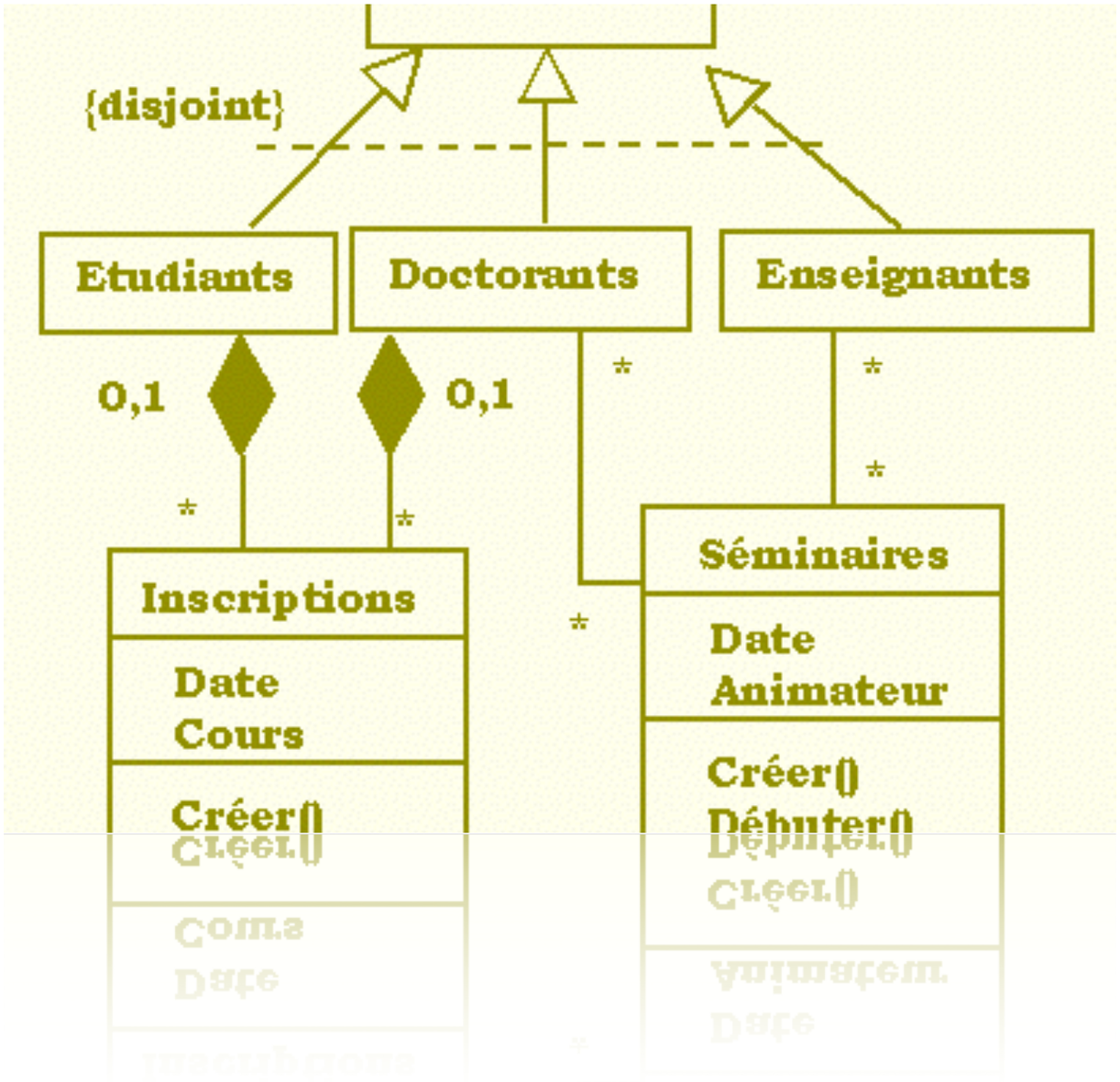
ITC313 - Informatique

Partie C / C++
Héritage multiple

Benoît DARTIES

Benoit.Darties@u-bourgogne.fr



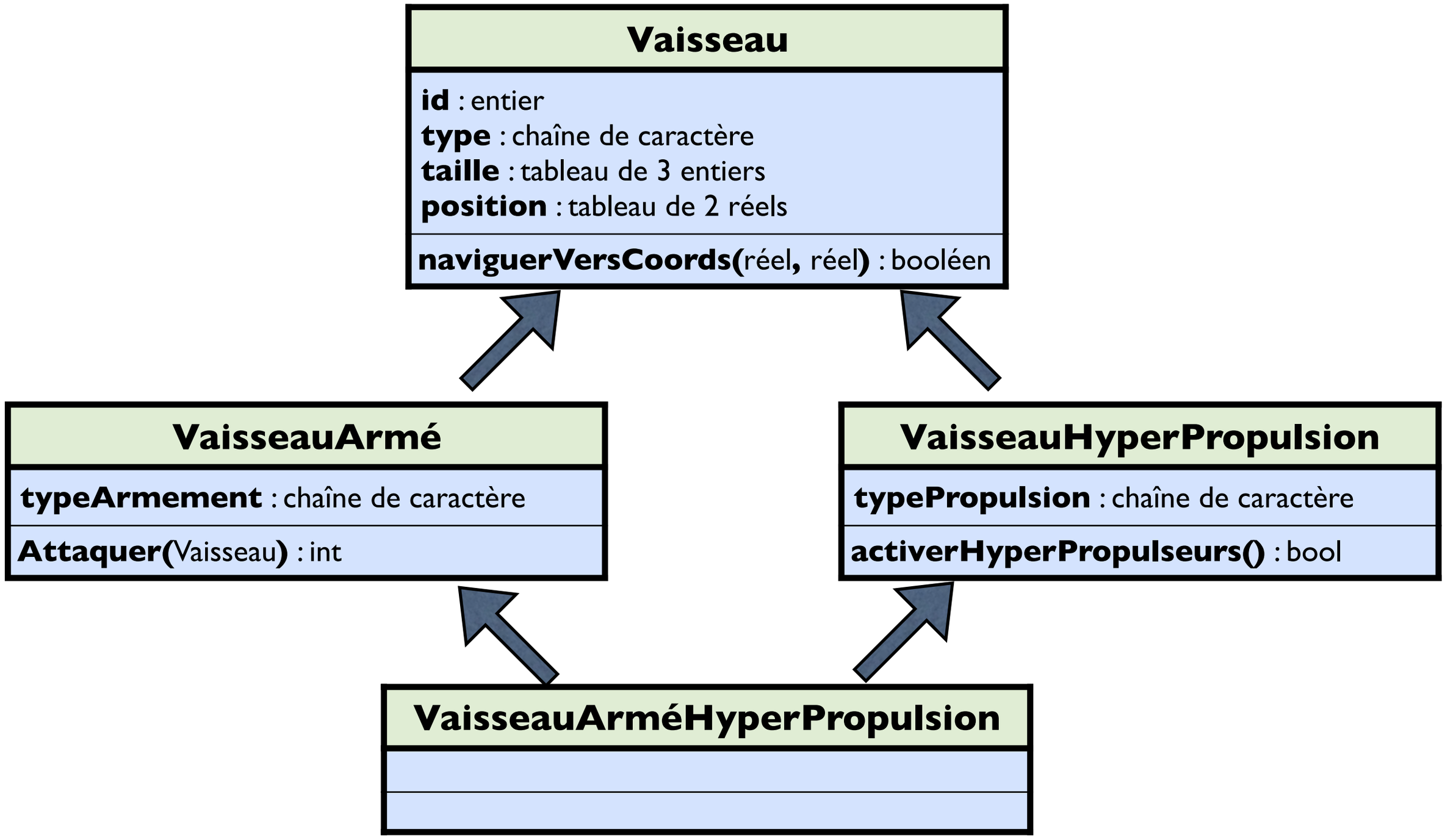
Héritage multiple

Héritage multiple

- Intérêt de l'héritage multiple :
  - Créer des classes d'objets dérivées de plusieurs classes de base
  - Récupération des attributs / méthodes des différentes classes héritées
  - Factorisation de code accrue

- Difficultés :
  - L'ordre d'héritage a son importance
  - Apparition de conflits :
    - méthodes héritées par plusieurs classes mères
    - méthodes (re)-définies dans plusieurs classes mères

Heritage multiple



Heritage multiple

- Héritage multiple et syntaxe :
  - Liste des classes mères précédées sur spécificateur d'héritage, et séparées par des point-virgules.
  - Importance de l'ordre des classes!
    - Impact sur l'ordre d'appel des constructeurs des classes mères

```
class nomClasseFille : public nomClasseMere1, public nomClasseMere2 {
  ...
};

// DIFFERENT DE

class nomClasseFille : public nomClasseMere2, public nomClasseMere1 {
  ...
};
```

