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

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Diagramme de classes UML

Concevoir la structure des classes

A. Beugnard & J.C. Royer

Diaporama – UE-IR-S6 –
Session 3– 1er semestre 2026

Avancement

Un programme

```
/**
* 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());
}
}
```

Un programme indenté

```
/**
* 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());
  }
}
```

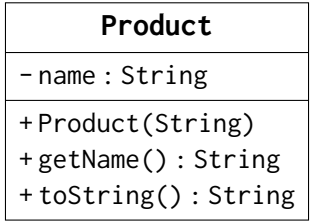
Un programme indenté et coloré

```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());
  }
}
```

Un programme simplifié

```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) { ... }
  /** rend une chaîne de caractères qui est le nom du produit */
  public String getName() { ... }
  /** 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 saut qualitatif



Encore plus simple

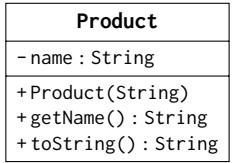
Product

Représentation et cycle de vie

Analyse

(Comprendre le problème)

Conception (Trouver une solution au problème)



Réalisation (de la solution)

```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());
  }
}
```

Boîte (UML) ou code ?

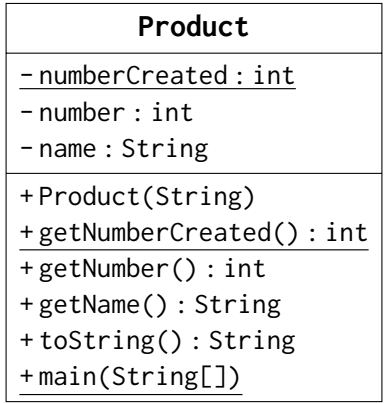
Pour réfléchir, les diagrammes ; pour réaliser, le code

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.

Avancement

Des classes en UML



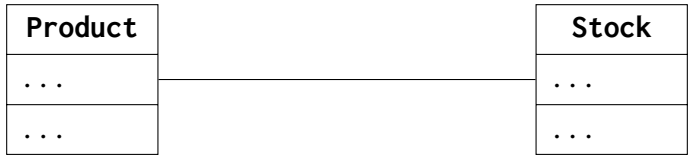
Relier des classes en UML : association

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 ?

Relier des classes en UML : association

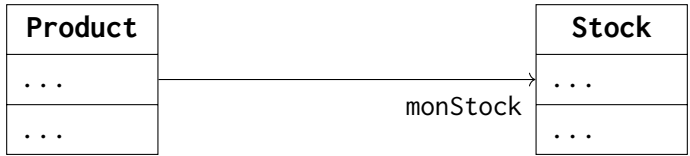
Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)



Comment exprimer qu'un Produit est dans un Stock ?

Relier des classes en UML : association

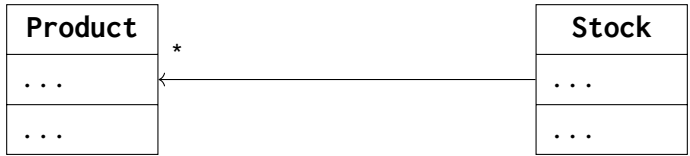
Association : lien entre classes qui dure dans le temps. Lien structurel entre classes (au sens « a un »)



Comment exprimer qu'un Produit est dans un Stock ?

Relier des classes en UML : association

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 ?

Relier des classes en UML : association

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 ?

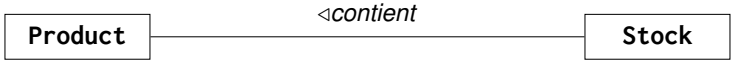
Un produit est dans un stock (java)

```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) {...}
}
```

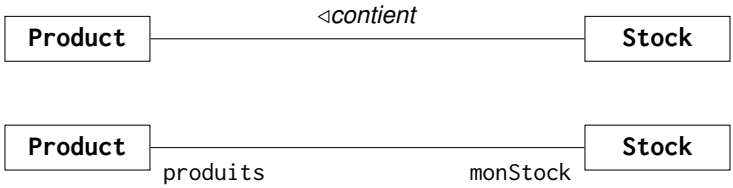
Un stock contient des produits (java)

