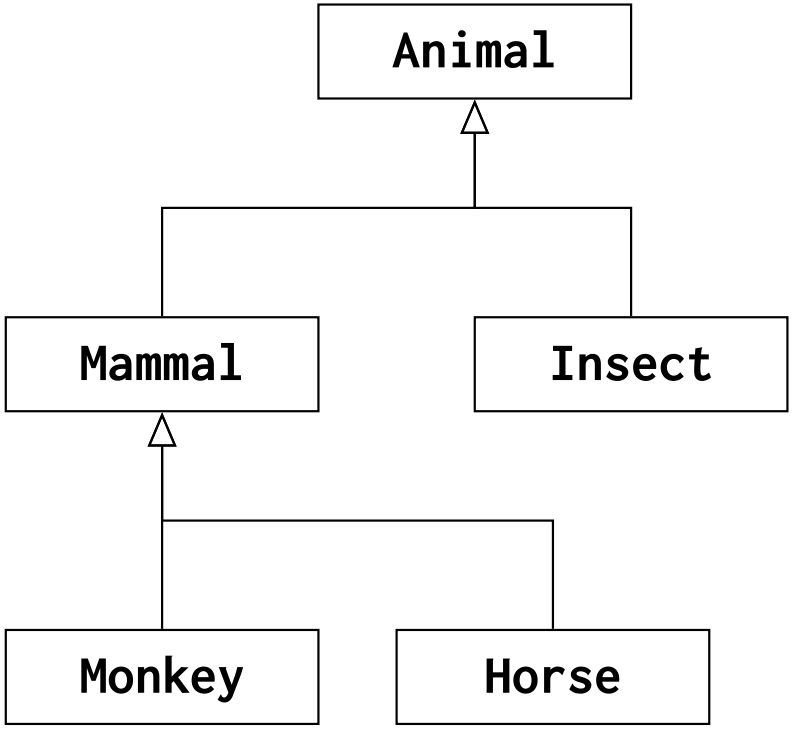


Les associations n'existent pas en Java.

Elles sont traduites par un *attribut*

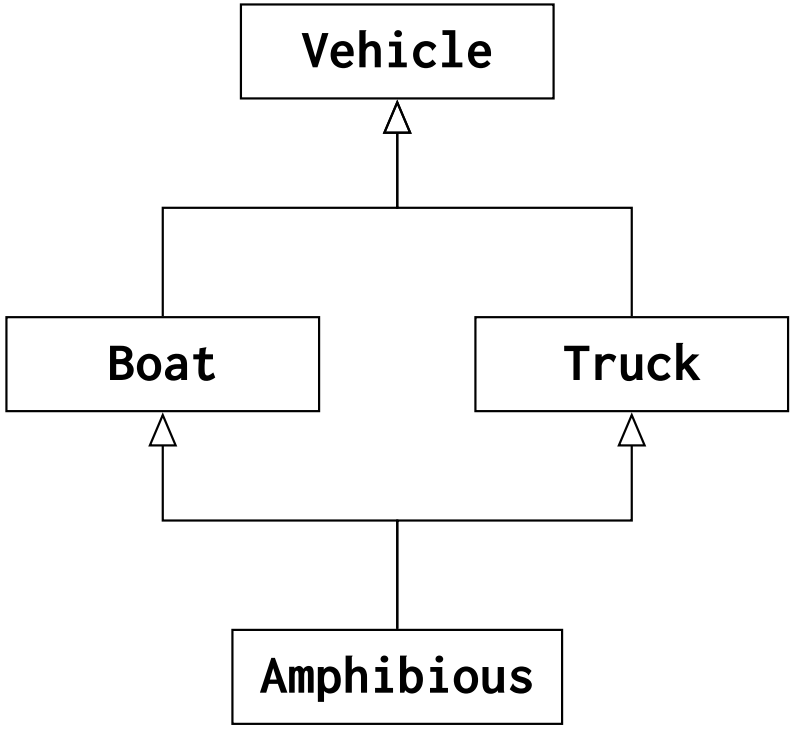
- qui référence une classe simple pour les cardinalités 0..1 ou 1
- qui référence un tableau ou une liste pour les cardinalités $>$1

Héritage simple (Java, ...)



Arbre d'héritage

Héritage multiple (UML, Python, ...)



Graphe d'héritage

Ne vous lancez pas dans la programmation trop rapidement.

- Faites des dessins (des plans)
- Réfléchissez
- Étudiez des variantes

L'héritage permet d'organiser et de réutiliser.

La liaison dynamique est le mécanisme qui rend la programmation objet plus réutilisable.