Heritage multiple et constructeurs

- Impact de l'ordre d'annonce des classes mères :

```
class VaisseauArmeHyperPropulsion : public VaisseauArme, public
   VaisseauHyperPropulsion {...};
```

  - Lors de l'instanciation de la classe :
    - **a) Création d'un VaisseauArme :**
      - ➡ Constructeur **`VaisseauArme()`**
    - **b) Création d'un VaisseauHyperPropulsion :**
      - ➡ Constructeur **`Vaisseau()`**
      - ➡ Constructeur **`VaisseauHyperPropulsion()`**
    - **c) Création d'un VaisseauArmeHyperPropulsion :**
      - ➡ Constructeur **`VaisseauArmeHyperPropulsion()`**

Heritage multiple et constructeurs

- Impact de l'ordre d'annonce des classes mères :

```
class VaisseauArmeHyperPropulsion : public VaisseauHyperPropulsion,
   public VaisseauArme { ...};
```

  - Lors de l'instanciation de la classe :
    - **a) Création d'un VaisseauHyperPropulsion :**
      - ➡ Constructeur **`Vaisseau()`**
      - ➡ Constructeur **`VaisseauHyperPropulsion()`**
    - **b) Création d'un VaisseauArme :**
      - ➡ Constructeur **`Vaisseau()`**
      - ➡ Constructeur **`VaisseauArme()`**
    - **c) Création d'un VaisseauArmeHyperPropulsion :**
      - ➡ Constructeur **`VaisseauArmeHyperPropulsion()`**

- Influence de l'ordre d'appel des constructeurs!
  - Par ex : **`VaisseauHyperPropulsion()`** et **`VaisseauArme()`** initialisent un attribut de **`Vaisseau()`** avec une valeur différente.

Héritage multiple et constructeurs

- Comportement par défaut et modification :
  - Constructeurs des classes mère sans paramètres utilisés
  - Possibilité de préciser les constructeurs des classes mère à employer
    - Principe identique à l'héritage simple
    - Liste des constructeurs à appeler séparés par une virgule
    - ordre d'appel **indépendant** de l'ordre d'héritage de classes
- Exemple :

```
class VaisseauArmeHyperPropulsion : public VaisseauArme, public
    VaisseauHyperPropulsion {
    ...

    public VaisseauArmeHyperPropulsion(String id) : VaisseauArme(id),
    VaisseauHyperPropulsion() {
       ...
    }
};
```