```java
public class Stock {
  /** le tableau contenant les produits */
  private Product[] produits;
  /** le nombre de produits déposés */
  private int size = 0;
  /** un constructeur avec comme paramètre la tail le du stock
  * @param s la tail le du stock */
  public Stock(int s) { produits = new Product[s]; }
  /** rajoute un nouveau produit dans le stock
  * @param p le produit qui est rajouté */
  public void add(Product p){
    if (p==null) return;
    produits[size++] = p;
  }
}
```

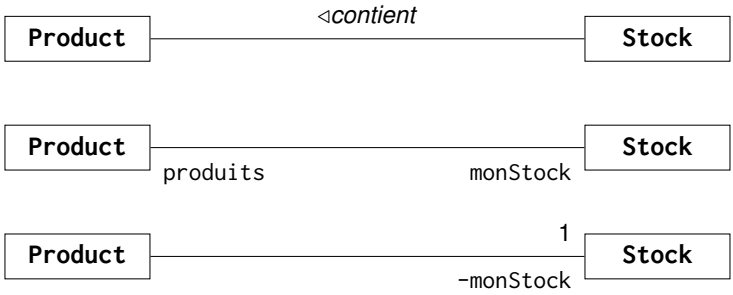
De nombreuses variantes



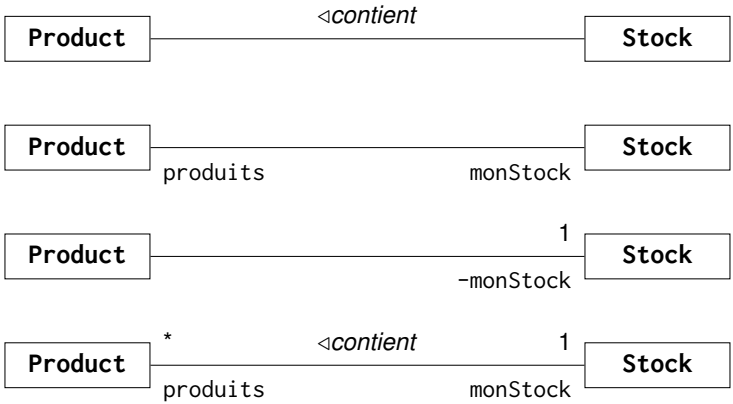
De nombreuses variantes



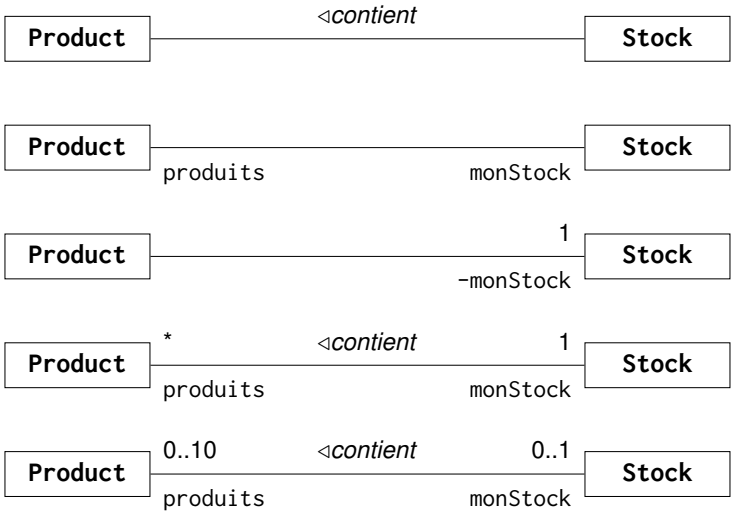
De nombreuses variantes



De nombreuses variantes



De nombreuses variantes



Association et Java

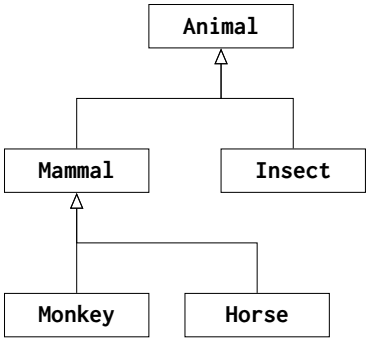
Les associations n'existent pas en Java.

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

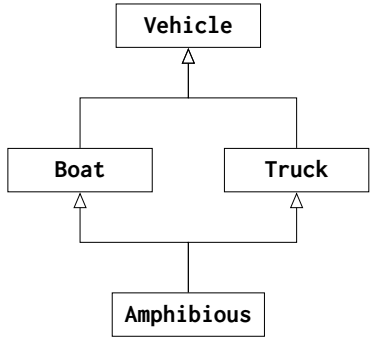
Différents héritages

Héritage simple (Java, ...)



Arbre d'héritage

Héritage multiple (UML, Python, ...)



Graphe d'héritage

Avancement

Conclusion

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.