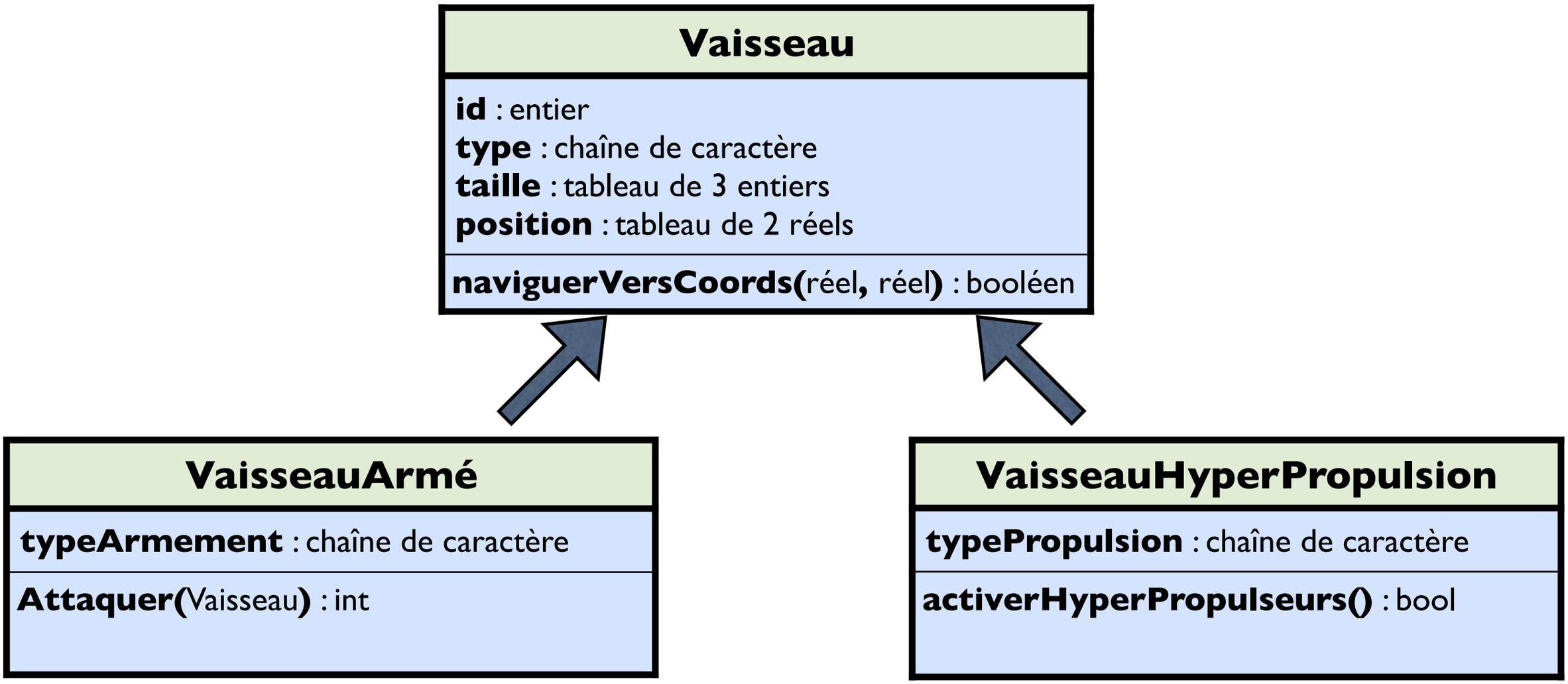
Heritage multiple et destructeurs

- Destruction d'un objet :
  - ordre inverse de déclaration des classes mères

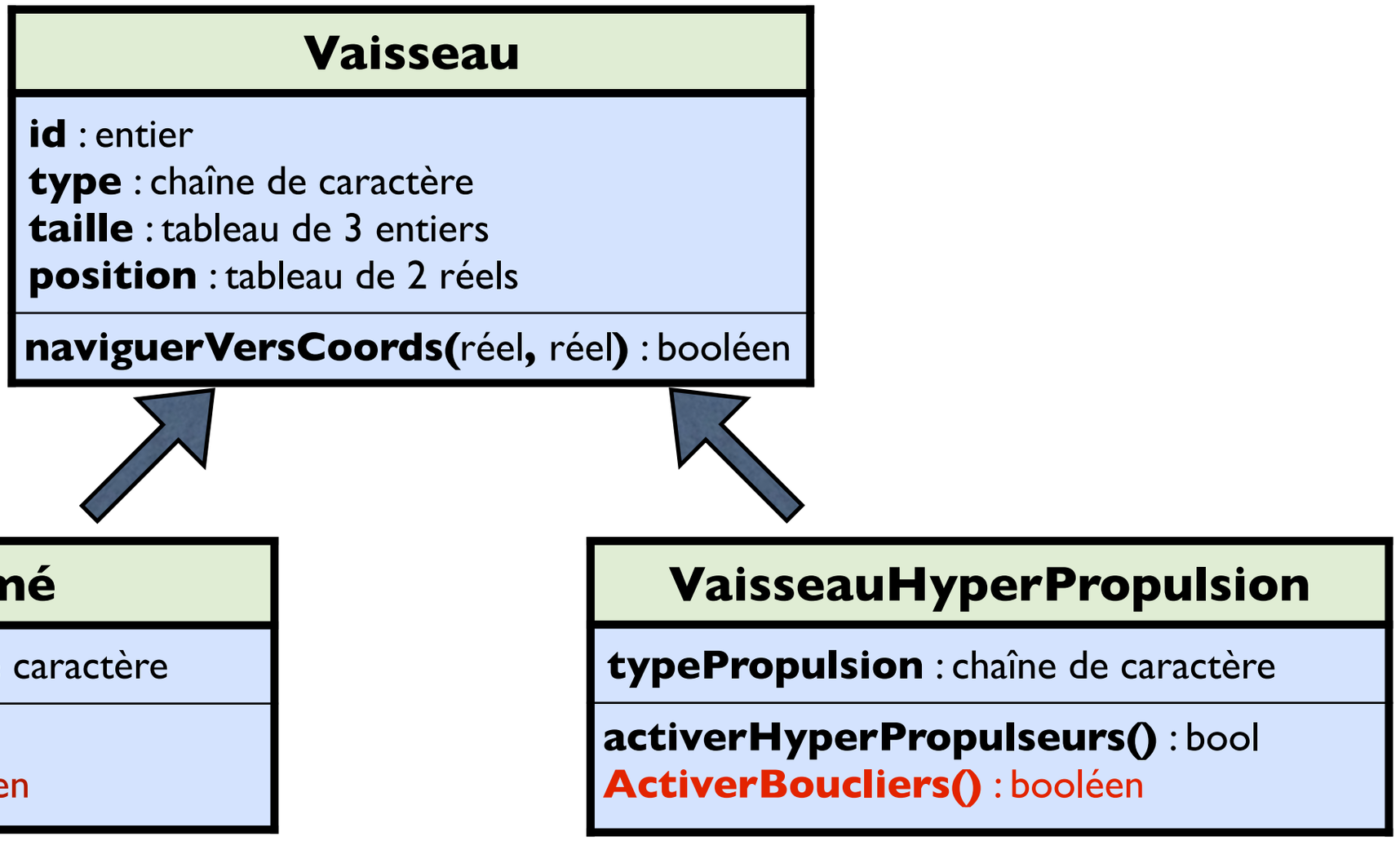
```
class VaisseauArmeHyperPropulsion : public VaisseauArme, public
   VaisseauHyperPropulsion { ...};
```

  - Lors de la destruction d'un objet de cette classe :
    - **a) Destruction d'un VaisseauArmeHyperPropulsion :**
      - ➡ Destructeur **`~VaisseauArmeHyperPropulsion()`**
    - **b) Destruction d'un VaisseauHyperPropulsion :**
      - ➡ Destructeur **`~VaisseauHyperPropulsion()`**
      - ➡ Destructeur **`~Vaisseau()`**
    - **Destruction d'un VaisseauArme :**
      - ➡ Destructeur **`~VaisseauArme()`**
      - ➡ Destructeur **`~Vaisseau()`**

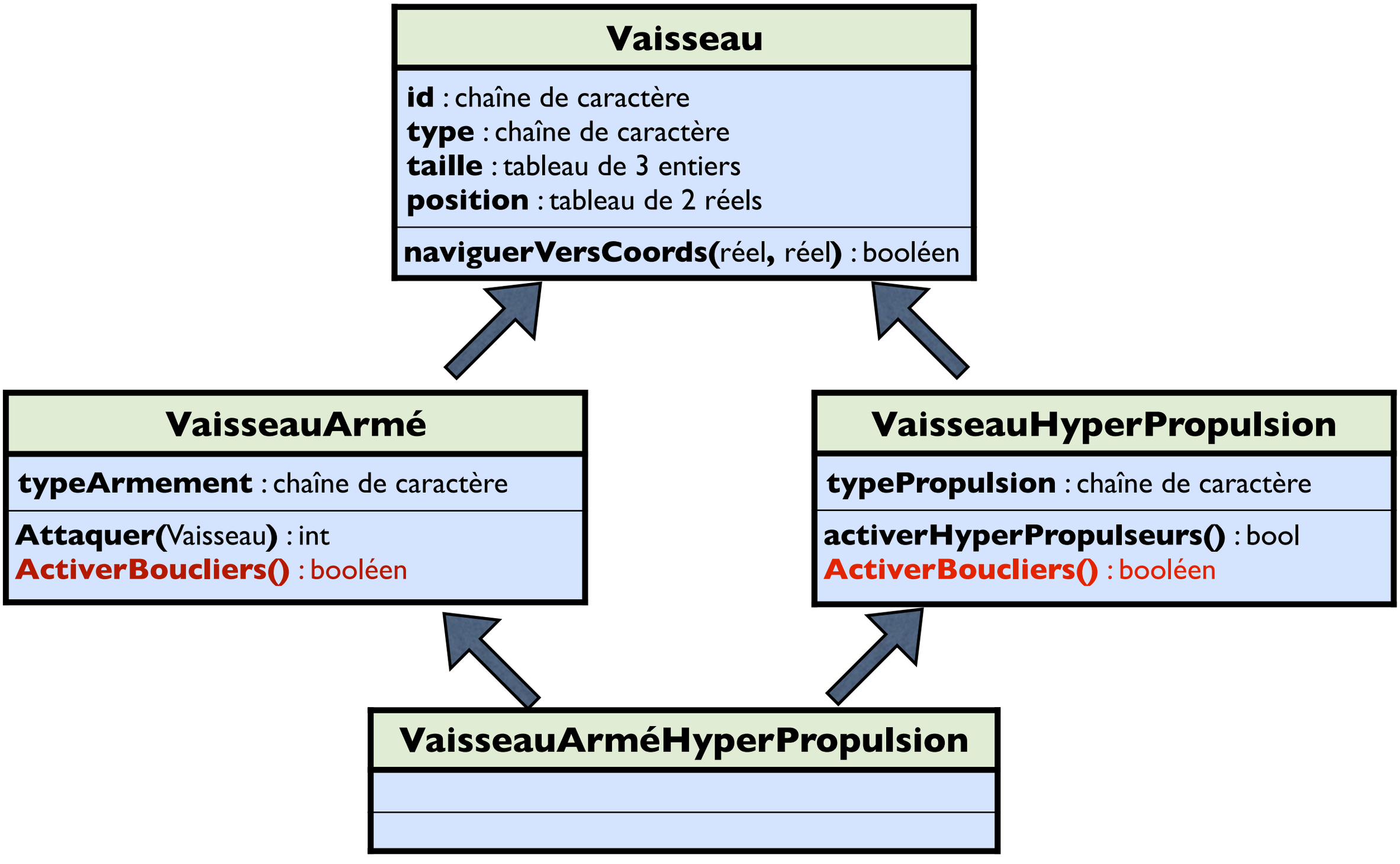
Heritage multiple et ambiguïtés



Heritage multiple et ambiguïtés



Heritage multiple et ambiguïtés



Heritage multiple et ambiguïtés

- Problème :
  - héritage concurrent de deux méthodes de même signature
  - les deux méthodes sont héritées!

```
int main() {
  VaisseauArmeHyperPropulsion Galactica;
  Galactica.activerBoucliers();
}
```

- Ambiguïté détectée à la compilation

```
 # g++ programme.cpp
programme.cpp:44: error: ‘activerBoucliers()’ is ambiguous
candidates are:
 bool VaisseauArme::activerBoucliers()
 bool VaisseauHyperPropulsion::activerBoucliers()
```

Heritage multiple et ambiguïtés

- Résolution de l'ambiguïté sur héritage concurrent
  - Solution 1 : utilisation de l'opérateur de résolution de portée **::**
    - annonce explicite de la classe mère à utiliser

```
int main() {
 VaisseauArmeHyperPropulsion Galactica;

 // méthode de VaisseauArme
 Galactica.VaisseauArme::activerBoucliers();

 // méthode de VaisseauHyperPropulsion
 Galactica. VaisseauHyperPropulsion::activerBoucliers();
 }
```

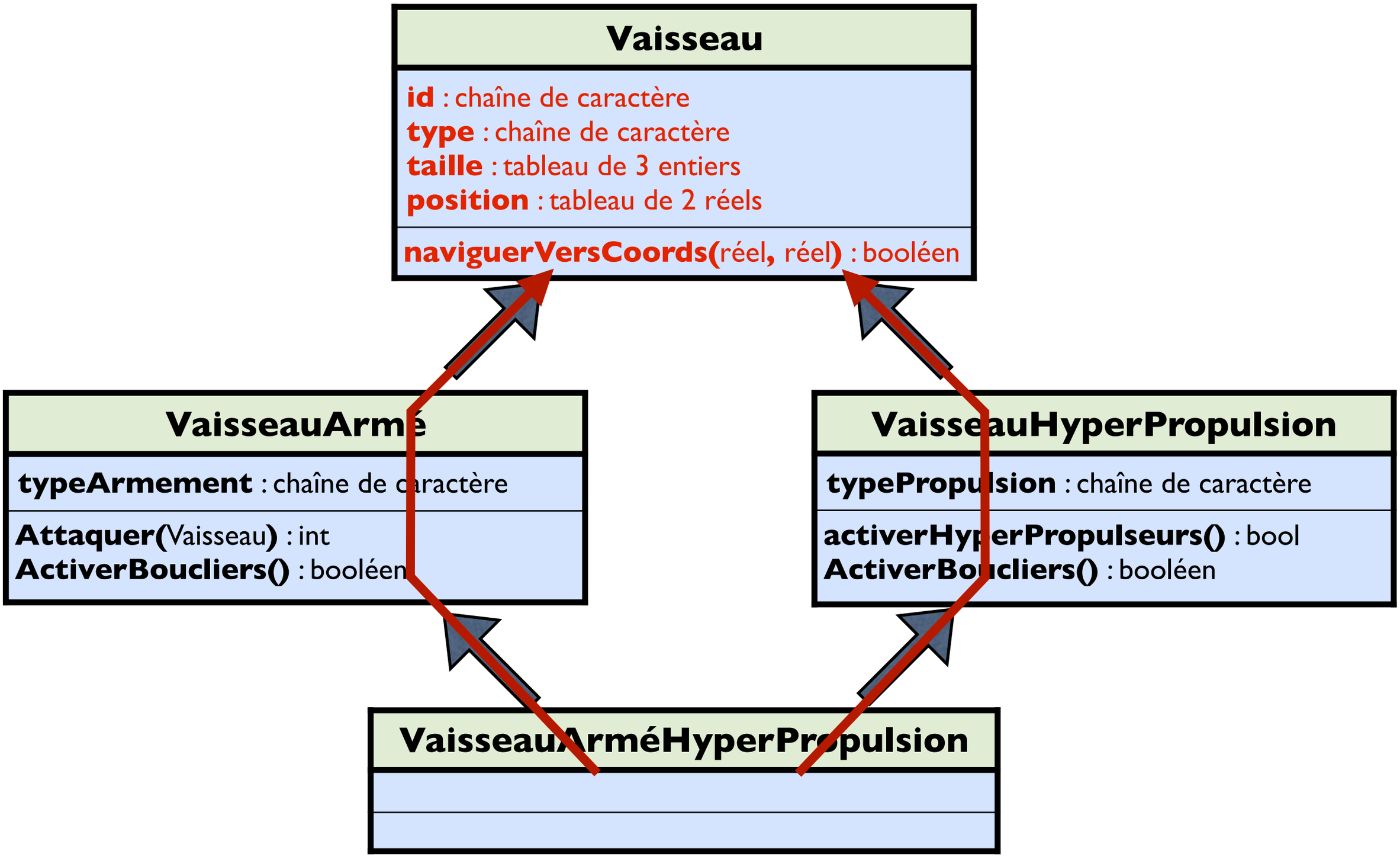
Heritage multiple et ambiguïtés

- Résolution de l'ambiguïté :
  - Solution 2 : redéfinition de la méthode ambigüe
    - Possibilité d'appeler la méthode d'une classe mère

```
class VaisseauArmeHyperPropulsion : public VaisseauArme,
 public VaisseauHyperPropulsion {
 ...
 public :
   bool activerBoucliers() {

     // exemple : privilégier la méthode de HyperPropulsion
     return VaisseauHyperPropulsion::activerBoucliers();
   }
 };
```

Heritage multiple et ambiguïtés



Heritage multiple et ambiguïtés

- Problème :
  - attributs/méthodes hérités de multiples façons

```
int main() {
  VaisseauArmeHyperPropulsion Basestar;
  Basestar.naviguerVersCoords(10.0, 93.2);
}
```

- Ambiguïté détectée à la compilation

```
 # g++ programme.cpp
programme.cpp:44: error: ‘naviguerVersCoords()’ is ambiguous
candidates are:
 bool Vaisseau::naviguerVersCoords()
 bool Vaisseau::naviguerVersCoords()
```

Heritage multiple et ambiguïtés

- Résolution de l'ambiguïté sur héritage multiple d'un élément
  - opérateur de résolution de portée `::` inefficace !
    - L'élément appartient doublement à la classe Vaisseau
    - Plusieurs chemins pour accéder à l'élément
  - Solution : **virtualiser** l'héritage depuis la classe contenant l'élément
    - L'attribut / méthode n'est plus hérité en soi
    - Remplacé par un pointeur sur l'attribut / méthode
    - Garantie de l'unicité de l'élément

```
class nomClasseFille : virtual public nomClasseMere() {
   ...
}
```

Heritage multiple et ambiguïtés

- Résolution de l'ambiguïté sur héritage multiple d'un élément

```
class Vaisseau { ...};
```

```
class VaisseauArme : virtual public Vaisseau { ... };
```

```
class VaisseauHyperPropulsion : virtual public Vaisseau
{ ... };
```

```
class VaisseauArmeHyperPropulsion : public VaisseauArme,
 public VaisseauHyperPropulsion {  ... }
```

Inutile ici